MASTER'S THESIS

The effect of efficacious alignment on Pair Programming in large scale agile environments

Lambrechts, G

Award date:
2022

The effect of efficacious alignment on Pair Programming in large scale agile environments

Program:	Open University of the Netherlands, faculty of Science
	Master of Science Business Process Management & IT
Course:	IM0602 Voorbereiden Afstuderen BPMIT IM9806 Afstudeeropdracht Business Process Management and IT
Student:	Lambrechts Gregory
Identification:	
Date:	27-Jun-2022
Tutor:	Pien Walraven
Second reader:	Rogier van de Wetering
Version:	1.0
Status:	Report

Abstract

The adoption of agile practices in many large organizations is a logical result of its proven adaptability in software development. However, the initial design was not meant to operate in a larger environment with many dependencies, but to be used in a small and independent team with decision-making power. Subsequently, agile at a larger scale comes with its own series of challenges to overcome as they are hard to control by top down enforcement and should be engaged with interactions that also inform, legitimate and socialize. Using the complex adaptive system based model of Co-evolutionary Information System Alignment (COISA), this study aims to uncover from which alignment facilitators the usage of pair programming interactions in large scale agile environment can be influenced. COISA is a recent approach to understand business and IT alignment in complex environments, such as the case presented in this study. This study shows several effects from various facilitators presented in the COISA model by collecting data from participants that have engaged in such sessions in a large scale agile environment. Pair programming remains an interaction between technically involved stakeholders, as it's only used within the operational context of IT implementations. This however does not mean it cannot influence or trigger other COISA processes such as IT usage or architecture as seen in the results.

Key words

Large-scale Agile, Pair Programming, Co-evolutionary information system alignment, efficacious alignment.

Table of contents

1. Introduction

1.1. Background

There is no question that Information Systems (IS) and Information Technology (IT) play a key part in modern day organizations and as a result have drawn the attention of researchers and practitioners for several decades now (Amarilli et al., 2017; Coltman et al., 2015). Over the years, studies revealed empirical evidence that alignment reveals positive effects on business performance (Amarilli et al., 2017) when key IT resources, IT infrastructure, knowledge assets, technical and managerial IT skills are aligned with business strategy and when appropriate structures are used to supervise the deployment and effective management of those resources (Coltman et al., 2015). However, research also suggests that alignment is still considered an unachieved objective for many organizations (Amarilli et al., 2017; Coltman et al., 2015; L. Kappelman et al., 2014; L. A. Kappelman et al., 2013), as organizations may have to cope with inflexible links between business and IT that impedes or delays the ability to quickly respond to environmental change (Benbya & McKelvey, 2006; Coltman et al., 2015). The reason that alignment is so difficult, is its co-evolutionary and emergent nature (Benbya & McKelvey, 2006) and has inspired researchers to interpret corporate IS as a socio-technical system (Lee et al., 2008). As a result it sparked the appliance of complexity science to the study of IS alignment as a co-evolutionary process (Amarilli et al., 2017) because organizations can be seen as interdependent complex systems (Thompson, 1967). The relevance between complexity science and organizational science that are undergoing transformations related to information systems, is therefore apparent (Allen & Varga, 2006).

Considering this, the complex adaptive system (CAS) based model of co-evolutionary information system alignment (COISA) can be used to serve as a theoretical lens. COISA is defined as a "continuously exercised alignment processes, characterized by co-evolutionary interactions between different Information System (IS) stakeholders, in pursuit of a common interpretation and implementation of what it means to apply IT in an appropriate and timely way, in harmony with strategies, goals and needs" (Walraven et al., 2020, pp3-4). However as investigated by Walraven et al. (2020), it is insufficient only to assess the manifestation of COISA interactions, as it is of no guarantee that these interactions are effective in pursuing alignment. Facilitators for having efficacious alignment can be stakeholder involvement, alignment motivation, interconnections and specific alignment decisions (Walraven et al., 2020). This research will build upon this earlier work by looking for specific practical approaches that benefit from efficacious alignment facilitators for uniting stakeholders within the IT implementations' COISA process. It is a stream of IS research, that addresses the need for a different approach to the alignment challenge given rapid change, unpredictability and blurring boundaries between business and IS functions (Allen & Varga, 2006; Amarilli et al., 2016; Benbya & McKelvey, 2006; Walraven et al., 2018, 2020).

1.2. Problem statement

With rapid change in both customer demands, regulations and technological progress, organizations encounter difficulties in coping with such unstable environments (Dybå & Dingsoyr, 2009; Uludag et al., 2019; Weill & Woerner, 2015). Being able to detect relevant changes and react on them causes software development to be confronted with constant need for change (Uludag et al., 2019). Because of this, agile development methods have gained popularity in organizations for its ability to handle change in fast paced environments and reducing cost (Ali Babar, 2009). And due to the successes of those methods, have inspired many companies to implement them on a larger scale despite being designed for small individual teams (Dikert et al., 2016; Dybå & Dingsoyr, 2009). However, as it is designed to work for small independent development teams, adoption at a larger scale brings in new challenges: inter-team coordination and communication, balancing intentional and emergent architecture or the effective coordination and alignment of multiple agile teams and development activities (Dikert et al., 2016; Ovaska et al., 2003; Uludag et al., 2017, 2019). It means that the larger the organization, the more difficult it will be to introduce agile methods and adopt it at a larger scale within the organization (Dikert et al., 2016; Dybå & Dingsoyr, 2009; Uludag et al., 2017). Agile teams are hard to control, even with reasonable effort, by enforcement and should be complemented with interactions that also inform, legitimate and socialize (Winter 2016) as delegating decision-making power to agile teams facilitates adoption towards existing enterprise architecture (Marks, 2014; Uludag et al., 2019). In essence, small agile teams can't work as independently as they are originally designed because they need to work as a larger whole and need to interact and align with more stakeholders. Not just between business and IT, but within IT development as well with

challenges like lack of training and coaching, difficult interfacing between teams, achievement of technical consistency, different agile interpretations or even agile teams that do not respect the larger company goals (Dikert et al., 2016).

1.3. Research formulation

Both technology- and more traditional companies already use eXtreme Programming (XP) approaches to keep pace with contemporary development cycles of information systems (Canfora et al., 2007; Chen & Rea, 2018; Vanhanen & Mäntylä, 2014). It includes the agile development techniques to work in pairs, where two developers working next to each other on the same problem, on the same workstation. Pair Programming (PP) has also grown as a successful, evidence-based collaborative learning method for programming-related educational courses (Hannay et al., 2009). Additionally, recent findings of Chen & Rea (2018) encourage that pair programming may even be applied in domains that do not necessarily require actual programming, including domains such as networking, analytics, IS strategy, and project management, with students not involved in IS courses showed improved levels of confidence and interaction on non-programming tasks like the before mentioned domains relevant to IS. A good start to use this for the alignment of agile teams would be the setup of a simple pair programming board, who wants to learn from who, to establish a peer teaching and coaching program across different teams (Wolpers, 2019). And as the cost-benefits of pair programming have been contested on in earlier research, efficacious alignment might be an additional motivator to think about when considering the usage of pair programming or facing alignment issues in large scale agile environments. Albeit in this context used as a mixed method, combining both solo and pair programming and not replacing one for the other. This is an important distinction from existing literature as a mixed usage solo and pair programming is rarely investigated (Sun et al., 2015). Additionally, research in pair programming is mainly focused as a collaborative learning technique in educational context or comparing the benefits of solo and pair programming in a professional environment, but not for investigating efficacious alignment facilitators (in large scale agile environments).

Some of the challenges that organizations face while adopting agile on a larger scale are for instance lack of training or coaching, difficult interfacing between teams, different agile interpretations between teams, the achievement of technical consistency or specialized knowledge kept in internal silo's (Dikert et al., 2016). Because of this, having a cross-team, peer-teacher training pair programming as suggested by Wolpers (2019) does seem like a suitable interaction mechanism that can help aligning different agile teams in their IT implementations. It is here where this research aims to uncover alignment facilitators for efficacious pair programming in complex environments such as agile on a larger scale. As a result, we would argue that pair programming could be benefited by facilitators that help reaching alignment between stakeholders within the IT implementations. By looking for efficacious alignment facilitators of the COISA model and see how they contribute to pair programming, we formulate our research question as follows:

> **RQ:** How do efficacious alignment facilitators contribute to pair programming in the IT implementations' alignment process of a large scale agile environment?

This paper is organized in the following manner: Firstly, we will present a theoretical foundation on which this paper is based. Secondly the applied research methodology and data collection substantiated. Thirdly the results will be discussed, followed by our conclusion, limitations and options for further research.

2. Theoretic Framework

2.1. Literature Review

At the centre of this study, Walraven et al.'s (2020, 2019, 2018) research concerning efficacious alignment for Electronic Medical Records is used as a cornerstone. The complex adaptive systems perspective of looking at alignment specifically includes an operational IT implementation as an alignment process, as well as facilitators that can be used in an empirical context. Because of this, it provides a well-suited theoretical lens in an attempt to understand the complexity of IT implementations using agile practices at larger scale and the additional overhead to make sure all those different teams work as a more coherent whole (which logically speaking requires those teams to align with each other). With this in mind, the agile development technique of pair

programming, where developers more intensively interact to solve a task, should benefit from efficacious alignment facilitators. To provide a better overview in placing these different strands of research, all three themes for this research, being 'COISA Facilitators', 'Pair Programming' and 'Large Scale Agile', are depicted in figure 1 below.

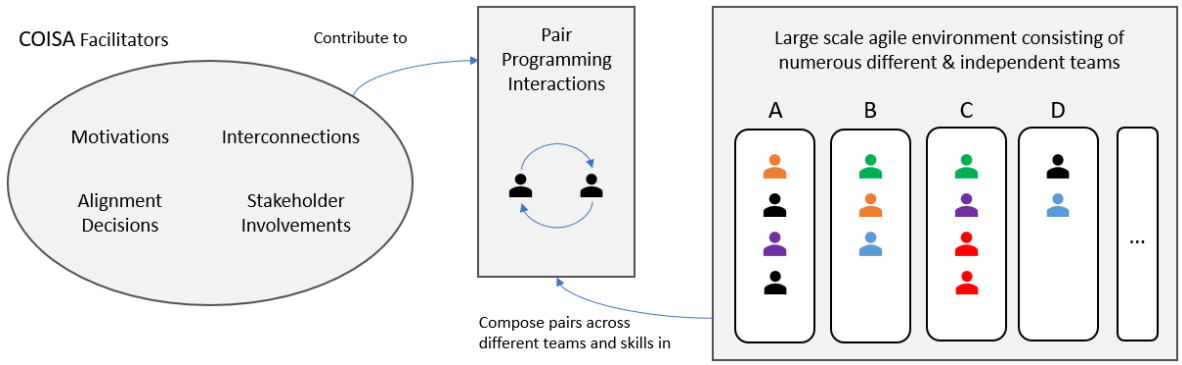

Figure 1. Conceptual model of the theoretical framework.

To provide the appropriate theoretical background for the research question, a literature search is executed using 2 additional identified themes from the formulated research question, namely being 'large scale agile' and 'pair programming' as depicted in Table 1. Using the building blocks method (van Veen & Westerkamp, 2017), the search queries can be narrowed or widened depending on the resulting articles from theme combinations. Various online academic databases are used to execute search queries: Association for Information Systems (AIS), EBSCO, IEEE Digital Library, ScienceDirect, Springer and Google Scholar. Note that articles are only selected if those have been cited at least once. The resulting articles were selected according to specific criteria in the following order:

1. Title relevance judged by whether it contains key words related to the research question.
2. Relevance of the abstract judged by whether it contains key words related to the research.
3. If the first 2 criteria are met, the articles' relevance can be judged by its actual content (results, discussion or industry).

It is important to note that combining more than 2 themes did not properly give relevant search results as literature concerning 'pair programming' is mostly investigated from an educational perspective or comparisons between solo and pair programming, but not from an alignment point of view. This led to the selection of a strand of research concerning the effectiveness of pair programming from the perception of software professionals specifically as investigated by Sun et al. (2015, 2016). Large scale agile in its turn yielded so many results that an additional theme was added to specifically look for large scale agile in its most recent/referenced literature review, which lead to Dikert et al.'s (2016) challenges and success factors in large scale agile environments due to its relevance and high citations.

For the combinations of all 3 themes, public search engines such as Google were also used to find (recent) non-academic literature. This specifically resulted in a hands-on guide for agile transformations, written by professional authors with numerous years of working with Agile methodologies. But only to illustrate that a peer/teaching pair programming approach to address alignment issues is living among active professionals in the field in the introduction of this research.

Theme 1	And	**Theme 2**	And	**Theme 3**	And	**Theme 4**
eXtreme programming	Or	Agile Development	Or	Efficacious alignment	Or	Literature review
Pair programming		IS Development		Business IT alignment		Current state
Collaborative learning		Large scale agile		Complex Adaptive Systems		
		Agile transformation		Co-evolutionary information system alignment		

Table 1. Online database search strategy.

Further literature was then sought by using the back and forward snowballing technique (Webster & Watson, 2002) on all 3 strands of research concerning either pair programming, large scale agile or COISA. With this technique, a total of 37 academic articles was selected for the theoretical foundation of this research, of which 14 have been selected for the first theme about pair programming, 10 for the second theme regarding large scale agile and finally 13 for the third theme regarding COISA. A list of papers by theme is available in appendix A.

2.2. Co-evolutionary Information System Alignment

COISA is a recent approach, in a pursuit to understand business and IT alignment in complex environments (Amarilli et al., 2016; Zhang et al., 2019). It is based on complexity-based conceptualizations of alignment, which are suited for organizations facing highly unstable environments and complex internal structures (Allen & Varga, 2006; Amarilli et al., 2016, 2017; Benbya & McKelvey, 2006; Walraven et al., 2018, 2019, 2020). In this strand of research, alignment is defined as a continues process with different two-way interactions in and between strategic and operational context including business, IT and third parties. This model identifies five holistic alignment processes in two organizational contexts: strategy formulation and implementations process within a strategic context on one hand, and IT usage and implementation in the operational context on the other, with Enterprise Architecture Management as a bridge connecting both (Walraven et al., 2020).

As this paper will investigate efficacious alignment in a large-scale agile environment, by using the pair programming software development method, this research will focus on the IT implementation alignment process in the operational context of the COISA model. IT implementation encapsulates all undertakings that are part of embedding IT within an organization, which include quality design, implementation of requirements and also dimensions such prioritization and change management (Walraven et al., 2018).

