FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Effective Communication in Agile Teams: A Pattern Language

Daniel Ribeiro de Pinho

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ademar Manuel Teixeira de Aguiar

July 22, 2020

Effective Communication in Agile Teams: A Pattern Language

Daniel Ribeiro de Pinho

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. João Carlos Pascoal Faria
External Examiner: Prof. Alfredo Goldman Vel Lejbman
Supervisor: Prof. Ademar Manuel Teixeira de Aguiar

July 22, 2020

Abstract

The Agile Manifesto states the importance and reliance agile software development methodologies place on teamwork and communication between team members and stakeholders. People from vastly different roles and backgrounds work together and, as such, it is important to ensure any information relayed between these parties does not have any errors. When this does not happen, unwanted events may occur, such as time wasted due to mistakes that happened, misunderstandings, and a client not receiving what they expected due to the requirements gathered by the development team not being accurate. As such, it is vital to ensure the communication in place is as effective as it can be.

This work aims to address the topic of effective communication in agile contexts. It includes the background about the relevant fields, presenting a list of relevant communication skills and issues to address, as well as describing them in slight detail. A review of three methodologies - Scrum, Extreme Programming (XP) and Large Scale Scrum (LeSS) - is also presented in this document, where each methodology's history, main characteristics, and approaches to communication are detailed. A literature search of existing work is present, where one can find information about how agile methods affect communication and common practices and elements. The area of pattern theory is explored, detailing the history of patterns as a problem-solving tool and explains the key concepts. Finally, the background also provides details about pattern languages related to the topic at hand.

The key contribution was the creation of a pattern language, using a method that employs research, writing and symbolising. The resulting pattern language contains by now thirty-one patterns and proto-patterns approaching concepts regarding both the physical and digital environments, the activities people partake in, the individuals themselves and how they interact with each other.

The pattern language's validation is mainly done by surveying industry professionals using a questionnaire asking about the patterns' implementation in their organisations. The survey results are positive, showing almost all patterns are implemented, most of which in their full capacity. In addition, the pattern language as a whole, along with five of its patterns, was presented at EuroPLoP 2020, where further feedback was received.

Keywords: agile software development, effective communication, patterns

Resumo

O Manifesto para o Desenvolvimento Ágil afirma a importância e o apoio que as metodologias ágil de desenvolvimento de software colocam no trabalho em equipa e na comunicação entre os membros da equipa e *stakeholders*. Há pessoas com papéis e origens muito diferentes a trabalhar em conjunto e, como tal, é importante assegurar que qualquer informação transmitida entre estas partes não tenha quaisquer erros. Quando isto não acontece, podem ocorrer eventos indesejados, tais como perda de tempo devido a erros que aconteceram, mal-entendidos, e um cliente não receber o que esperava devido ao facto de os requisitos recolhidos pela equipa de desenvolvimento não serem precisos. Como tal, é vital assegurar que a comunicação em vigor seja o mais eficaz possível.

Este trabalho tem como objetivo abordar o tema da comunicação eficaz em contextos ágeis. Inclui informação sobre as áreas relevantes, apresentando uma lista de competências e assuntos de comunicação relevantes a abordar, bem como descrevendo-as com algum detalhe. Uma revisão de três metodologias - Scrum, Extreme Programming (XP) e Large Scale Scrum (LeSS) - é também apresentada neste documento, onde a história de cada metodologia, características principais, e abordagens à comunicação são detalhadas. Está presente uma pesquisa bibliográfica do trabalho existente, onde se pode encontrar informação sobre como os métodos ágeis afectam a comunicação, bem como práticas e elementos comuns. A área da teoria dos padrões é explorada, detalhando a história dos padrões como ferramentas de resolução de problemas e explicando os conceitos-chave. Finalmente, também são fornecidos detalhes sobre linguagens de padrões relacionadas com o tópico em vigor.