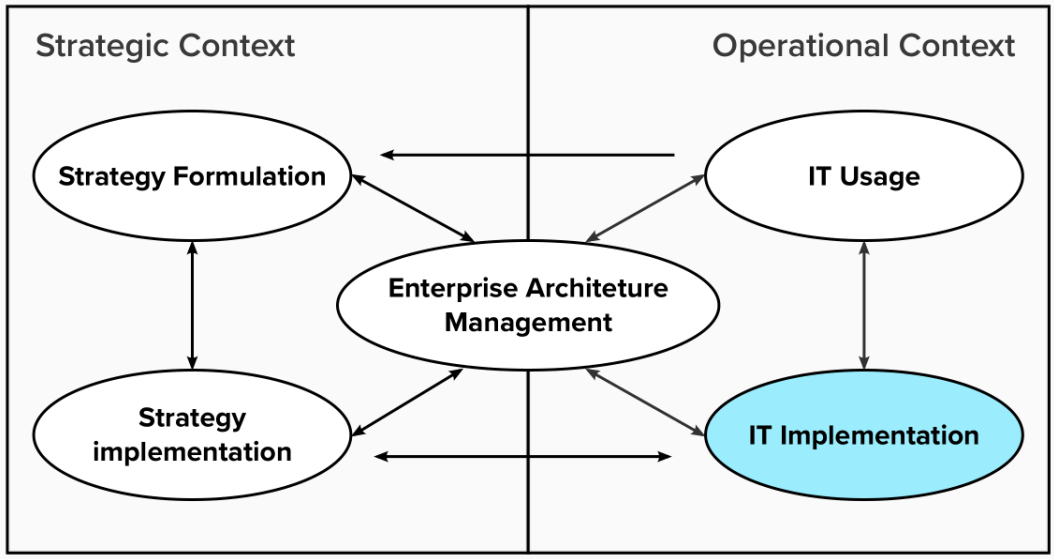

Figure 2. COISA processes characterized by co-evolutionary IS stakeholder interactions (Walraven et al., 2020).

2.3. Efficacious COISA interactions

To ensure that actual alignment occurs as a continued process in complex environments and not just the manifestation of COISA interactions, recent work from Walraven et al. (2020) suggests specific facilitators for efficacious COISA interactions from case study research and focus groups. These facilitators are divided into four overarching categories, namely: (1) alignment motivation, (2) stakeholder involvement, (3) interconnections and (4) alignment decisions. Each category should answer a specific question to ensure efficacious interactions within the COISA model respectively being: why, who, how and what?

2.3.1. Alignment motivation

The alignment motivation category includes seven facilitators relevant to the alignment in IT implementations, being: (1) accountability & mandate, (2) planning & monitoring, (3) intrinsic motivation of actors, (4) perceived (PP) benefits, (5) misalignments and lastly (6) legal obligations. All these facilitators are intended to motivate IS stakeholders in participating in co-evolutionary interactions within a specific process, essentially to answer why those interactions need to take place. Although end-user training as a motivation facilitator was not mentioned with the IT implementation of Electronic Medical Records (EMR) as investigated by Walraven et al. (2020), it connects well with pair programming, while being one of the most successful evidence based collaborative learning methods for programming-related courses (Hannay et al., 2009; Yang et al., 2016). Not to mention the lack of training and coaching being one of the many challenges in large scale environments (Dikert et al., 2016).

2.3.2. Stakeholder involvement

Stakeholder involvement relates to answering who should be involved, by selecting the appropriate actors that can ensure efficacious alignment. This category defines seven facilitators applicable to alignment in IT implementations, specifically: (1) representation of different perspectives, (2) internal & external actors, (3) champions or motivators, (4) translators or unity in language, (5) knowledge of PP, (6) unofficial leaders and (7) openness to perspectives. Although this study specifically focusses on alignment within the IT implementation, which could limit the importance of different perspectives (from other contexts), selection of the right actor combinations is crucial for making efficient pair programming pairs (Chen & Rea, 2018) across different agile teams.

2.3.3. Interconnections

Interconnections answers by which means or how stakeholders can interact with each other to ensure efficacious alignment and knowing where the knowledge is located in the organization. These facilitators involve: (1) formal governance, (2) transparency, (3) existing informal networks and (4) supporting tools. As a large-scale agile environment involves multiple teams who should work within the same larger company goals (Dikert et al., 2016), being able to connect the relevant IT implementation stakeholders across those different agile teams within the organization is already a crucial step before any pair programming can start.

2.3.4. Alignment decisions

The alignment decision facilitators relevant to IT implementation alignment include (1) common guidelines to ensure co-evolution between interactions in operational alignment processes, (2) a central coordination of a group of stakeholder representatives with clear vision and without a personal agenda, (3) emergent decision making from interactions that can be seen as logical improvements and lastly, (4) having a good supporting infrastructure. These are specific decisions that are taken in the alignment process itself that can subsequently be used in future COISA interactions within those same processes.

2.4. Large Scale Agile development

Agile development is a collection of iterative and incremental software engineering methods captured by the 'Agile Manifesto' (Fowler et al., 2001) serving as an alternative for the more traditional 'waterfall' methods involving up-front planning and strict change management (Dikert et al., 2016). Although agile methods have shown that they can improve satisfaction for both customer and developer, evidence also shows that these methods are ill fitted for undertakings on a larger scale (Dikert et al., 2016; Dybå & Dingsoyr, 2009) as larger organizations have more dependencies between projects and teams. Agile teams must interact with other organizational units which are often non-agile in their nature making inter-team coordination difficult. Additionally, the shift the short-term planning and goals raises concern as businesses and customers often build on long term roadmaps (Boehm & Turner, 2005). The foundation of Agile is not necessarily a set of tools or practices but a change of mindset, requiring stakeholder education (Boehm & Turner, 2005).

We consider large scale agile as software development organizations with 50 or more people or at least six teams which includes companies that as a whole focus on software development, as well as development units' part

of larger (non-software focused) organizations that create inhouse software applications (Dikert et al., 2016). One of the popular agile methods is XP (Hamed & Abushama, 2013) which is a collection of practices for enabling efficient incremental development (Beck, 1999). Including the technique of pair programming, which we will use as an interaction point between stakeholders within the IT implementations COIA process in large scale agile environments. Many development implementations combine XP with the Scrum method (Fitzgerald et al., 2006) focusing on the project management viewpoint of agile (Schwaber & Beedle, 2001).

2.5. Pair programming

Pair programming is a collaborative learning strategy used for either computer science education or extreme programming (Goel & Kathuria, 2010; Yang et al., 2016) where pairs work together to achieve a common goal (Preston, 2006). It's two developers that work together on one task, with one as the driver controlling the keyboard, and the other as the navigator which can provide alternatives to the programming task at hand (Arisholm et al., 2007; Dybå et al., 2007; Sun et al., 2015). If applied correctly, pair programming should always provide better quality code, as it provides a mechanism for real-time problem solving and quality control where the navigator helps to plan, identify and prevent any syntactic or strategic deficiencies in code or design document, thinks of alternatives, and asks questions to what the driver is doing (Dou & He, 2010). Pair programming was first introduced as one of the ten XP principles (Sun et al., 2016). As it is a development technique of an agile software development methodology, it also fits within the context of large scale agile.

Earlier work studying the effectiveness of pair programming against solo programming of software professionals showed mixed results on various factors like effort, knowledge transfer, cost, defect rate or job satisfaction (Sun et al., 2015). Research results by Sun (2015) indicate that pair programming is a viable alternative to solo development, but more so under particular conditions. Pairs seem to perform better in larger projects when having higher expertise or a stronger background in pair programming itself. This implies that the project scope in terms of size and complexity can also be a driver for the usage of pair programming. But at the same time is also dependent on the programmers at hand and how pairs are composed between junior and more experienced senior colleagues (Sun et al., 2015).

In essence, PP should be chosen over solo development when the complexity is high (Sun et al., 2016). As the complexity of a system increases, so do the alternatives, unknown connections, interrelated task and diversity of information needed to solve the problem. Pairs of experienced developers will work the fastest on complex problems, but at a price. Meaning such pairs would be ideal to solve critical and complex tasks. But when the goal is to provide cross-training opportunities, making pairs from junior and senior developers is recommended. However, in all compositions, knowledge transfer occurs when working in pairs and increases with higher complexity or when there is a larger gap between the pair's seniority or earlier experience in PP (Sun et al., 2016). As the pair composition is of crucial importance, both junior and senior participants need to be investigated.

3. Research approach

3.1. Methodology

The goal of this research needs to interpret human social interactions in a continual process, a subjectivist and interpretivist perspective is handled to study the phenomenon's details and understand what is happening, and how it is experienced. To be able to gather data on alignment facilitators, an exploratory study will be conducted as it is well suited to discover what is happening and gain insights about our subject of interest in its empirical context (Saunders et al., 2019). The usage of pair programming in large scale agile environments as a method for alignment needs to be explored in its details, a specific qualitative research strategy is chosen to fit to those needs. A case study is an in-depth inquiry into a topic or phenomenon within its real-life setting (Saunders et al., 2019; Yin, 2018). Due to its ability to create insights from intensive research into the analysis of a phenomenon in its empirical context that lead to rich empirical descriptions and theory development (Saunders et al., 2019), it fits the needs of this research perfectly.

To assess whether pair programming can contribute to alignment in a large scale agile environment, an inductive approach is taken to generate meaning from collected data. As mentioned in the literature review, no or very little evidence was found suggesting the usage of pair programming for alignment, nor the usage of the COISA model to study alignment interactions in large scale agile environments. Because an interpretivist philosophy is followed, semi-structured interviews will be conducted, which give the ability for the interviewee to explain certain terminology or meanings to the phenomena that he or she is describing. The structured interview themes however, are prepared from theory. Meaning that data collection starts deductively to ensure consistency across the interviews, so that findings can be reflected against the used theory and enable answering the research question (Saunders et al., 2019). Interviews are recorded, transcribed and subsequently coded so that it can be compared with (coded) documentation and to provide a way to analyse the qualitative data.

3.2. Case Description

The selected case for this research is a European marketing department within a large global pharmaceutical company selling products in varying therapeutical areas and with production facilities across multiple counties. They have recently introduced new ways of working, adopting agile practices at a large scale in their operational teams, across all their regional markets or countries. Another aspect of this enterprises' ambition is its aspiration in becoming a data driven organization, developing and implementing various informative dashboards to measure marketing activities across different platforms such as e-mail, web portals, social media or (virtual) events and across their different markets. All markets within the European region uses the same marketing or analytical applications from a global IT back bone, except for a few very customized country-specific solutions. Because of the culture specific nuances, budgets and different local regulations between European countries, the department's agile teams operate in a de-centralized manner within each market, which also leads to silo and custom development across agile teams for their data & analytical products, as products can also be used in different nuances across different teams. Because of the shared usage of tools and common overlap in dashboard needs, development stakeholders across different teams started to organize pair programming sessions in order to align on their ways of working, goals, targets and differences to develop and scale applications faster across different markets. As the organization adopts agile development practices at a large scale and pair programming sessions are organized across different teams in that environment perusing the same aligned company goals, the case organization fits perfectly to the needs of this study.

3.3. Data Collection

Data is collected using semi-structured interviews with various stakeholders involved directly with the usage of pair programming. In total, 4 interviews were conducted. Developers with junior or senior skill levels in their development activities, that have participated in pair programming sessions at the case organization will be interviewed. The reason for this focus on developers, is to have sufficient data collection on experiences of different pair compositions that occurred during pair programming sessions. Apart from the interviews, any documentation regarding pair programming sessions will be collected from the case organization. All interviews will be done remotely through MS Teams due to strict COVID-19 restrictions within the pharmaceutical sector and each local role being located within a different country within Europe.

Ref	Role	Experience
A	Local Developer	Senior
B	Local Developer	Junior
C	Regional Developer	Senior
D	Local Developer	Senior

Table 2. Semi-structured interviews with different stakeholders.

3.4. Validity, Reliability & Ethics

To ensure that there is only a limited risk of impacting participants in negative way, the ethical principles as proposed by Saunders et al. (2019) has been handled for this research. Participation is strictly voluntarily and

only conducted with clearly informed consent on the implications of this study. A participant can quit at any given moment and his or her anonymity is always guaranteed in the data collection.

Each interviewee will receive a copy of the transcription for review so that they can confirm the collected results, or provide feedback so that interpretation errors can be avoided. Although the researcher has worked at the case organization and engaged himself in pair programming sessions, at the time of the interviews this was not the case anymore. As such the researcher has no more direct ties to the organization as an employee, client or prospect. Given the background in the organization, it helped interpreting the qualitative results and documentation gathered within this study, especially regarding the complexity of such environments. To be able to triangulate the interview results, documentation can be shared during the interview, but for compliancy reasons no proprietary documentation will be gathered and stored outside of the case organization. As the focus of this research is pair programming, multiple developers will be interviewed that engage in pair programming sessions for more representable data and construct validity. Additionally different experience levels within the interviewed roles, as pair composition is crucial for pair programming performance and as such the internal validity of this study.

4. Results

4.1. Overview

From the collected data of several developers participating in PP sessions in a large scale agile environment across different markets or countries in the European region. Facilitators from all categories for efficacious alignment in COISA processes could be found. Especially facilitators related to motivation in particular, followed by alignment decisions had at least 2 which were mentioned by all participants. The result for each category will be elaborated further within this chapter to answer our research question: *How do efficacious alignment facilitators contribute to pair programming in the IT implementations' alignment process of a large scale agile environment?*

Motivation	A	B	C	D
(Prevention/Solving of) Misalignments	X	X	X	X
Perceived benefits (of PP)	X	X	X	X
Scale knowledge & IT implementations	X	X	X	X
Training & Learning		X		X
Planning & Monitoring				X
Stakeholder involvement				
Different perspectives	X	X	X	X
Unofficial leaders		X	X	X
Champions/Motivators	X		X	X
Leadership				X
Internal & External Stakeholders				X
Interconnections				
Supporting tools	X	X	X	X
Existing/creating informal networks	X		X	X
Transparency	X		X	X
Formal Governance			X	X
Alignment decisions				
Central coordination	X	X	X	X
Common guidelines	X	X	X	X

Supporting infrastructure	X			X
Emergent decision-making			X	

Table 3. Overview of all found facilitators during data collection.

4.2. Motivation – Why?

In total, 5 facilitators were found within the motivation category, being (1) perceived benefits, (2) the prevention or solving of misalignments, (3) scaling of knowledge & IT implementations, (4) Training & Learning and (5) planning & monitoring. The remaining facilitators from the theory will also be discussed where possible. From this list we note that the scaling of knowledge & IT implementations is a new facilitator in comparison to the initial theoretical framework used for this research, but was highlighted by each participant as a motivation in engaging in pair programming sessions. To illustrate this, a quote from participant A:

> *In general this pair programming that we did is an intuitive attempt to close some gaps in skills, in scaling applications (from or to other markets), which was very necessary at the time.*

Firstly, the motivations from developers for engaging in PP sessions within the case organization are deeply rooted in the prevention or solving of misalignments within IT implementations, overall perceived benefits received from such (PP) sessions and the scaling of knowledge and applications across different teams (operating in separate markets or countries). Because of the (bi-directional) knowledge transfer that occurs between participants, both parties can uncover different, new or better ways to code or create a solution during pair programming and at the same time tackle local discrepancies in applications. Effectively scaling applications or subject matter expert knowledge to other markets instead of starting from scratch, which should increase time-to-market. For instance to spread the usage of a new data warehouse and migrate existing applications as mentioned by participant D (local) and confirmed through participant C (regional):

> *If it's something big or new that all the countries need to use and need to be introduced (a specific module, way of working, framework,…). Then it's going from the regional core to every market. We have done it in the past where we say: instead of a big workshop that isn't efficient because everyone learns at different speeds and also there might be customization needs due to differences between markets (language for instance): let's organize multiple pairing sessions where a few subject matter experts align with each market individually and do it that way.*

By sharing with each other, developers are able to write code better & faster, either from inspiration or re-use and/or discover flaws in the their current work on common or similar subjects or implementations and facilitate more code consistency across the different agile teams across the region. One participant even going as far to say it's a 'necessity' to efficiently realize an IT implementation. Although not perceived as perfect, it should make code more maintainable as it gets more recognizable by others because of that continuous sharing and re-using. One of the participants formulated this behaviour nicely:

> *Alignment is a given in every pair programming session. You are both looking at code, deliverables, usage of the right folder structure, naming conventions, right processes, are we using best practices? All that is part of the alignment amongst you and the other person… If a person creates code like that, it's also part of governance and synchronizing with each other: "Hold on, it seems you aren't properly aware of that common practice". It might not initially be a problem in the beginning due to developers being in different markets, but might give problems down the road if a best practice is not applied.*

Secondly, the motivation for training and learning was present with 2 participants, one being a less (technical) experienced local developer, the other an experienced senior developer. When you pair junior and senior pairs, there is a bigger knowledge transfer: the senior shows his way of working and by doing so with pair programming, aligns in that way with the less experienced developer, that starts to write his or her code in the same way. This could be confirmed by participant B, who was still applying practices he learned from another

developer such as the idea's, tools and ways of working, while not having an experienced coding background which was required to create or change existing reporting applications. Combining all the beforementioned regarding motivation, it is interestingly also the senior developer who introduced the use of pair programming, that sees a positive impact on projects. Teams can adjust their applications to be more scalable to other markets, by sharing good practices & vision on data and in a larger whole, increase time to market. Another could imagine that pair programming might have a beneficial impact on accountability or mandate, as code is more easily meant to be shared and watched by others, but this could however not be properly confirmed from other interviews. The fact that pair programming was also meant as a regional experiment, but not mandated from the companies (top-down) agile transition, could explain why.

Lasty, no intrinsic motivation was coded from any of the participants, but might be masked behind the experienced benefits gained from pair programming:

> *This motivates you because you learn something new in a fast way and you can continue in an 'unleashed' way. Your colleagues are also happy you can re-use their existing work. If you can deliver faster, your manager is also happy.*

4.3. Stakeholder involvement – Who?

In total, 4 facilitators were found within the stakeholder involvement category, being the (1) representation of different perspectives, (2) internal & external actors , (3) champions or motivators, (4) unofficial leaders and (5) openness to perspectives. The remaining facilitators mentioned in the theoretical framework: translators or unity in language and knowledge of PP will also be discussed in this section.

Champions or motivators are there that take effort in organizing sessions and form pairs across different teams. This is a trigger person or organizer, that can match pairs together by having good knowledge of what goes on in other markets. What is interesting to mention is that pair programming is not part of the large scale agile ways of working that where implemented top-down in the case organization. A pair programming experiment, as it was called by one of the participants, was started by a community (considered as the internal stakeholders for pair programming) lead within the region and meant as an intuitive mechanism to ensure some kind of code consistency across different teams working on similar projects and priorities and scale each other's work. Although at the time of the interviews not a lot of sessions where occurring anymore as different teams are working on separate tasks and diverging, pair programming might come back into place once pilots start finishing and can potentially be shared with others. The exception are sessions occurring between more centralized (regionally operating) teams and local (country or market allocated) teams, which were still occurring more regularly. In that perspective, from aligning to diverging and re-aligning in a cycle, pair programming could be seen as an essential part in the case organization's continuous IT implementation alignment process.

Each developer taking part could be seen as an unofficial leader for a specific topic, in which this person is experienced. For a specific subject area someone can be the contact point or 'lead' for that specific knowledge and share his experience with other markets in pair programming sessions, even for participants without a lot of technical experience. And by learning from each other through pair programming, participants can also share each other's perspectives on how to develop a solution as there can be many different ways in creating an application or solving a problem in already existing application or code. And it's when you pair seniors with each other, which for the purpose of this review can be seen as 'experienced developers' (A,C,D), where different perspectives would truly alignment across different teams, which can be directly quoted by one of the experienced participants (D):

> *But the true alignment in the company happens more when you pair a senior with a senior when they see different methodologies, it creates discussions… "Why do you do it like that or like that?". When you have such a pair they have to accept one convention and it's where alignment begins. Because at the end, if each time we begin to agree on some convention, there can be global conventions that can start appearing. When seniors align with each other, we see different ways of working and when you learn and discuss what's the best way of working like that as well.*

For openness of perspectives, we could assume that this would be 'a given' or even requirement for pair programming with the above statement. As different perspectives are shared during pair programming from which decisions or proposals can be made and picked up with a larger group a larger group (see 'Alignment Decisions'). We could argue the same for translators or unity in language, as all participants spoke different native languages and all interviews were conducted in English and all participants share a background in IT. Without some kind of unity in language, such long sessions (about 4 hours) regarding in-depth technical topics would seem impossible to do. As such an argument could made that some kind of unity in language is a required for pair programming to effectively occur, not just alignment, as pair programming only operates within the IT implementation for the investigated case. Knowledge of pair programming as a facilitator could also not be directly extracted from the collected data, but it was mentioned that a scrum master helped setting up the basic methodology and centrally documented standards are available and are expected to adhere to. As stated by one of the participants, the key is more in mixing the right people or unofficial leaders together to maximize the knowledge transfer or address the problem or case the pair programming session is organized for:

> *That's the power of pair programming, it's simple, it's only a minimal need, two people. However, mixing those different is important.*

Lastly, for leadership it can be stated that this facilitator was only found from data collected from one participant. This should not really be a surprise that this was also the main champion or motivator that initially proposed the usage of pair programming across the different teams to address the organization's challenges with IT implementations. Because pair programming is organized across different teams that are located in different markets and countries, at least some level of higher level leadership buy-in is needed. Developers spend time with each other, on a subject that isn't always necessarily related to a local priority of one of the participants. As these sessions are interactions between developers, there is also no direct leadership involvement in those interactions, only when discussions from pair programming require leadership attention. This was made more clear by results concerning external stakeholders, in the form of local contractors that are not included in the repeatedly mentioned community within the organization. Which is understandable as those are specific local investments that aren't allowed to continuously engage in other priorities, but at the cost of efficiency as those case are more isolated from the rest, making their development less maintainable in a larger whole as stated by the same participant. There are exceptions though, as regional consultants are still involved and the involvement of a scrum master for the initial pair programming set up. It is for this reason the internal & external stakeholder representation facilitator was included within the results of this research.

4.4. Interconnections – How?

In total, 4 facilitators were found within the Interconnections category, being (1) formal governance, (2) transparency, (3) existing informal networks and (4) supporting tools. All the facilitators from the theoretical model could be found from the participants.

Rules & best practices, based on industry standards are documented, but not formally enforced top down from an Agile perspective. These are expected to adhere to, like organizing around a specific topic or need, the switching of driver & navigator, not stopping halfway for other meetings, respect time limits and bringing a 'good mood' or a 4 hours limit within a day. Together with regular community meetings and a central information portal, supported by IT leadership, it forms the formal governance in the case organization around pair programming. It should be stated that pair programming is not implemented as mandated by XP methodology, but accustomed to the needs of the organization, as developers are also based in different locations or are more regularly working from home due to COVID. Meaning the pair programming sessions (in the case organization) are all remote. Screen sharing and switching (executor/watcher) roles make supporting tooling a necessary requirement to even start engaging in such sessions, not to mentioned the different tools that are required to actual create or fix applications and their code.

It's also not surprising that one of the participants did mention a certain 'lack of interpersonal contact' in an international pandemic setting, as in larger meetings or outside of the community it's mostly professional conversations. Within the community you can create a more casual & feel good atmosphere without having

management present. And in pair programming sessions, as you spend a lot more time with each other to deep-dive into issues or code, there is also time for small talk. Not just on usage of tools, plugins or tips, tricks & other general productivity topics, but also just small talk around mood or culture. Together with the follow up that can come after pair programming, should tasks not get finished, this can form new informal connections within its community.

> *"What is interesting to see is that even though we are working more in silo's, I know perfectly what is going on in different teams within our region."*

Lastly, with pair programming, there is an in-depth dive into a specific problem or code which is not possible in large or short meetings. Teams are transparent with each other down to a code level, even without being in the same market or country and as such, make the numerous differences (and resulting needs) between countries more clear and increasing visibility of what is going on in other markets, however limited by the size or scope of the community of internal stakeholders in which knowledge can be shared.

> *Our common calls can be with 15 people or more. That is too many to deep dive into low level technical things. So pair programming 1:1 is where you can really spent time discussing just the output of one single function in the code, for more than 2 hours if you want.*

4.5. Alignment Decisions – What?

In total, 4 facilitators were found within the Interconnections category, being (1) common guidelines to ensure co-evolution between interactions in operational alignment processes, (2) a central coordination of a group of stakeholder representatives with clear vision and without a personal agenda, (3) emergent decision making from interactions that can be seen as logical improvements and lastly, (4) having a good supporting infrastructure. All the facilitators from the theoretical model could be found from the participants.

As already touched upon during the analysis of stakeholder involvement, it is when you pair experienced or senior developers with each other, that true alignment can occur once they start comparing their ways of working or coding. As such they can make emergent decisions on how to write their code during a session, when the pair tries to solve a problem, but it depends where the issue resides. Should it be a more common problem, pair programming sessions can produce proposals for new coding principles or broader solutions that can be discussed together within a central group of (internal) stakeholders, or as usually referenced in this research: the community engaging in pair programming. And over time, code can be improved upon by finding new or more efficient ways of building or even using applications, meaning existing guidelines will have to be updated or new ones created within that community. It should be important to mention that no management is involved in these community meetings, meaning the group focusses on technicalities & efficiency within their IT implementations. In the same way, discussions or issues on supporting architecture can help escalating the issue and eventually voicing it to the right stakeholders and channels.

> *Mostly improvements on infrastructure are discussed during our general community calls with more stakeholders present. That format is used more to discuss our grievances and infrastructure problems. At least there you have more effect, because more people that experience the same problems are listening in. Then they can also vote for the importance of such (infrastructure) problems and can also vote to have it fixed.*

To summarize the above we should state that emergent decisions during pair programming sessions are possible, but depend on the scope in which the decision need to be made. It's there where the central coordination of a group of stakeholders becomes vital. Pair programming can be a trigger where pairs come up with proposals of new guiding principles, conventions or problems regarding infrastructure. So there needs to be a more centralized group where these topics can be channelled, either to implement within the community or to escalate a community need further to the right stakeholders that are able to tackle or decide on that. And as already mentioned earlier in the motivation section, common guidelines are easily shared during such sessions to positively influence code consistency and expectations or even IT usage shown from participant B.

5. Discussion

From the results it can be suggested that, because facilitators from all categories as described by Walraven et al. (2020) could be found, there is a strong indication of efficacious alignment facilitators contributing to pair programming in several ways. But it's limitations should not be underestimated. Pair programming remains an interaction between technically involved stakeholders, meaning it is mainly or even only, executed within the operational context of IT implementations. This however does not mean it cannot influence or trigger alignment in other COISA processes that govern IT usage or architecture as we have seen in the results. To be able to make alignment more understandable within a complex and unstable environment such as large scale agile, the CAS-based theory of COISA is used as the theoretical lens within this research. Because it's a relatively new approach, there has been no research so far combining the theory of agile methods with COISA. Subsequently, to be able to answer *how efficacious alignment facilitators contribute to pair programming in the IT implementations' alignment process of a large scale agile environment*, this research uncovers which specific facilitators for efficacious alignment occur while using pair programming in a large scale agile environment from participants involved in such sessions.

5.1. Required facilitators

Some facilitators are arguably a requirement before pair programming in itself can occur. For instance, emergent decision making is possible up until a certain level, but more common guidelines or supporting infrastructure cannot be implemented by pair programming itself. But it can trigger a discussion when such a structure is in place, where discussions can be channelled through a centralized group or community from which these sessions are organized, so that it can be picked further with the right stakeholders. Then, either guidelines can be spread and communicated through that same community, or be escalated further to an EAM alignment process to improve supporting architecture, or the creation of guidelines for IT usage.

And because teams can be allocated in completely different markets or countries, at least some form of leadership buy-in as a stakeholder involvement is needed before such sessions can occur, as in most cases pairs have different priorities across different markets or countries. Together with the previously mentioned community, some form of (formal) governance should also be expected where knowledge pockets, pairs and rules to adhere to (when engaging in pair programming sessions) are centrally available to its community. Additionally, the right supporting tools are needed to be able to engage or interconnect in pair programming remotely, being software to enable screen sharing to enable the role of watcher and executioner.

5.2. Strong motivational foundation

Pair programming can have a certain cost associated to it, as you are using 2 resources for the same task that might not be a priority for one of the resources pairing. Although in the case organization, pair programming only occurs on need basis, there should be a strong motivational need that should be addressed within the context of their large scale agile environment, next to having leadership buy-in. Also because the effectiveness of pair programming versus solo programming showed mixed results on various factors like effort, knowledge transfer, cost, defect rate or job satisfaction from software professionals (Sun et al., 2015). So there should be a perceived benefit. From the results we also see that perceived benefits are influenced by other facilitators like different perspectives and transparency which are all facilitators for the knowledge transfers that occurs in pair programming sessions.

This experiment was proposed and started by one of its community seniors to address the different directions markets where going into in terms of technical implementations. As a result the pair composition is determined for **a)** a case specific need or topic, where someone that is very knowledgeable in on that topic can be leveraged for a pair programming session to help with the other person's case, **b)** to more quickly scale applications across different markets and immediately solve local discrepancies or **c)** for general learning of less experienced or new developers. Which interestingly can directly be connected with the motivational facilitators (1) perceived benefits, (2) the prevention or solving of misalignments, (3) scaling of knowledge & IT implementations and (4) training & learning. At the centre of pair programming we can see the impact of in-depth & transparent knowledge sharing in all kinds of different scenarios: writing better & more efficient code, address or uncover

misalignment, solve knowledge gaps and training needs. Resulting in better code consistency so that in the end everything becomes more manageable and implementations can be delivered faster by scaling from one market to another and breaking through (differently aligned) silo's in the organization. In the end there is better alignment between developers. With some of the challenges that organizations face while adopting agile on a larger scale are for instance lack of the difficult interfacing between teams, the achievement of technical consistency or specialized knowledge kept in internal silo's (Dikert et al., 2016), we can argue that pair programming can help address those challenges. But not for training & learning, as that specific challenge in large scale agile is just related to agile ways of working and does not describe IT implementations' specific training & learning in the context of pair programming.