A contribuição chave foi a criação de uma linguagem de padrões, utilizando um método que emprega a investigação, a escrita e a simbologia. A linguagem de padrões resultante contém agora trinta e um padrões e proto-padrões que abordam conceitos relativos tanto ao ambiente físico como ao digital, às actividades em que as pessoas participam, aos próprios indivíduos e à forma como interagem uns com os outros.

A validação da linguagem de padrões é feita principalmente através de inquéritos a profissionais da indústria, utilizando um questionário que faz perguntas sobre a implementação dos padrões nas suas organizações. Os resultados do inquérito são positivos, mostrando que quase todos os padrões são implementados, a maioria dos quais na sua totalidade. Além disso, a linguagem de padrões como um todo, juntamente com cinco dos seus padrões, foi apresentada na EuroPLoP 2020, onde foi recebido mais feedback.

Keywords: desenvolvimento ágil de software, comunicação eficaz, padrões

Acknowledgements

I would like to start by thanking my parents, António and Olga, for always being there and going through hell and back just to ensure I was well and that nothing was missing. They are my heroes and make me want to always be a better man than I was before.

Many thanks to my sisters, Brígida and Joana, and my brother-in-law, Válter, for the many happy memories and for helping me appreciate the value in life. To my nieces, for being little bundles of joy who make me smile whenever I see them.

My friends and colleagues from MIEIC, thank you for helping me accept my quirks and pushing me into a journey that made me a more confident person. The good times we shared are something I will cherish forever.

I would also like to thank one of the main entities in my life that led me to work on this topic for my dissertation: BEST Porto. Thank you for helping me come out of my shell, expand my horizons and grow even more as a person. Thank you for all the experiences, the sleepless nights, the challenges, the dinners, the meetings, the teams, and the friends I made there. You guys are another family I have away from my own, and I wouldn't be here without all of your support.

My journey through this dissertation would not have been the same without a few special people. Guedes, Rúben, João, Barbosa, Sérgio, Rafael, Zé, and Matilde, thank you for the moments we shared grieving over the work we were doing. We're all in this together.

I'd like to thank the industry professionals who took the time out of their schedules to participate in the dissertation validation survey. Additionally, many thanks to Elissaveta Gourova, Michael Krisper, Valentino Vranić, Waheedullah Khail and Ömer Uludağ, who provided feedback on my work leading up to and during EuroPLoP 2020. Your friendliness and insights were very valuable.

And finally, I would like to thank the one person who made all of this possible: my supervisor, Professor Ademar Aguiar. Thank you for believing in me and in this topic since day one, and for being patient and understandable when things would not go as entirely planned. I have learned so much thanks to you, and this work would not have been complete without your encouragement and guidance.

Daniel Pinho

“People are deeply nourished
by the process of creating wholeness.”

Christopher Alexander

Contents

List of Figures

List of Tables

Abbreviations

BEST	Board of European Students of Technology
LeSS	Large Scale Scrum
PO	Product Owner
PLoP	Pattern Languages of Programming
SM	Scrum Master
XP	Extreme Programming

Chapter 1

Introduction

1.1 Context

When working in the software engineering world, one can find several paradigms and methodologies to develop software, some being better suited for some situations than others. One of these paradigms is agile software development. The first agile methodologies, based on techniques, concepts and principles borrowed from lean manufacturing, appeared in the early 1990s. As the years went on, these methodologies started to mature and eventually, in 2001, several people representing different methodologies gathered in a ski lodge in Utah to attempt to find commonalities between them. This gathering resulted in the writing of the *Manifesto for Agile Software Development* [5], commonly and henceforth referred to as the Agile Manifesto.

The Agile Manifesto [5] mentions four values that are common in agile methodologies: valuing individuals and interactions, working software, collaboration with customers and being able to respond to change. These concepts are valued over ones that are more common in traditional, plan-driven methodologies: processes and tools, extensive documentation, contract negotiation and following plans. It is important to note that these concepts still have value; the prior set of ideas is simply valued more.

A set of twelve principles, which outline commonalities in agile methodologies, supports these four central values [5]:

1. Continuous delivery of working software;
2. Accept changing requirements;
3. Frequent delivery of working software;
4. Teamwork between developers and business people;
5. Support and trust motivated individuals;
6. Face-to-face communication is the most effective method to convey information;

7. Working software as a measure of progress;
8. Agile processes require a sustainable pace;
9. Technical and design attentiveness enhances agility;
10. Simplicity is valued;
11. The best work comes from self-organizing teams;
12. Teams regularly reflect on their work and attempt to improve.

One can find a particular focus on communication and collaboration among these principles. After all, software projects tend to be extensive undertakings, and sustainable development cannot be a solo effort. This focus can be further seen in the fourth, fifth, sixth and eleventh principles [5], which explicitly approach the interactions between different roles, their environment and team dynamics.

The different roles found in Agile projects include coaches, management/business people, developers, and clients. As one could infer, they take on vastly different responsibilities and tasks during development, and may also have different backgrounds. We can see the importance of high-quality communication between all roles, due to the diversity of people's experiences and past, which can influence their ability to be on the same page as the project goes on.

Agile software development's iterative nature [4][5][10] ends up relating to the concept of feedback loops. These can range from weeks to days, to hours and even to seconds. Most of these feedback loops exist due to the communication between stakeholders, for example, between developers when pair programming or between the development team and the customers when discussing the product and its requirements [42]. These feedback loops are only useful when communication is effective among all parties involved. When this does not happen and communication is sub-par, several unintended consequences may occur, such as misunderstandings, loss of productivity or even, for example, the delivered product not comparing to the clients' vision due to the wrong requirements being collected.

Due to the importance of communication in Agile software development, together with the Manifesto's importance on the team's continuous improvement (seen in the twelfth principle), [5] one should strive to ensure the communication present within Agile contexts is effective.

1.2 Motivation

In addition to his academic studies, during his university path the author also spent time performing activities for a student non-governmental organisation he joined halfway through his degree, the Board of European Students of Technology (BEST).

BEST is a non-representative European student organisation that is present in several faculties and universities across Europe, having as its mission the development of technology students everywhere. This mission is further augmented by BEST's activities, which focus on three main areas: education involvement, career support, and complimentary education [8]. For these activities to be able to come to fruition, in addition to the Local BEST Groups (LBGs) present in each university, BEST also has in its structure a set of nine departments [9][1], dedicated to several areas ranging from corporate relations to marketing, in addition to educational content.

One of these departments of BEST is the Training Department, consisting of several members (tutors and trainers) certified by BEST as being able to provide training sessions to other members. These training sessions are delivered to members a few times per year and can focus either on soft skills, such as leadership, team dynamics, public speaking and communication, among other topics; or hard skills, which can include topics in the areas of IT and marketing, among others. The activities performed by the author during his time in the organisation, along with the training sessions he received, provided a greater of soft skills and their importance in one's personal and professional lives.

During his university path, the author also encountered a set of subjects in which students had to develop software to external entities, emulating real industry work. During the project development present in such subjects, teams were expected to employ agile methods, incorporating concepts such as sprints, project backlogs, and sprint planning and retrospective meetings. The agile methodologies in play during these projects, just like in the industry, required proper communication, which would not always exist due to team members also having other academical responsibilities. This lack of consistent effective communication was a cause of frustration to the author, leading to the subject of this dissertation, as a way to develop tools that could foster effective communication in agile contexts.

1.3 Objectives

The aforementioned goal of "fostering effective communication in agile contexts" can be somewhat vague when described as such. In order to address this issue and provide a higher potential for effective results, the dissertation has four objectives that all contribute to this main goal:

Objective 1: Research effective communication: This first objective stems from the desire to explore the existing literature, as well as agile methodologies' documentation to find common elements regarding communication and how these elements create an environment where communication is more effective.