5.3. Stakeholder involvement & pair composition

Results by Sun (2015) indicate that pair programming is a viable alternative to solo development, but more so under particular conditions. The pair composition is of vital importance depending on the goal that needs to be achieved. We should however mentioned that Sun's (2016) research is mainly based on the concept of only use on or the other for development tasks, measured on time and effort. Within this research, a mixed method is investigated where both seem to complement each other due to solo programming actually benefitting from pair programming by sharing knowledge from the right expert. Meaning that comparing performance would be very difficult or even impossible with the environment that has been studied in this research, but rather focus on the composition of the involved stakeholders in pair formation that can influence that performance.

Although knowledge transfer occurs for all kinds of compositions, 'junior' and 'expert' pairs have a larger gap in knowledge and are an ideal fit for cross-training opportunities with pair programming (Sun et al., 2016). This scenario was also uncovered by this research from both a less technical experienced and experienced developer. But from the results we also see that anyone, even less experienced technical profiles can be an 'expert' ('Subject Matter Expert') or unofficial leader in a specific subject others can connect with. This should further confirm, that indeed knowledge transfer always occurs, regardless of which stakeholder is involved, but is still influenced by the composition of the pair. But given the larger gap in technical skills, the transfer of knowledge should still always be bigger. Results also suggest that champions or motivators should be present, that can help trigger pair programming sessions across the community and help set up the right pairs. As mentioned in section 5.1, central coordination of a group of stakeholder or representatives with clear vision and without a personal agenda, is needed to keep an overview over organized sessions within its community. This could potentially answer Dikert's (2016) call for interactions and alignment with more stakeholders, not just between business & IT but within IT development as well. Because agile teams can't work as independently as they are originally designed, as they need to work as a larger whole (Dikert et al., 2016; Uludag et al., 2017, 2019).

5.4. Inform, legitimate and socialize

Winter (2016) suggests to use interactions with agile teams that inform ("to communicate knowledge"), legitimate ("able to be defended with logic or justification") and socialize ("mix socially with others") because they are hard to control or enforce. Here we see a great fit with the uncovered alignment facilitators benefitting pair programming in a large scale agile environment. Based on the results described in the previous chapter we argument the following.

Firstly, the continuous (in-depth) knowledge sharing of common practices or the scaling of knowledge & applications between different teams, especially between regional and local levels within the organization, effectively inform the participants. Participants will over time be better informed of best practices and knowledge that exists within the organization, although for real overview we argue that this is mainly due to the orchestration and composing of pairs when all (programming) stakeholders sit together. Secondly, the benefits gained from pair programming sessions effectively defend its use. Results show clear advantages to be gained from such sessions, even going as far that some tasks are almost impossible to complete without pair programming sessions. Thirdly, because pairs engage in lengthy sessions where they can deep dive into specific (technical) topics, informal discussion also occurs. Not just on professional things like used tools, technical approaches or tips & tricks, but also on personal matters creating informal connections across different teams.

As these people know each other, communication between the two can still occur through chat or e-mail, for instance to follow up on something that occurred during their session.

5.5. Implications, limitations & future research

Overall we can conclude that alignment facilitators positively contribute to efficacious pair programming across all investigated categories, and has the potential to address challenges faced within large scale agile environments based on our analysis, although not without its limitations: results did not show any shared perspectives on agile ways of working but actual technical work. And the overall coordination of agile teams is still perceived as being very difficult, potentially misaligned, or has a lot of room for further improvement. Within a team however, following an incremental agile development methodology is positively received. This means that while we can see a role for the pair programming development technique in large scale agile environments, it will certainly not be able to address all challenges faced in such environments as found by Dikert et al (2016). And given the amount of challenges, pair programming only has the potential to address some of them, which could be expected as this research also only looks at the IT implementation's context of COISA. Additionally only 1 case has been investigated in the pharmaceutical sector, limiting external validity. Although it should be noted that pair programming or agile practices, and thus its challenges, are present in many organizations (Dikert et al., 2016; Dybå & Dingsoyr, 2009). We should also note that the developers are mainly or almost only occupied with analytic products, which can not be easily compared to web or mobile applications.

There are several area's where future research can be seen: To begin, only alignment facilitators have been investigated in the IT implementations' context of COISA, by interviewing the same technical role of a developer. Other roles, within the IT implementation or other processes from Walraven's (2019, 2020) COISA process model should be investigated to get a broader picture, and build further upon the research question in this study. Secondly, there is so far no academic research that investigates how pair programming can contribute to large scale agile challenges as described by Dikert et al (2016) but however shows potential. Finally, pair programming as a mixed method should be investigated further as the results in this study show promise in using it as a hybrid between pair and solo programming.

References

Allen, P. M., & Varga, L. (2006). A co-evolutionary complex systems perspective on information systems. *Journal of Information Technology*, *21*(4), 229–238. https://doi.org/10.1057/palgrave.jit.2000075

Amarilli, F., Vliet, M., & Hooff, B. (2016, January). *Business IT Alignment through the Lens of Complexity Science*.

Amarilli, F., Vliet, M., & Hooff, B. (2017, January). *An Explanatory Study on the Co-evolutionary Mechanisms of Business IT Alignment*.

Arisholm, E., Gallis, H., Dybå, T., & Sjøberg, D. (2007). Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. *Software Engineering, IEEE Transactions On*, *33*, 65–86. https://doi.org/10.1109/TSE.2007.17

Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70–77. https://doi.org/10.1109/2.796139

Benbya, H., & McKelvey, B. (2006). Using coevolutionary and complexity theories to improve IS alignment: a multi-level approach. *Journal of Information Technology*, *21*(4), 284–298. https://doi.org/10.1057/palgrave.jit.2000080

Boehm, B., & Turner, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *Software, IEEE*, *22*, 30–39. https://doi.org/10.1109/MS.2005.129

Canfora, G., Cimitile, A., Garcia, F., Piattini, M., & Visaggio, C. A. (2007). Abstract Evaluating performances of pair designing in industry. *Journal of Systems and Software*, *80*, 1317–1327. https://doi.org/10.1016/j.jss.2006.11.004

Chen, K., & Rea, A. (2018). Do Pair Programming Approaches Transcend Coding? Measuring Agile Attitudes in Diverse Information Systems Courses. *Journal of Information Systems Education*, *29*(2), 53–64. https://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=130161423&site=ehost-live

Coltman, T., Tallon, P., Sharma, R., & Queiroz, M. (2015). Strategic IT Alignment: Twenty-Five Years on. *Journal of Information Technology*, *30*. https://doi.org/10.1057/jit.2014.35

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, *119*, 87–108. https://doi.org/https://doi.org/10.1016/j.jss.2016.06.013

Dou, W., & He, W. (2010). A Preliminary Design of Distributed Pair Programming System. *Education Technology and Computer Science, International Workshop On*, *1*, 256–259. https://doi.org/10.1109/ETCS.2010.371

Dybå, T., Arisholm, E., Sjøberg, D., Hannay, J., & Shull, F. (2007). Are Two Heads Better than One? On the Effectiveness of Pair Programming. *Software, IEEE*, *24*, 12–15. https://doi.org/10.1109/MS.2007.158

Dybå, T., & Dingsoyr, T. (2009). What Do We Know about Agile Software Development?. *IEEE Software*, *26*(5), 6–9. https://login.ezproxy.elib11.ub.unimaas.nl/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=afh&AN=43973897&site=ehost-live&scope=site

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems*, *15*(2), 200–213.

Fowler, M., Highsmith, J., & others. (2001). The agile manifesto. *Software Development*, *9*(8), 28–35.

Goel, S., & Kathuria, V. (2010). A Novel Approach for Collaborative Pair Programming. *Journal of Information Technology Education*, *9*, 183–196. https://doi.org/10.28945/1290

Hamed, A., & Abushama, H. (2013). *Popular agile approaches in software development: Review and analysis*. 160–166. https://doi.org/10.1109/ICCEEE.2013.6633925

Hannay, J. E., Dybå, T., Arisholm, E., & Sjøberg, D. I. K. (2009). The effectiveness of pair programming: A meta-analysis. *Inf. Softw. Technol.*, *51*, 1110–1122.

Kappelman, L. A., McLean, E. R., Luftman, J. N., & Johnson, V. L. (2013). Key Issues of IT Organizations and Their Leadership: The 2013 SIM IT Trends Study. *MIS Q. Executive*, *12*.

Kappelman, L., McLean, E., Johnson, V., & Gerhart, N. (2014). The 2014 SIM IT key issues and trends study. *MIS Quarterly Executive*, *13*, 237–263.

Lee, S., Kim, K., Paulson, P., & Park, H. (2008). Developing a Socio-technical framework for business-IT alignment. *Industrial Management and Data Systems*, *108*, 1167–1181. https://doi.org/10.1108/02635570810914874

Marks, E. (2014). *Governing Enterprise Agile Development Without Slowing It Down: Achieving Friction- Free Scaled Agile Governance via Event- Driven Governance.* https://agile-path.com/governing-enterprise-agile-development-without-slowing-it-down/

Ovaska, P., Rossi, M., & Marttiin, P. (2003). Architecture as a Coordination Tool in Multi-site Software Development. *Software Process: Improvement and Practice*, *8*, 233–247. https://doi.org/10.1002/spip.186

Preston, D. (2006). Using collaborative learning research to enhance pair programming pedagogy. *ACM SIGITE Newsletter*, *3*, 16–21. https://doi.org/10.1145/1113378.1113381

Saunders, M., Lewis, P., Thornhill, A., & Bristow, A. (2019). *Research Methods for Business Students*.

Schwaber, K., & Beedle, M. (2001). *Agile Software Development with Scrum* (1st ed.). Prentice Hall PTR.

Sun, W. N., Aguirre-Urreta, M. I., & Marakas, G. M. (2015). Effectiveness of Pair and Solo Programming Methods: a Survey and an Analytical Approach. *AMCIS*.

Sun, W. N., Marakas, G. M., & Aguirre-Urreta, M. I. (2016). The Effectiveness of Pair Programming: Software Professionals' Perceptions. *IEEE Software*, *33*, 72–79.

Thompson, J. D. (1967). Organizations in action: Social science bases of administrative theory. In *Organizations in action: Social science bases of administrative theory.* McGraw-Hill.

Uludag, Ö., Kleehaus, M., Xu, X., & Matthes, F. (2017). *Investigating the Role of Architects in Scaling Agile Frameworks*. https://doi.org/10.1109/EDOC.2017.25

Uludag, Ö., Nägele, S., & Hauder, M. (2019). *Establishing Architecture Guidelines in Large-Scale Agile Development Through Institutional Pressures*.

van Veen, M., & Westerkamp, K. (2017). *Deskresearch*. Pearson.

Vanhanen, J., & Mäntylä, M. (2014). A systematic mapping study of empirical studies on the use of pair programming in industry. *International Journal of Software Engineering and Knowledge Engineering*, *23*. https://doi.org/10.1142/S0218194013500381

Walraven, P., van de Wetering, R., Helms, R., & Caniëls, M. (2020). *Aligning effectively: the case of Electronic Medical Records*.

Walraven, P., van de Wetering, R., Helms, R., Versendaal, J., & Caniëls, M. (2018, January). *Co-evolutionary IS-alignment: a Complex Adaptive Systems Perspective*.

Walraven, P., van de Wetering, R., Versendaal, J., & Caniëls, M. (2019). *Using a co-evolutionary IS-alignment approach to understand EMR implementations*.

Webster, J., & Watson, R. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, *26*. https://doi.org/10.2307/4132319

Wolpers, S. (2019). *Agile Transition – A Hands-on Guide from the Trenches*.

Yang, Y.-F., Lee, C.-I., & Chang, C.-K. (2016). Learning motivation and retention effects of pair programming in data structures courses. *Education for Information*, *32*, 249–267. https://doi.org/10.3233/EFI-160976

Yin, R. K. (2018). *Case Study Research and Applications: Design and Methods* (6th ed.).

Zhang, M., Chen, H., Lyytinen, K., & Li, X. (2019, January). *A Co-evolutionary Perspective on Business and IT Alignment: A Review and Research Agenda*. https://doi.org/10.24251/HICSS.2019.749