Objective 2: Foster collaborative environments: Communication quality and effectiveness are influenced by the environments where the organisation people are present. Ensuring that both tangible and intangible aspects of one's environment are compatible with collaborative interactions can help with fostering the effectiveness of one's communication.

[1]Source derived from BEST's intranet (not publicly available).

Objective 3: Empower developers and stakeholders: While individuals are influenced by their environment, they also affect the quality of their own communication. As such, it is important to provide individuals with tools and skills that help them get their point across when communicating and also assimilate any information they need.

Objective 4: Enhance agile communication: The previous three objectives culminate in this fourth one, which aims to implement good practices in agile environments. Communication between developers and stakeholders plays a key role in agile software development, aiding them in being more effective and, in turn, helping them to able to produce higher quality software.

These objectives are intended to be achieved through the construction of a pattern language, where useful skills, roles, activities and environmental concepts are documented in the form of patterns which can be easily applied in relevant situations.

1.4 Structure

The remainder of this document is structured as follows. Chapter 2 presents the background and a literature review of relevant fields, including communication, agile methodologies, the intersection of these two fields, pattern theory, and other related work. Chapter 3 presents the main problem to be addressed, detailing the concepts of the resulting work and the methodology. Chapter 4 presents the pattern language, approaching its structure and including the patterns themselves. Chapter 5 approaches the validation of the pattern language. Chapter 6 draws conclusions from the work done, as well as outlining future work.

Chapter 2

Background

This chapter sheds some light on the background and history of the subjects related to this work: communication, agile methodologies, and the intersection of these two fields. The information on these topics which can be found here results from a literature review of academic papers, books and, in the case of the agile methodologies, their documentation. Since the dissertation's expected result is a pattern language (see section 1.3), the field of pattern theory has also been a subject of research. Finally, to conclude this chapter, some insights on other pattern languages are presented.

2.1 Team communication

Dance [21] attempted to define the concept of communication, but reached the conclusion that this is a concept whose definition can vary wildly depending on the context. He encountered definitions that would range from verbal manifestations, to clarifications and reductions of uncertainty, to exchanges of information, to exertions of power, among other concepts. In this work, we will focus on the concept of communication as the transmission of information between two or more parties.

Effective communication is described as the type of communication where the recipient of the message being conveyed understands its meaning and is able to return that same information when prompted [49]. As such, effectiveness when communicating is of great importance in a field where projects are complex, there is a large number of people involved, a lot of uncertainty and lack of agreement, as happens in software engineering [16].

When the communication in place is not effective, several undesirable effects may happen; out of these, we find time being wasted, mistakes, and misunderstandings [49], which may end up costing more than expected to the organisation. Therefore, it is important to strive for higher effectiveness when communicating to reduce these side effects.

2.1.1 The communication skill-set

In their book, *Interpersonal Communication Skills in the Workplace*, McIntosh et al. [49] approach the topic of communication in the workplace, and discuss the set of skills that are noteworthy and relevant to have in a work environment. Out of those skills, the ones deemed pertinent to the issue at hand were: verbal communication, nonverbal communication, virtual communication, listening, feedback, persuasion, coming up with new ideas, overcoming barriers to communication, and dealing with causes of poor communication.

Verbal communication

Verbal communication is tightly knit with the concept of language; it is through this form of communication that words are used to convey concepts, messages and transmit knowledge [75].

Verbal communication is used simultaneously with nonverbal communication; things such as the tone of voice or cadence transmit additional information that may not be present if, for example, one were reading a transcript of what was said; these elements convey the emotions and the state of the speaker at the time of interaction [49].

Verbal communication raises the importance of semantics and the ideas people transmit in sentences [59], as a clear message is important to ensure that communication is effective.

Metaphors are also used in more informal communication, as they are a part of a person's culture and identity. They can also be used to create additional meaning in what is being said [22].