Appendix A: References per research theme

Theme 1 "Pair Programming"	1	Arisholm, E., Gallis, H., Dybå, T., & Sjøberg, D. (2007). Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. *Software Engineering, IEEE Transactions On*, *33*, 65–86. https://doi.org/10.1109/TSE.2007.17
	2	Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70–77. https://doi.org/10.1109/2.796139
	3	Canfora, G., Cimitile, A., Garcia, F., Piattini, M., & Visaggio, C. A. (2007). Abstract Evaluating performances of pair designing in industry. *Journal of Systems and Software*, *80*, 1317–1327. https://doi.org/10.1016/j.jss.2006.11.004
	4	Chen, K., & Rea, A. (2018). Do Pair Programming Approaches Transcend Coding? Measuring Agile Attitudes in Diverse Information Systems Courses. *Journal of Information Systems Education*, *29*(2), 53–64. https://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=130161423&site=ehost-live
	5	Dou, W., & He, W. (2010). A Preliminary Design of Distributed Pair Programming System. *Education Technology and Computer Science, International Workshop On*, *1*, 256–259. https://doi.org/10.1109/ETCS.2010.371
	6	Dybå, T., Arisholm, E., Sjøberg, D., Hannay, J., & Shull, F. (2007). Are Two Heads Better than One? On the Effectiveness of Pair Programming. *Software, IEEE*, *24*, 12–15. https://doi.org/10.1109/MS.2007.158
	7	Goel, S., & Kathuria, V. (2010). A Novel Approach for Collaborative Pair Programming. *Journal of Information Technology Education*, *9*, 183–196. https://doi.org/10.28945/1290
	8	Hamed, A., & Abushama, H. (2013). *Popular agile approaches in software development: Review and analysis*. 160–166. https://doi.org/10.1109/ICCEEE.2013.6633925
	9	Hannay, J. E., Dybå, T., Arisholm, E., & Sjøberg, D. I. K. (2009). The effectiveness of pair programming: A meta-analysis. *Inf. Softw. Technol.*, *51*, 1110–1122.
	10	Preston, D. (2006). Using collaborative learning research to enhance pair programming pedagogy. *ACM SIGITE Newsletter*, *3*, 16–21. https://doi.org/10.1145/1113378.1113381
	11	Sun, W. N., Aguirre-Urreta, M. I., & Marakas, G. M. (2015). Effectiveness of Pair and Solo Programming Methods: a Survey and an Analytical Approach. *AMCIS*.
	12	Sun, W. N., Marakas, G. M., & Aguirre-Urreta, M. I. (2016). The Effectiveness of Pair Programming: Software Professionals' Perceptions. *IEEE Software*, *33*, 72–79.
	13	Vanhanen, J., & Mäntylä, M. (2014). A systematic mapping study of empirical studies on the use of pair programming in industry. *International Journal of Software Engineering and Knowledge Engineering*, *23*. https://doi.org/10.1142/S0218194013500381
	14	Yang, Y.-F., Lee, C.-I., & Chang, C.-K. (2016). Learning motivation and retention effects of pair programming in data structures courses. *Education for Information*, *32*, 249–267. https://doi.org/10.3233/EFI-160976
Theme 2 "Large Scale Agile"	1	Boehm, B., & Turner, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *Software, IEEE*, *22*, 30–39. https://doi.org/10.1109/MS.2005.129
	2	Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, *119*, 87–108. https://doi.org/https://doi.org/10.1016/j.jss.2016.06.013
	3	Dybå, T., & Dingsoyr, T. (2009). What Do We Know about Agile Software Development?. *IEEE Software*, *26*(5), 6–9. https://login.ezproxy.elib11.ub.unimaas.nl/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=afh&AN=43973897&site=ehost-live&scope=site
	4	Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems*, *15*(2), 200–213.
	5	Fowler, M., Highsmith, J., & others. (2001). The agile manifesto. *Software Development*, *9*(8), 28–35.
	6	Marks, E. (2014). *Governing Enterprise Agile Development Without Slowing It Down: Achieving Friction- Free Scaled Agile Governance via Event- Driven Governance.* https://agile-path.com/governing-enterprise-agile-development-without-slowing-it-down/
	7	Ovaska, P., Rossi, M., & Marttiin, P. (2003). Architecture as a Coordination Tool in Multi-site Software Development. *Software Process: Improvement and Practice*, *8*, 233–247. https://doi.org/10.1002/spip.186
	8	Schwaber, K., & Beedle, M. (2001). *Agile Software Development with Scrum* (1st ed.). Prentice Hall PTR.
	9	Uludag, Ö., Kleehaus, M., Xu, X., & Matthes, F. (2017). *Investigating the Role of Architects in Scaling Agile Frameworks*. https://doi.org/10.1109/EDOC.2017.25
	10	Uludag, Ö., Nägele, S., & Hauder, M. (2019). *Establishing Architecture Guidelines in Large-Scale Agile Development Through Institutional Pressures*.
Theme 3 "COISA"	1	Allen, P. M., & Varga, L. (2006). A co-evolutionary complex systems perspective on information systems. *Journal of Information Technology*, *21*(4), 229–238. https://doi.org/10.1057/palgrave.jit.2000075
	2	Amarilli, F., Vliet, M., & Hooff, B. (2016, January). *Business IT Alignment through the Lens of Complexity Science*.

	3	Amarilli, F., Vliet, M., & Hooff, B. (2017, January). *An Explanatory Study on the Co-evolutionary Mechanisms of Business IT Alignment*.
	4	Benbya, H., & McKelvey, B. (2006). Using coevolutionary and complexity theories to improve IS alignment: a multi-level approach. *Journal of Information Technology*, *21*(4), 284–298. https://doi.org/10.1057/palgrave.jit.2000080
	5	Coltman, T., Tallon, P., Sharma, R., & Queiroz, M. (2015). Strategic IT Alignment: Twenty-Five Years on. *Journal of Information Technology*, *30*. https://doi.org/10.1057/jit.2014.35
	6	Kappelman, L. A., McLean, E. R., Luftman, J. N., & Johnson, V. L. (2013). Key Issues of IT Organizations and Their Leadership: The 2013 SIM IT Trends Study. *MIS Q. Executive*, *12*.
	7	Kappelman, L., McLean, E., Johnson, V., & Gerhart, N. (2014). The 2014 SIM IT key issues and trends study. *MIS Quarterly Executive*, *13*, 237–263.
	8	Lee, S., Kim, K., Paulson, P., & Park, H. (2008). Developing a Socio-technical framework for business-IT alignment. *Industrial Management and Data Systems*, *108*, 1167–1181. https://doi.org/10.1108/02635570810914874
	9	Thompson, J. D. (1967). Organizations in action: Social science bases of administrative theory. In *Organizations in action: Social science bases of administrative theory*. McGraw-Hill.
	10	Walraven, P., van de Wetering, R., Helms, R., & Caniëls, M. (2020). *Aligning effectively: the case of Electronic Medical Records*.
	11	Walraven, P., van de Wetering, R., Helms, R., Versendaal, J., & Caniëls, M. (2018, January). *Co-evolutionary IS-alignment: a Complex Adaptive Systems Perspective*.
	12	Walraven, P., van de Wetering, R., Versendaal, J., & Caniëls, M. (2019). *Using a co-evolutionary IS-alignment approach to understand EMR implementations*.
	13	Zhang, M., Chen, H., Lyytinen, K., & Li, X. (2019, January). *A Co-evolutionary Perspective on Business and IT Alignment: A Review and Research Agenda*. https://doi.org/10.24251/HICSS.2019.749

Appendix B: Interview template

At all times, probe further with **why**, **how**, **what**, **who** or when questions.

Theme: Large Scale Agile

1. How do you perceive the efficacy of agile development methodologies within the organization?
   Probing questions:
   - Or within the region/department?
   - Are there non-agile teams or departments with dependencies?
2. What kind of problems do you face in terms of alignment or team coordination across the different developers?
   Probing questions:
   - Have you encountered problems regarding for instance technical consistency of code?
   - Are there different interpretations of the 'ways of working'/agile techniques across the development teams?
   - Do you feel you are properly trained (or coached) to work on development tasks within a larger picture?

Theme: Pair Programming

1. How are pairs being composed?
   Probing questions:
   - How experienced are you with pair programming?
2. How does pair programming help solve (some) of those earlier problems?
3. What kind of perceived benefits have you taken from such pair programming sessions?
   Probing questions:
   - How where you able to learn or increase knowledge on certain topics?
   - How was it helpful in reducing complexity?
   - How where you able to execute your work more efficiently? (or not)

Theme: COISA facilitators

1. Motivation (why?) - Why or how has pair programming with developers from other teams motivated you in pursuing better alignment?
   Probing questions:
   - Why did it influence the accountability and mandate of your job?
   - Why did it have an effect on planning & monitoring of your regular tasks?
   - Why has it motivated you?
   - Why or how has it benefited you?
   - Has it uncovered any misalignments across different teams?

2. Stakeholder investment (who?) - Who is involved in the pair programming program?
   Probing questions:
   - With who have you engaged in pair programming sessions? Are different perspectives represented?
   - Are external stakeholders involved in such sessions?
   - Are any 'champions' or 'motivators' (should be senior developers in this case) involved in such sessions?
   - Is there effective communication or can there be a different use in semantics or meaning?
   - Are there any unofficial leaders or experts involved (without a formal mandate)?
   - How experienced are you with pair programming? (2c)
   - Did you ever learn new perspectives or ways of working from other stakeholders?

3. Interconnects (how?) – How do you engage in pair programming sessions or through which channels?
   Probing questions:
   - Describe any formal governance used or needed to organize a pair programming session.
   - How do pair programming sessions lead through transparency across other teams?
   - Are there any informal connections or relations contacted and/or extended in pair programming sessions?
   - Which tools are used during pair programming sessions? Have you ever considered moving to another tool seen in a pair programming session?
4. Alignment decisions (what?) – Have there been any alignment decisions that originated through pair programming sessions?
   Probing questions:
   - Any creation or update of common guidelines for IT implementations?
   - Have there been any topics, subjects or issues that arose and picked up by a central group of stakeholders?
   - Have there been any emergent (technical) decisions that took place during a pair programming session?
   - Have pair programming sessions lead to any improvement on supporting infrastructure in terms of alignment across teams?

Appendix C: Interview codes

Interview A			
Interview Quotes	**Code**	**Alignment facilitator**	**Facilitator Category**
Q: So the tooling options and the code: do you see some consistency across the different teams? So if you have a scrum master, I can image he would also need to coordinate with other scrum teams? **A:** No, that absolutely doesn't happen consistently. Both from the scrum master coordination and code consistency in the git pull/request (code) culture. For the technical consistency you can even have distributed teams with 1000's of people working on the same open source code, without a scrum master or a lot of meetings. But they use version control using pull & push requests for code and have reviewers that effectively review to code which ensures some sort of technical consistency. This is missing, and the title of your thesis is... Pair programming right? And this is why it was necessary in our context, to do the pair programming. It was started by a (developer) community lead because it was intuitively needed that we get some mechanism of ensuring some technical consistency & knowledge transfers/sharing because of now other things we could have had, but don't have (yet). **Q**: So it's actually pair programming because the entire adoption and coordination of agile at a larger scale has its company-wide challenges, so with pair programming you align yourselves on an IT implementation level? **A**: Yes, and in our market we now have 4 active contributes to our code, I started a thing called MERGE calls, we don't have merges in GIT, but we do it on calls. So that at least we can collaborate more on the actual technical aspects of the code.	Pair programming initiative as a mechanism of ensuring some technical consistency & knowledge transfers/sharing when supporting tooling (& processes) is not (yet) in place.	(Prevention/Solving of) Misalignments	Motivation
Q: How has pair programming motivated development or developers or people working with your D&A space? Has this motivated these roles in pursuing better alignment? **A**: By my opinion is yes, I would even say in a fairly big way. .. Sometimes you learn something very valuable, or you see something that is maybe not ideal and in both cases there is a big benefit. I can see from our team great things (code) how to do things better but also some flaws that were there in the code, which makes room for suggestions of better code.	By sharing with each other developers are able to write code better & faster, either from inspiration or re-use, and/or discover flaws in the their current work on common/similar subjects.	Perceived benefits (of PP)	Motivation

Q: Did you have a feeling with certain sessions you did uncover issues from other people or different alignment or understandings? **A**: Yeah, we were, yes, there is always something that is not how we imagined it at first, or change from your own idea on how to do something because suddenly you find a better way and gladly take it from someone else and re-use idea's.	By engaging with each other through pair programming, you also share each other's perspectives on how to develop a solution (tools, documentation, conventions, approach,…).	Different perspectives	Motivation
", in general this pair programming that we did is an intuitive attempt to close some gaps in skills, in scaling applications (from or to other markets), which was very necessary at the time."	Different levels of maturity exists between or within teams. Teams need to work on a priority that is already worked out by another team/market.	Scale knowledge & IT implementations	Motivation
Q: Do you ever have any champions or motivators that drive such sessions? **A**: This might not be pair programming as mandates by XP methodology, so we don't have such people that would be fixed on XP and then trying to implement pair programming by the book. ….*But then*, people who have made an attempt to organize and document and find pairs for people, those people are there, but me being not one of them.	Champions or motivators are there that take effort in organizing sessions and form pairs across different teams.	Champions/Motivators	Stakeholder involvement
Q: Does programming lead to more transparency across different agile teams or development tracks (or projects)? **A**: Yes, absolutely. *We do have a lack of interpersonal contact*. We aren't in the same office, we have limited time in meetings to discuss items. Our common calls on Tuesdays with 15 people or more. That is too many people making it impossible to deep dive into low level technical things. So pair programming 1:1 is where you can really spent time discussing just the output of one single function in the code, for more than 2 hours if you want.	With pair programming, there is an in-depth dive into a specific problem or code which is not possible in large or short meetings. Teams are transparent with each other down to a code level, even without being in the same team (or country).	Transparency	Interconnections
Q: Do you perceive if any information connections or relations are made during such pair programming sessions? **A**: The collegial side of things, we do discuss personal matters during such sessions. It's a good experience for socializing a bit. **"***We do have a lack of interpersonal contact*.**"**	Pair programming can create informal relations between the participants.	Existing/creating informal networks	Interconnections
Q: Do you perceive that you already had a lot of benefit from those pair programming sessions? **A**: Yes, definitely, even with the only few that we did on the regional level it was	Pair programming can create informal relations between the participants.	Existing/creating informal networks	Interconnections

great, it's not just working on a task that you are currently engaged with, it's also for uhm general conversation that happens during those sessions: tooling? What kind of plugins for IDE we use? Tricks? Without being in the same office and without those '*coffee breaks*' it's very useful to spend some time together around certain problems, but also to cover general topics of work & productivity. **..**			
.. **Q**: So it's definitely helping to execute your work more efficiently? **A**: Yes absolutely, absolutely, I'd even say it's almost necessity to get anywhere and realise something [within this environment]	By sharing with each other developers are able to write code better & faster, either from inspiration or re-use, and/or discover flaws in the their current work on common/similar subjects.	Perceived benefits (of PP)	Motivation
Q: Yes. And to my next question, which tools do you use during pair programming sessions to facilitate those calls? **A**: With the basic tools, MS Teams for screen sharing with or without video-chat. Something to write the code in or draw out some diagrams or documentation in Confluence. MS Teams also gives the option to take control of another screen making you able to type & click on someone else their devices while they can watch.	The pair programming sessions are all remote. Screen sharing and switching (executor/watcher) roles make supporting tooling a necessary requirement to even start engaging in such sessions.	Supporting tools	Interconnections
Q: Can common guidelines originate from a session? **A**: That is good point, a lot of learning emerge from such sessions. **Q**: Do you document or discuss them with a larger audience? **A**: Yeah, it does happen, but I currently can't think of a few examples now, but people have been bringing up their findings that could potentially be developed into guidelines for the larger thing/architecture that isn't usually covered by pair programming. **Q**: Can I also understand that the shared meetings that you have is also used to share those findings (if relevant)? **A**: Yes	Pair programming sessions can produce proposals for new coding principles that can be discussed together with a central group of stakeholders or community.	Common guidelines & central coordination	Alignment decisions

Q: Did pair programming do any improvement on supporting infrastructure in terms of alignment across teams that you know of? **A**: That could be beneficial but mostly improvements on that are discussed during our general community calls with more stakeholders present. That format is used more to discuss our grievances and infrastructure problems. At least there you have more effect, because more people that experience the same problems are listening in. Then they can also vote for the importance of such (infrastructure) problems and can also vote to have it fixed. More easy switch to action **Q**: But then potentially you can find such issues during pair programming that is escalated during such larger meeting (snow-ball effect). **A**: Yeah, definitely.	During PP, discussions or issues on supporting architecture can help escalating the issue and eventually voicing it to the right stakeholders and channels	Supporting infrastructure & central coordination.	Alignment decisions
Q: Do you feel that it influenced other team's code or the accountability or mandate of certain jobs that you can think of? **A**: Accountability in general, I can't know, I don't know how to comment on that? **Q**: Or the planning & monitoring of tasks? Has a session ever impact ongoing tasks? **A**: Uhm, with those sessions you have do have more follow up with other people (within or outside a specific team). *This can have a positive influence on task monitoring and subsequently on accountability, (your code can be seen by other people)*.	Because code is more shared and meant to be scaled, it should motivate in qualitative code, usable to other markets.	Accountability & Mandate	Motivation

Interview B			
Interview Quotes	**Code**	**Alignment facilitator**	**Facilitator Category**
Q: But here I'm really looking towards your motivation? Like the influence it has on your day-to-day task, or the motivation you have to do your tasks in an aligned way? "Hmm what I just seen, inspired me to use it like that as well and implement my code in the same way, or improve on that" **A**: Yes, it somehow affected but not 100%, this was built up because haven't done pair programming for some time now. But I still see my personal style of working improving towards the direction. I'm now documenting more & more things. When you try to solve a problem, you have your mind focussed on that and you do stuff and you forget about other things like documentation. But it helps moving forward, it really helps to others understand what you have done. On that aspect yes, it put me towards that direction. It gave this initial hint or 'kick' for me to start: "ok, this was a good example, let's document that". Let's try and experiment with what I learned today. Let's take it from there and search for more information how something is done. *Now that I think of it*, I had not a pair programming, but an exchange trough chat with some stuff with other guys and a guy came back with some kind of solution with was not fitting 100% but I took it and customized it to my needs in the background, what helped me a lot. But that example wasn't purely due to pair programming. But this discussion was kicked off from such an (unfinished) pair programming session. Even things that can be exchanged, not related to the specific task, but while I'm programming I'm making some notes and did more into something. New idea's, new tools or ways of working can be learned from for instance you or [other Regional Data Engineer] during pair programming which I actually still apply now.	By sharing with each other developers are able to write code better & faster, either from inspiration or re-use, and/or discover flaws in the their current work on common/similar subjects.	Perceived benefits (of PP)	Motivation
"…New idea's, new tools or ways of working can be learned from for instance you or [other Regional Data Engineer] during pair programming which I actually still apply now. The same naming for example is one aspect that really helps towards alignment." **Q**: Can you give an example on that? **A**: The naming conventions… We are going to name our fact tables start with F_ and dimensions with D_ and so on.	Learning of principles on naming, coding approach and tools that are used within the company.	Training & Learning	Motivation

Q: But this should be more centralized in coding principles, but you learned it more through pair programming? **A**: I did, I learned it mostly through pair programming, like naming conventions, the different data layers and how to handle & store data files within our own country folders. Yeah.			
Q: At any point in those sessions, did you uncover misalignments between teams? **A**: Yes. For sure. **Q**: You have some examples that you can think of? **A**: On top of my head, uhm, stuff about how the bounces were calculated for Google Analytics or the case where we were considering interaction model towards customers and specifically how event data were implemented there, it made it a Frankenstein table as it's data was in their on another granularity (no towards a customer). This was a misalignment in my eyes (between regional & local), which was also the case for other local markets/countries. This was a big thing not that have that in there (it didn't feed the expectations). So now it's being worked out in the scope of our new project. But again, coming to the initial discussions there was a misalignment also on the business level. *It's not a matter of implementation, but it's a matter of business decision/alignment on how to treat the rules and the data that we have.*	Because of the knowledge transfer that occurs between participants, both parties can uncover different, new or better ways to code or create a solution during pair programming, in this case different calculations for "bounces" or how "event data" is across different teams or markets.	(Prevention/Solving of) Misalignments	Motivation
Q: Given your more decentralized organization and marketing teams, it can get very specific within a market because of culture & so on, so customization can be useful but you do start from a base. How did pair programming at any point in time help with this? How did it come into the picture? **A**: It helped me a lot because I wasn't a heavy coder let's say, but more a low code profile. So first of all it helped greatly with knowledge transfer, not only the business, because I felt like I was closer to business knowledge than to technical and coding knowledge. It helped me understand how things should be implemented, what each line can do, for me at least, I don't know for other guys. Putting this knowledge into practice is the best way to learn. On the job training is the best way for me. I consider the pair programming not only a way to ensure code consistency and knowledge of what each code does, but also general knowledge transfer sessions. I had the guys and vice versa in my cases where I was leading the pair programming and doing the knowledge transfer, explaining what my code did. I believe that either way it helped me greatly understand what the code does and helped me align on how code should	Different levels of maturity exists between or within teams. Teams can work on a priority that is already worked out by another team/market in pair programming to transfer (technical) knowledge.	Scale knowledge & IT implementations	Motivation

be shaped. This in turn can ensure consistency of it, but also greatly teach me coding on the job. Felt like a great experience.			
Q: In pair programming session you have different perspectives represented? **A**: It's purely technical, there is zero business involvement. Our pair programming sessions aren't a standard way of working, but an experiment, so I'm not expecting a business representative. But pure technical sessions. -- **Q**: Did you learn any new perspectives on ways of working from other stakeholders? **A**: I think that I got some new ideas on the way of working namely that I could work with other tools and how to better document my code and use naming conventions: how to name my tables & fields & so on. This helped me a lot.	By sharing knowledge with each other through pair programming, you also share each other's perspectives on how to develop a solution (tools, documentation, conventions, approach,…).	Different perspectives	Stakeholder involvement
Q: Do you see those weekly calls to propose those calls, do you see the involvement of any champions or motivators that drive this? **A**: I could say that if there is an experienced programmer, they should be considered as a champion and on the other hand if there is a topic, for example let's say myself recognizing that we have an issue in the model that we were building (granularity event, previous example). So I could lead that. It's either way, or you become on taken from your experience or from your business experience on a specific topic. I saw such cases where people were getting more leading roles towards specific topics. **Q**: Do you see any unofficial leaders? People taking responsibility without an official mandate. **A**: Yes, one example is myself having some kind of leading role with implementation on customer consent. It was more like a task oriented lead. So people from 2 or 3 countries were coming to me trying to get to me to get some experience and knowledge on how to implement that. Or other hand we saw that people with great experience specific projects were becoming the centre of interest when other countries wanted to benefit from their knowledge and their experience. Know that this also helped countries having as a centre,1 person for a specific project that also helps towards getting accustomed towards a specific implementation and speeding up to process of scaling up things to other markets.	Each experienced developer taking part could be seen as an unofficial leader for a specific topic, in which this person is experienced. For a specific subject area someone can be the contact point for that specific knowledge and share his experience with other markets.	Unofficial leaders	Stakeholder involvement
Q: You might have already answered this a bit, how do you engage in pair programming sessions? Or through which channels do you engage in those sessions?	The pair programming sessions (in the case organization) are all remote.	Supporting tools	Interconnections

A: More or less the tools we all use and are aligned for our roles: the same database client to connect to our central repository, test statements in the used IDE with version control.. **Q**: But specifically for pair programming? **A**: We specifically use MS Teams were one person shares the screen and write code whilst the other guys takes notes and an extra set of eyes looking at what is being done, maybe correcting stuff. So noticing: "oh you made that mistake, you don't need to do that, but do it like this". We are exchanging places between coder & watcher (for instance a switch each hour).	Screen sharing and switching (executor/watcher) roles make supporting tooling a necessary requirement to even start engaging in such sessions.		
Q: Have there been any alignment decisions that were made through pair programming sessions? **A**: Yes. **Q**: Could you please elaborate on an example? **A**: For example, I made a decision myself where *I told the business* on how to resolve the problem we uncovered and they accepted it and we could continue with investing in a fix. (the event data granularity) *This was identified through a pair programming session* and worked on later on in next releases of the model…	Decisions on how to write code can be made during a sessions, when two experts pair with each other to solve a (common) problem, but it depends when the issue resides.	Emergent decision-making	Alignment decisions
..The other example was the A/B testing in a session that we had. We realized that the naming convention or a taxonomy thing which I can't fully recall (some months ago) and we realized we could not properly track the A/B testing e-mails because each country had a different approach on how to do that in marketing tool. We couldn't properly identify the A/B testing e-mails from the others. When we realized at that time, we immediately could get the input from a business stakeholder in [Country/Market] during our sessions. And when we were sure about our issue we escalated towards product owners/ regional stakeholder in order to make a decision on how this should be tackled. **Q**: I do remember the SFMC team created common guidelines, put them on a central page so that marketeers could use that and update the usage of the marketing tools (in terms of naming).. **A**: This was also similar to the google analytic bounces. It was somehow miscalculated which I realized during pair programming session, after which I investigated myself after which I came back with a guide on how to correctly calculate the bounces. I	Pair programming sessions can lead to the uncovering of issues or decisions that need to be handled within centralized groups of relevant stakeholders.	Common guidelines & central coordination	Alignment decisions

shared this during our weekly calls as well.			
Q: You feel comfortable with code consistency across your markets and so on? **A**: Hmmm, across markets you say? To be honest I don't have a clear view on that on the larger scale. So I would be more experienced and knowledgeable about answering this question at a later stage when our project is going to be scaled to other markets. Now in smaller cases, it was way back in time when I had to scale something up or scale in, and, I felt like very comfortable with the communication that we've had within our teams. I was not pretty sure about the code consistency itself and why I'm saying that is one of the reasons was the code versioning tools that we didn't have in place. So it was a real challenge to maintain the consistency but we had a very good communication, maybe better within our way of working by having alignment calls for each market where each had their own code versions so that things were not mixing up (a lot). **Q**: So you mean scaling that you are picking application from one market to another? **A**: Yes, exactly. **Q**: So sharing the same code.. **A**: Yes exactly, sharing the code but each creating their own version of the code.	Different levels of maturity exists between or within teams. Teams can work on a priority that is already worked out by another team/market in pair programming to transfer (technical) knowledge.	Scale knowledge & IT implementations	Motivation
Q: Given your more decentralized organization and marketing teams, it can get very specific within a market because of culture & so on, so customization can be useful but you do start from a base. How did pair programming at any point in time help with this? How did it come into the picture? **A**: It helped me a lot because I wasn't a heavy coder let's say, but more a low code profile. So first of all it helped greatly with knowledge transfer, not only the business, because I felt like I was closer to business knowledge than to technical and coding knowledge. It helped me understand how things should be implemented, what each line can do, for me at least, I don't know for other guys. Putting this knowledge into practice is the best way to learn. On the job training is the best way for me. I consider the pair programming not only a way to ensure code consistency and knowledge of what each code does, but also general knowledge transfer sessions. I had the guys and vice versa in my cases where I was leading the pair programming and doing the knowledge transfer, explaining what my code did. I believe that either way it helped	Learning of principles on naming, coding approach and tools that are used within the company.	Training & Learning	Motivation

me greatly understand what the code does and helped me align on how code should be shaped. This in turn can ensure consistency of it, but also greatly teach me coding on the job. Felt like a great experience. **Q**: You also mentioned before you weren't that experienced in coding & technicalities, but your projects are beginning to be more complex, but since you can sit together with other experts, it's ideal for knowledge transfer & alignment and it greatly helps you. **A**: Greatly, greatly yeah. But on the other hand, just a comment here, I think I had shared with you when we did our own sessions. Maybe this is creating, when you are having a experienced & not so experienced person in pair programming, this is bringing the experienced one a little down and the less experienced a bit up, in order to align to the way of working. So on the one hand it could delay things? At least in the beginning, but, to my mind this a price I would be willing to pay, when I have you know inexperienced people linking up with experienced profiles. Maybe in the beginning you might have a pitfall & delay, but this will pay off in the period after (longer run.			

Interview C			
Interview Quotes	**Code**	**Alignment facilitator**	**Facilitator Category**
Q: How has pair programming with other developers motivated you in perusing better alignment? **A**: To me it's a given, in every pair programming session, the alignment. While you are looking, you are generally looking at things: on the code, on the deliverables, on the process, whatever you are looking at. And alignment is a part of that. You make sure you keep in sync with each other: no only on your processes, but also in terms of our usage of the right folder structure, naming conventions, right processes, are we using best practices? All that is part of the alignment amongst you and the other person… If a person creates code like that, it's also part of governance and synchronizing with each other. "Hold on, it seems you aren't properly aware of that common practice". It might not initially be a problem in the beginning due to developers being in different markets, but might give problems later on down the road if the practice is not applied [avoiding integration issues with other teams, that require more rework]	Pair programming initiative as a mechanism of ensuring some technical consistency & knowledge transfers/sharing when supporting tooling (& processes) is not (yet) in place.	(Prevention/Solving of) Misalignments	Motivation
Q: Did you ever have pair programming sessions where you felt motivated to align with a specific market team because of what they are doing? **A**: Most of the pair programming. Even on both ends, as an peer teacher/student, if you are the one having the problem and you need a solution using a pair programming session. You are Enlighted, you dive into something because your problem is being solved with help and you can deliver. If you are on the giving end. It's a bit of your time you need to give or your input to solve it, but it also gives you a good feeling that you were able to help a market. The team [community/company] as a whole is moving forward. But as a team you are better. This dripples down outside of your team, to management you make it visible that markets are helping out each other, investing time in others. Overall the team is going to move faster as well. You have agile way of working for yourself, but also as a team and as a department/division. All small effects that you do, you need to keep your helicopter view. And if you do many of those where everybody helps everybody: in the end you will move faster as a whole. For instance when you need to integrate a new model into your application, and you aren't fully aware of it at first. But it exists. Together with a Subject Matter Expert, in a PP session, that person can explain you the bits and pieces of that model so that you can get started with it, and dive into it in a proper way. This motivates you because you learn something new in a fast way and you can	By sharing with each other developers are able to write code better & faster, either from inspiration or re-use, and/or discover flaws in the their current work on common/similar subjects.	Perceived benefits (of PP)	Motivation

continue in an 'unleashed' way. Your colleagues are also happy you can re-use existing work. If you can deliver faster, you manager is also happy.			
Q: I also understand that in most of your sessions you combine experienced & less experienced people in pairs. So 'champions' are almost always involved, either with a less experienced or other expert. **A**: It can depend on the problem, either 2 experts to solve a new problem, or a subject matter expert to rework an existing solution and do it together. **Q**: Do you feel there are any unofficial leaders/experts within the developer community? **A**: Everyone is an SME in my opinion, everyone here has their own specialty.	Each experienced developer taking part could be seen as an unofficial leader for a specific topic, in which this person is experienced. For a specific subject area someone can be the contact point for that specific knowledge and share his experience with other markets.	Unofficial leaders	Stakeholder involvement
Q: More in the context of: "You have a subject about matter A, then you best go that person"? **A**: In setting up the pairing, we had a triggering person. A person introducing pair programming session and that pulls a bit on using those principles within the team. In terms of decision making on what problems to solve using pair programming, that is usually the team or the manager overall I think. If you go towards the session, usually the SME is taking the lead. But in terms of planning on when.. there is no lead in it.	Champions or motivators are there that take effort in organizing sessions and form pairs across different teams.	Champions/Motivators	Stakeholder involvement
Q: Did you ever learn any new perspectives or ways of working from other stakeholders/experts? Examples? **A**: It's a bit half-half with the role I'm in. If I'm on the receiving end (4O% of my sessions let's say), I learned a lot. It basically opens doors. **Q**: Do you know of a specific example? **A**: We work with a specific tool for reporting & BI. In there you have different levels of knowledge, methodology or ways of working. You can do things in such different ways.. How you use variables, balance between logic in application and scripts. How far can you go in there? With scripts you can build different levels, sub procedures, things like that, those kind of introductions, I learned a lot from someone else. Sometimes it was (indirectly) started by a colleague, by guiding others through the application in a demo and share how he solutioned it. I've had cases where "I need to have the same in my application as I'm running against the same" or "I need to know about that as I need the same in my application", "can we have a pair programming	By learning from each other through pair programming, you can also share each other's perspectives on how to develop a solution as there can be many different ways in creating an application or solving a problem in one.	Different perspectives	Stakeholder involvement

session?". If you don't have a specific solution for your problem and another application (or part of it) can be leveraged to re-use that and develop faster. Again it depends on the problem, but scenario's like this very typically go through a pair programming session about technology & code. More advanced coding & framework solutions. **Q**: So even as a subject matter expert you still learn from others and from other perspectives? **A**: Yeah because overall you can be an SME on a certain topic, if I take for example Python… you can be an expert on the language but Python also uses different kinds of modules. You can do the same thing using different modules. And you might be an expert on using that particular model. For example python is programming language and you can intertwine it with excel and automate that within python, but you do that XLS-wings which is a specific module you can use, but you can also use OpenPyXL, which is a different module and a slightly different way of working. So you can be an overall expert but you might still reach out to an expert using other modules.			
Q: Is there any formal governance used or needed to organize pair programming sessions? **A**: There are some documented rules & best practices. Adhere to standards if any, organize around a specific problem,… Overall pair programming, there are some best practices from an Agile Manifesto perspective brought into the company, <u>but aren't enforced by the company</u>. But within the sessions we are expected/needed to adhere to those standards. Allow switching of roles, try to respect a 4 hour limit, not stop halfway because of other meetings, make sure you keep being productive, bring your good mood in the session,.. .. We do stand-up meeting with my own 'core' team, but we have weekly meetings with the developers (the developer community) for technical topics & problems where pair programming can be organized. So a technical meeting across different teams/markets. SO if it's not through chat or e-mail, usually it's discussed/proposed during that weekly format. From that discussion, a pair programming session can be organized out of a question, problem or new subject area..	Rules & best practices, based on industry standards are documented and expected to adhere to, but not formally enforced top down from an Agile perspective.	Formal Governance	Interconnections
Q: How do you engage with each other? Do you use specific tools? **A**: We first align agenda's together and connect through teams. Usually the one with the problem shares his screen as you dive into the problem. Along the way you can switch. Depending on personality. The SME might show how to do it, by taking over	The pair programming sessions are all remote. Screen sharing and switching (executor/watcher) roles make supporting tooling a	Supporting tools	Interconnections

the screen. But it might also depend on the personality of the receiving end: "I want to learn, tell me and I'll do it". On the longer term that can be more efficient if the person learns more that way is how I look at it. **Q**: Have you used those tools like Teams, screen sharing,.. Have there ever been tools someone else uses that made you switch to the same? **A**: Not really from my end I think. We mostly use the same common platforms. Maybe different whiteboard tools to share diagrams online.	necessary requirement to even start engaging in such sessions.		
Q: How does pair programming session lead to transparency across teams? **A**: Across Teams? We always align within our team. You have the core team and also differe.. **Q**: With team, you mean the developer community? **A**: Yes **Q**: The other markets and their developers (together with local/core business owners etc) still operate in separate small agile teams, they have their own board & priorities set by their management. You have near or over 10 markets! I'm looking for transparency across those different teams. Examples? **A**: In that case, most sessions were across the different teams. We do stand-up meeting with my own 'core' team, but we have weekly meetings with the developers (the developer community) for technical topics & problems where pair programming can be organized. So a technical meeting across different teams/markets. SO if it's not through chat or e-mail, usually it's discussed/proposed during that weekly format. From that discussion, a pair programming session can be organized out of a question, problem or new subject area…	With pair programming, there is an in-depth dive into a specific problem or code which is not possible in large or short meeting or workshop. Teams are transparent with each other down to a code level, even without being in the same team (or country).	Transparency	Interconnections
..If it's something big or new that all the countries need to use and need to be introduced (a specific module, way of working, framework,…). Then it's going from the regional/core to every market. We have done it in the past where we say: instead of a big workshop that isn't efficient because everyone learns at different speeds and also there might be customization needs due to differences between markets (language for instance): let's organize multiple pairing sessions where a few subject matter experts align with each market individually and do it that way. Because of customization or/and experience level of the person within the market. Each market	Different levels of maturity exists between or within teams. Teams need to work on a priority that is already worked out by another team/market and transferring knowledge & experience.	Scale knowledge & IT implementations	Motivation

can have a different level of expertise or background, luggage, experience from the past which someone else might not have. That was one of the reasons to have more pair programming sessions instead of a whole workshop. Especially big workshops electronically within a team's call, it doesn't always work well in my opinion. At least not for a lot of our scenario's.			
Q: Are there any informal connections or relations across teams created with pair programming? **A**: Usually there are some kind of afterthoughts or an SME in session explains how to integrate or solve problems using a certain practice or informing in on the technical things that person is to do. After the session, that person that needs to solve the problem, is continuingly solving the problem after the session as well. So he might still face little issues that were not clear from the session that can be clarified informally through separate (informal) chat, email or call. Usually there is. Even if it's just to say thank you for the help and so on.	Pair programming can create informal relations between the participants.	Existing informal networks	Interconnections
Q: Have there been any emergent decisions? For instance I was in a session regarding A/B testing (basic marketing method) where we briefly included someone from the platform team where we raised the issue, which was followed by the creating and sharing of new guidelines on the related marketing platform we had the session about. **A**: Yeah, and with different scales. Decisions on level of code: writing code in a specific way (or convention) which everyone should follow. But it can also be something bigger, for example, if part of the framework is changing, you might need to decide in the pairing session on the approach. But then, the bigger it gets, the more chance it's two SME's that sit together in a session and decide on the best approach. They should have the oversight on technical dept and framework to make good decisions. So decisions can be made at different levels. When you want to solve a problem and you are on the receiving end where you receive knowledge. There or after the session, you can still decide to use that approach or guidelines that were shared whilst there being more than 2 options. Decisions need to be made at different levels. For me this also fits within agile, if a decision or it's effect didn't work, then change it with a new task on the board.	Decisions on how to write code can be made during a sessions, when two experts pair with each other to solve a (common) problem, but it depends when the issue resides.	Emergent decision-making	Alignment decisions

Q: So you can have emergent decisions during such sessions, but if you have any decision topics or issues that also arose during pair programming: can that be picked up by a central group of stakeholders? **A**: Depending on the impact of issue. If it's something broader, it can brought be during a session and raise a ticket to discuss in a stand up or weekly meeting for a broader discussion and decision (or further escalation). Usually that is how it goes: Can we decide on our own? Apply the change and move forward. Or is it something bigger and need to go through broader formal governance. Then let's create a task and bring it on the table at the right level.	Should decisions need to be made with a broader group of stakeholders, there is a central group or format where this can be discussed.	Central coordination	Alignment decisions
Q: Has something like that resulted in creation or updates on common guidelines for IT implementations or even usage? **A**: There are a lot of cases, but I can't come up with a concrete example now. But I think that is how things just evolve. **Q**: I think I earlier gave an example of that, but that is just my own (platform guidelines). But if you could think of one like that. **A**: It can also be a mix of the 2 (chuckle). For example. We are building applications and normally these days, we each work on our own application which we develop and deliver. But along the way I always try to improve the end to end data flow (source -> application). So that things can be more efficient. That together with certain switches in ways of working in our data flows, so my development is already focussing on the future structure to create an easier switch in the future. **Q**: Did this come from pair programming? **A**: Pieces came from a pair programming session. Initially I envisioned it on my own, but then, after a pair programming session with another SME, "we might have a problem or not" and I was showing my idea and data flow and if it would facilitate an easier switch in the future. It's to increase efficiency for tomorrow. We decided on a way of working, so that we can switch easy in the future, including other markets. **Q**: So you had a problem and pieces the solution together by doing a pair programming session with other SME which resulted in guidelines for data flows. Did you create awareness for that to other teams? **A**: Yes, because other data engineers & application builders started to use to same data flows or code on top of a common data platform.	Along the way, or over time code can be improved upon by finding new or more efficient ways of building or even using applications, meaning existing guidelines will have to be updated or new ones created.	Common guidelines	Alignment decisions

Q: And, do you feel if those pair programming sessions lead to any improvement on supporting architecture then? For instance your example of better data flows? **A**: It can lead to that. The main focus of the pair programming is to solve the problem at hand, but like I mentioned, along the way you also mention governance, best practices, naming conventions and if you see.. sometimes you come up with multiple solutions during such sessions. Use the existing way? But for your application you might also want to improve along the way. What is needed or what are the limits to what we can improve? **Q**: So in this case it leads to improvement on that reporting platform and improvement of its infrastructure. Does the platform pick up such items out of PP or that started there? (or other central group) **A**: Usually that is done when application go to production, when the platform team reviews applications before moving them to a production environment, and it follows standards from a platform point of view. **Q:** So you basically have a data flow that was then approved by the platform team after reviewed and finetuned with other SME's first. **A**: Yep. And usually they are also informed beforehand. Because people of that platform can also join stand up calls or bigger meetings, where scenario's like this can be mentioned. It depends from case to case how that goes.	Along the way, or over time other than the problem at hand during the session, more broader topics such as architecture governance, best practices, conventions are also discussed.	Technical infrastructure	Alignment decisions

Interview D			
Interview Quotes	**Code**	**Alignment facilitator**	**Facilitator Category**
Q: A bit more on pair programming again: let's talk about facilitators for efficacious alignment with such pair programming sessions. How has PP motivated you or other developer in pursuing better alignment? **A**: I don't know if PP had an impact in our case for "we need more alignment". The flow of PP is that you stay at a grand level. And at that level we haven't done pair programming before. We aligned but is was more a problem of leadership to be aware that misalignment costs us a lot of money. It's more a discussion with leadership who bring such situations forth instead of the PP itself. In PP you do align even without the feeling you align. **Q**: Does uncovering misalignment act like a motivator to start using pair programming? **A**: Yes, I proposed it because everyone was going in different directions for the same or similar project. We had a lack of alignment and it was a way to algin better the people developing (scaling) the application. **Q**: So, now you are working on different project but are looking back to pair programming to share on those projects afterwards. **A**: Yes, align & share.	Pair programming initiative as a mechanism of ensuring some technical consistency & knowledge transfers/sharing/scaling.	(Prevention/Solving of) Misalignments	Motivation
Q: I'm going to ask that question a bit differently: have such session impacted a project? "What I learn now, I'd like to re-use and that will make my current tasks more efficient". **A**: In that way yes. It impacts projects. In a way I would like to be impact. For instance, when a project can be adjusted to make it more scalable, so that I can scale it more easily to my market. We can share the good practices & vision on data. When you PP on other projects is a way as well to share good practices and align on the best approach.	By sharing knowledge with each other through pair programming, projects are impacted in a positive way due to the sharing of good practices and vision on data so application are also more scalable towards other teams.	Benefits - Planning & Monitoring	Motivation
Q: For your stakeholder investments: who is typically always involved in the program? What is you stakeholder representation? Are different perspectives represented? **A**: On the business side was very transparent. We just mentioned it, "ok it's a technical thing", they don't really care how it works, just that it works. From IT leadership we had a good support to make the PP experiment. Because we were aligned about the,	IT Leadership needs to support the usage of pair programming, as in this context it is organized across different teams [/Markets].	Leadership	Stakeholder involvement

we are seeing that there are alignment problems and PP can help, it will not solve everything but it can help to go towards a better solution. …			
... Also to reduce the different maturity levels across the different markets. That was one of the other points: the different skills levels across the local markets in the community. To fix the different maturity levels of senior & junior developers a pair programming sessions set up was created to remedy together with the different alignments. [local discrepancies]	Different levels of maturity exists between or within teams. Teams need to work on a priority that is already worked out by another team/market and transferring knowledge & experience.	Scale knowledge & IT implementations	Motivation
Q: Do you ever include people from outside of your community? **A**: No, external people (contractors) are preferably avoided in PP sessions. With the exception of regional consultants. We do have local contractors that develop, bet they are not part of the community for pair programming. It's a sensitive topic however. My personal point of view: maybe we are wrong with that, we should be inclusive. <u>But, it is very understandable</u> as it's a local market that is providing the budget for such consultants, those should not engage in effort of other markets, when other markets don't invest in external consultants. I'm more in an open source mentality. What is good for one, is good for others. If everyone was like that it would be better, but it's my point of view, not business. **Q**: Well, it's then clearly more a budgeting & organizational discussion. **A**: Yes but it costs us efficiency. We have some very good developer contractors. But they do it their way and it may not be fully aligned in the way we do it ourselves. Meaning the whole is less maintainable because of the differences between our markets. **Q**: Where there any external stakeholders included in those sessions? **A**: At the beginning we had the regional scrum master who helped us set up the basic methodology.	There is little to no (continued) involvement from external stakeholders, which can cause misalignment between some teams, this is however a problem that sits outside of the IT implementation's context as it involves budgeting of resources & teams.	Internal & External Stakeholders	Stakeholder involvement
Q: Are there any champions or motivators involved in such sessions? (interpreted: unofficial leaders or experts). **A**: In [Market/Country X] we have someone who is a good senior and advocates data governance. And yeah, there are several people who take a 'lead' on several subjects depending on their experience.	Each experienced developer taking part could be seen as an unofficial leader for a specific topic, in which this person is experienced. For a specific	Unofficial leaders	Stakeholder involvement

	subject area someone can be the contact point for that specific knowledge and share his experience with other markets.		
Q: You already mentioned you were doing it digitally? So through a channel that facilitates screensharing/camera? **A**: Aha! Yeah, because we mainly do PP on code in scripts, we are working with MS Teams; one sharing a screen and showing IDE to write, another to look. In PP the main idea is to have a driver and a coder. Only the coder touches the code while the driver watches and explains or corrects if he sees a mistake. We don't have unit testing in [our reporting platform] but in some methodologies, the driver should make the unit tests.	The pair programming sessions are all remote. Screen sharing and switching (executor/watcher) roles make supporting tooling a necessary requirement to even start engaging in such sessions.	Supporting tools	Interconnections
Q: Is there any formal governance used to organize those sessions? How are pairs planned? **A**: The planning of the session we do on confluence, a central information point and during the weekly community meeting we can discuss who wants to pair with who on what subject. On the confluence we document who pairs with who, related to a ticket. Thus, organization is done during community meeting & confluence page *"From IT leadership we had a good support to make the PP experiment. Because we were aligned about the, we are seeing that there are alignment problems and PP can help, it will not solve everything but it can help to go towards a better solution"*	Rules & best practices, based on industry standards are documented and expected to adhere to, but not formally enforced top down from an Agile perspective. IT leadership does support the usage of PP.	Formal Governance	Interconnections
Q: Did you ever learn any new perspective or other ways of working from other stakeholders? **A**: Myself not really, but because I usually am the expert, I was the one that proposed the PP set up to address the problems we had. From the [Market/Country X] person I received an interesting share of his practices, him having more of a PP background. **Q**: So it's more from you that other learn? **A**: Yeah, and, learn from me and from doing. I for instance knew the (PP) methodology theoretically and we are now learning by doing it.	By sharing knowledge with each other through pair programming, you also share each other's perspectives on how to develop a solution (tools, documentation, conventions, approach,…).	Different perspectives	Stakeholder involvement
"because I usually am the expert, I was the one that proposed the PP set up to address the problems we had"	Champions or motivators are there that take effort in	Champions/Motivators	Stakeholder involvement

	organizing sessions and form pairs across different teams.		
Q: From those PP sessions: do you have also have informal connections coming out of that? **A**: I have discovered some people during the pair programming sessions, because outside of PP you just talk professionally. With the community we can go more casual & feel-good topics. It's why we avoid mgmt. in those meetings. But in PP you have more time to discuss and discover other people on mood, psychology, culture & so on… Looks a little bit like a tattoo session.	Pair programming can create informal relations between the participants.	Existing informal networks	Interconnections
Q: How do you feel that PP sessions lead to more transparency across teams? Any example? **A**: When you peer with someone from another country, you peer on a local project you see what they do and discover how they do it. And for that it gives you a more transparent.. you know more about what is happens in the other markets. This is good as you can see what is common from market to another and what not. For instance: language can be very important data, but not the case for others where you only have 1 language. In Portugal they just speak Portuguese, in the Netherlands it's only Dutch, in Belgium there is French, Dutch and even German. This kind of information can have influence because cultural differences can give different behaviour on marketing. And pharmaceutical products can have different regulations with the federal government. **Q**: That relates also a bit to those different business needs that you mentioned. **A**: Yes of course, and for that our market has different needs than for instance [Market/Country A] or [Market/Country B]. We don't have the same way to manage priorities (of customers) or management of sales representatives. That is interesting to see. Towards the principle benefit of PP; when we became aware of that and when they (region) begins to ask regional development dashboards we are more aware were will be the pain points to do it. ..	With pair programming, there is an in-depth dive into a specific problem or code which is not possible in large or short meeting or workshop. Teams are transparent with each other not just down to a code level, but also make the numerous differences (and resulting needs) between countries more clear.	Transparency	Interconnections
.. *Towards the principle benefit of PP; when we became aware of that and when they (region) begins to ask regional development dashboards we are more aware were will be the pain points to do it*. **Q**: So by doing all of that, the moment you go to a higher level project, you have more knowledge of the gaps between your markets to get that bigger or common goal?	Pair programming initiative as a mechanism of ensuring some technical consistency & knowledge transfers/sharing/scaling.	(Prevention/Solving of) Misalignments	Motivation

A: Yeah for sure. Because for example, we have some market that use priority by customer, while others use priority by product and when we make a regional dashboard but you have to manage the different priorities. I don't know if it helps us to find solutions, but at least we know there is a problem. **Q**: Well, first step of solving a problem is recognizing there is one. Sometimes it first needs to be visible.			
A: And for that it gives, I will not say transparency, but it gives more visibility between the markets. And what is interesting to see is that we are working more in silo's, but I know perfectly what is going on in [Market/Country A] or [Market/Country B] within our region, but not in [Market/Country C]. **Q**: I understood [Market/Country C] was also in a separate region organizationally, so that would explain a bit there. **A**: In a way it's sad because the face some of the same problem I would assume. And they find smart solutions and we are not aware of it… One solutions would be to do pair programming across not just different countries within a region but across regions.	With pair programming, there is more awareness of what goes on in different teams on a more in-depth level, limited by the scope of the community that is participating.	Transparency	Interconnections
Q: So pair programming is limited your region, not really global? **A**: Yeah, it was committed more in our own region and if we look at XP methodology it's more about a consistent group of developers who are working together. But. Why not go further and think about peer programming or even working and do it with random employees of the company. It can be interesting to see the result by sharing visibility understand each other's the process. Might be a bit too extreme, but at least what is happening if we peer 2 random engineers within the company? One in Chili and one in Singapore? What will happen in time? **Q**: Well, you'll need a specific task of course. So that both can work towards a common goal. I can't imagine it being completely random, but more in the style: "there is this problem or piece of code that they can work together on"...	Pair programming needs to be supported by leadership at some level before it can occur, when different peers can work on a common problem.	Leadership	Stakeholder involvement
..	Pair Programming initiative as a mechanism of ensuring some	(Prevention/Solving of) Misalignments	Motivation

A: Yeah, for sure, we arrive at the same problem as now, we have less pair programming sessions because there isn't a common project now. **Q**: But that will come over time once pilots start finishing? **A**: Yes, I expect that. I don't know the future, but I expect that. -- **Q**: Can I understand from this that, your IT implementation are taking other directions in local markets: is this because is going in other directions or? **A**: There are 2 different trends that come together: The business needs/ask/priorities are different. We begin to work on several pilots in parallel. In [Country/Market A] they have a local database extension. In [Country/Market B] they are working on a pilot about data governance tool. In [Country/Market C & D] we are working on a data science pilot. So different subjects. The idea is to try stuff in parallel in different countries and having different business priorities, this puts us local developers on different paths. So currently pair programming is becoming increasingly difficult. It's too different for us now. I think we are going back to PP session direction, once all the pilots begin to be scaled at a regional level. We will go back to a more sharing process and scaling process. Then PP can have an interest (again).	technical consistency & knowledge transfers/sharing/scaling. When different teams start to work on different projects, PP can be used to scale applications with each other again and re-align when needed.		
A: There is sometimes no clear role, we have a kind of lack of data governance, I mean that sometimes we need a place to say: Ok, here we define the business concept, KPI's, …" And for now each people make it in their own way, but there is no real harmonization for that and now view to get that more harmonized. Or wishful thinking: "it would be good that we have this", but it wasn't being picked up with someone taking a lead on that, or takes a long time. **Q**: And because of that, have you for instance encountered problems regarding technical consistency of code? **A**: Yes. For sure. Not each country has their own code because we copy paste it from each other, but there is inconsistency between a lot of projects. For example we are working with [our reporting platform] scripts, and sometimes the code is copy pasted (by another team or country). But each time an app is copied there are modifications and all the code begin to diverge from app to app. There is no [clear/detailed] global view…	Pair programming initiative as a mechanism of ensuring some technical consistency & knowledge transfers/sharing/harmonization.	(Prevention/Solving of) Misalignments	Motivation

..It's why I proposed pair programming. To unify it, not from a top down approach, but from a horizontal approach.	Champions or motivators are there that take effort in organizing sessions and form pairs across different teams.	Champions/Motivators	Stakeholder involvement
Q: Those problems you highlighted, how does pair programming help solve that? **A**: A lot of time we make the same stuff in several different ways because we don't know a lot from each other. We have/had markets that work on similar things at the same time. By sharing the knowledge, it can break a few silo's that way in a more flexible way. You can do pair programming with almost everybody and without having a big development program were you work in sprint. You and me we work together today and work on the same task. It's pretty easy to implement. That's the power of pair programming, it's simple, it's only a minimal alignment you need, two people. However, mixing those different is important.	Pair programming initiative as a mechanism of ensuring some technical consistency & knowledge transfers/sharing/scaling.	(Prevention/Solving of) Misalignments	Motivation
Q: So pair programming is used to scale application to other countries? **A**: Yeah or for example of we have a data science project where we begin to have capabilities in [Market/Country X], so we can imagine starting pair programming with other markets for the use cases so that other market can also try in implementing data science.	Pair programming initiative as a mechanism of ensuring some technical consistency & knowledge transfers when sharing or scaling applications.	(Prevention/Solving of) Misalignments	Motivation
Q: Did PP sessions lead to any common guidelines for IT implementations or common models? **A**: <u>Informally yes, definitely. But in a formal way no.</u> **Q**: Could you elaborate? **A**: For example we created a data model that was shared by pair programming sessions, but it was always in 1:1 sessions. There is some referential documentation. But we aren't good at having 1 reference that needs to be enforced across. It's not enforced, so it's not formal. We have a project that is now using a data governance tools, and I hope this will solve our problem a bit. But we'll see, because it's more a process issue than tooling. If the tool is there, maybe we will be more in this kind of process [with more central and basic reference documentation instead of localized] **Q**: Have there been any topics or issues that arose during PP sessions that could be picked up by a central group of stakeholders.	Pair programming sessions can produce proposals for new coding or documentation principles that can be discussed together with a central group of stakeholders or community. However, this community or central group should also be able to provide the appropriate processes to govern this from PP.	Common guidelines & central coordination	Alignment decisions

A: No... Maybe. It's like hypothesis, maybe it's because our leadership is less about technicalities and sometimes it sounds very technical and we as technologist we don't advise enough on the business implications that the technical debt gives. **Q**: But a central group of stakeholder can be a central group of developers that have that technical capacity. **A**: Aah ok. Yeah, there was stuff that was done in documentation and so on by the group. Yes. But uhm…. **Q**: But more informally like with your guidelines or IT implementations or? That is maybe what you mention here. **A**: Yes but now, we have done documentation on confluence as well and. It was a little bit self-learning process to know how to properly do documentation: how to document, how to evolve and we learn from each other from PP, but maybe not so much because the focus there is not on documentation. *But seeing what the others do and doing the same way does occur.* **Q**: So you do have a kind of a centralization of documentation & guidelines, but it's a but unclear as to how it's further picked up..			
Q: ..For instance on the version control which is supporting infrastructure, that is something that has been centrally picked up? **A**: Yes, peer programming & community discussions helped to arise this issue and to define what we need as a community to solve that. As well as to find the right stakeholders to discuss that.	During PP, discussions on issues can help escalating the issue and eventually voicing it to the right stakeholders.	Supporting infrastructure	Alignment decisions
Q: Maybe specifically regarding alignment: What I mean by that is that different people come together and work towards a common or align towards a common goal and how to best get there. My focus is really on how pair programming helps with alignment. Hearing about the issues more related to agile are very interesting, but a larger scope. Let's focus on the method of pair programming. **A**: For alignment, when you pair a senior with junior. Of course there is a big knowledge transfer. Because the senior shows his way of working and by doing so, in this kind of pair, you align in the way that, the junior makes the stuff like his senior. …	Learning of principles on naming, coding approach and tools that are used within the company when pairing experienced with inexperienced resources.	Training & Learning	Motivation
..But the true alignment in the company happens more when you pair a senior with a senior when they see different methodologies, it creates discussions… "Why do you do	By sharing knowledge with each other through pair programming,	Different perspectives	Stakeholder involvement

it like that or like that?". When you have such a pair they have to accept one convention and it's where alignment begins. Because at the end, if each time we begin to agree on some convention, there can be global conventions that can start appearing. When seniors align with each other, we see different ways of working and when you learn and discuss what's the best way of working like that as well. **Q**: Both seniors can basically challenge each other's way of working while a junior would just inherit a way of working? **A**: A junior could also teach something to a senior, because maybe he's seen some new technology while just exiting school? **Q**: If a junior developer doesn't know full well to program, then he can at least learn it from a senior and learn his practices. If you have discussions between seniors: you are only aligning with 2 people. How do you upscale such a discussion? **A**: That is the power of peer programming and why it's super agile. We don't have a global harmonization, but if you do pair programming you begin to align more and more with time. Iteration by iteration. At the end you begin to be very aligned as a group pair programmers.	you also share each other's perspectives on how to develop a solution (tools, documentation, conventions, approach,…). Usually when more senior resources or 'unofficial leaders' align with each other.		
Q: So, because of all those differences between markets and that is why more personalized pair programming sessions are organized to help with the migration. Those pair programming sessions: what are other benefits you could take from such sessions? **A**: The biggest benefit is to see how other markets or regional levels work. Knowing better the person, but also sharing the best practices. In fact it makes a kind of alignment on best practices over time when you mix the people. **Q**: Was it for instance helpful in reducing complexity in some cases? **A**: That is a good question. I hope it will.. (smile), but I'm not sure I have proof of that. But if you don't do pair programming, some stuff is done and some stuff is done in another way or place. So in a sense it lowers the complexity. At least, there isn't so much code, but it's more or less in the same way, making it more maintainable. As code will then better fit expectations.	By sharing with each other developers are able to write code better & faster, either from inspiration or re-use, and/or discover flaws in the their current work on common/similar subjects. At the same time making new informal connections, share best practices and making core maintainable as it gets more recognizable by others.	Perceived benefits (of PP)	Motivation
A: .. What I see this year is that people start to go in different directions. Some go to [our reporting platform], others begin to do data governance or working with a local	Different levels of maturity exists between or within teams. Teams	Scale knowledge & IT implementations	Motivation

data warehouse extension or more towards data science. So now it's harder to make pair programming between two countries because we don't work on the same project with the same skills/tools. And it makes sense to align then, as we are working on different priorities. But we still have pair programming between regional & local markets on regional scoped projects. For example: we migrated our data warehouse from an old system to a new one. During the migration the global data engineers worked with local digital technologist in pair programming sessions. **Q**: So someone from the region will align with each market individually to give them training. In group our really in pair programming? **A**: We had one group kick-off, but *we estimated that pair programming was better on a market level. It's more personalized for the people and you execute & learn at the same time. When you do a pair programming session you executed your migration by immediately trying so solve local discrepancies.* Pair programming is learning by doing. If we do a group session it would just be a learning, but we can't do a migration with all local exception (for each market) in one big session. So pair programming is a better fit.	need to work on a priority that is already worked out by another team/market and transferring knowledge & experience.		

Q: How do you perceive the efficacy of agile development methodologies within the organization or regional department? **A**: It's a tough question. Uhm. Because it's a multinational company, we have some decisions and capacity at a global level, but also on a continental (regional) level. And also at a local market/country level. Inside the country level, but maybe because we are in a small country, it works well because here it's a smaller organization and have less the problem of a large organization. So it works a bit like in a start-up mode. A lot of communication & face to face meetings between the professionals. It works quite well. But it's when we begin to work on the continental or global level it because a big challenge, because, yeah, I perceive a gap in methodology and silo's with certain groups of people not working on the same time scale or same objectives. And then you have more frictions. The problem I face and from what I've seen from previous consultant jobs in other big companies is that: a big organization have a hierarchy and always a top down approach. This makes it hard to have a bottom up approach or give place to a bottom up approach. This uni-directional from global to local, but never from local to global or only rarely. **Q**: Basically it works well in your market because you can work more independently within your team/market? **A**: Yeah and there is also a scale factor & physical factor. Here you can meet and know the people better, not so many stakeholders, you are better aligned.			