Finally, Kerbrat-Orecchioni [38] discusses the interaction that happens when two people are engaging in verbal communication; conversations are a back-and-forth between two or more people, and the importance of listening skills comes into play during this.

Nonverbal communication

Knapp et al. [41] state that nonverbal communication is a skill that comes naturally to human beings, as we are a social species. In our daily lives, we send and receive nonverbal messages, which may or may not be intentional.

Examples of nonverbal messages include eye movement, facial expressions, physical posture and vocal behaviour. This type of messages conveys information about one's emotions, intentions and their position regarding the environment they are located in [41].

It is also worth noting that nonverbal messages often coordinate with verbal communication; gestures can be independent from speech (e.g. a thumbs-up), or they may be coordinated with it, complementing, enhancing or moderating what is being said, or even conflicting with the message being relayed in cases of when someone is lying [41].

Virtual communication

Virtual communication is increasingly common with advances in technology, making it possible for team members to communicate in more ways aside from face-to-face communication. Additionally, it allows for communication in distributed teams, which are more and more common [49].

There are several communication methods, which can be divided into two types: synchronous, which allow for real-time back-and-forth interactions, and asynchronous, which may require people to wait to receive a response. McIntosh et al. [49] mention several communication methods: intranets, e-mail, collaborative software, instant messaging, web/video conferencing, and phone conferencing.

Intranets are private web portals that are only accessible to organisation members. In these, one can find information such as a personnel directory, information about the day-to-day operations of the organisation, among other information. Some intranets may also allow for booking of meetings and/or conference rooms and communication with other organization members. Intranets are examples of asynchronous communication [49].

Organisation members often use e-mail to communicate with each other, creating asynchronous messages that the recipients receive in their inboxes. Users can send e-mails to their colleagues to collaborate work, receive memos from the organisation, attach files and Internet links to e-mail messages [49].

Instances of collaborative software provide users with a shared workspace where they can share documents, work in the same documents in real-time, and even discuss with team members. They are mostly an example of asynchronous communication but may have some semi-synchronous elements.

Instant messaging (IM) allows users to communicate with each other using messages that are sent instantly to their recipient, often being shorter in length compared to e-mails. Unlike e-mail inboxes, which have a list of every message thread received by the user, IM services show a view of people the user has interacted with and display their conversations. Despite their near-instant speed to reach their destination, the recipient may not read their messages right away, making this a method with a nature that is both synchronous and asynchronous [49].

Web/video conferencing allows for organisation members to communicate synchronously using a video feed. Depending on the situation, users may also be able to conference while showcasing a presentation; this is useful when the person delivering it cannot be physically at the meeting. This type of conferencing usually requires an Internet connection and a camera. Larger meetings may require a conference room for them to be conducted and, as such, these rooms may require hardware to be installed to make video conferencing possible [49].

Lastly, phone conferencing functions similarly to video conferencing, with the difference that it may not require an Internet connection and only a telephone, and that there is no video [49].

Listening

Listening is a skill that goes hand in hand with verbal communication, as it deals with the information receiving part of a conversation. As McIntosh et al. [49] refer, listening is not only recognizing words but truly focusing on them and making sense of the message being transmitted. This contrasts with the act of hearing, which is an automatic and effortless action performed by the brain as the ears capture sounds and filters out the irrelevant ones [80].

Effective listening brings to the workplace several benefits, such as reducing work that would need to be redone because of a poor understanding of instructions, learning more about the organisation and its operations, increasing the value of the words one says when they speak, and enhancing harmony in the workplace [49].

McIntosh et al. [49] mention four big tips to improve listening and make it more effective: creating the right atmosphere, showing interest, paraphrasing, and asking clarifying questions. Reducing distractions. both external and internal, helps in creating the right atmosphere for communicating. Ways to show interest and listen better to someone include maintaining eye contact, using appropriate facial expressions depending on what is being conveyed, being in a suitable body position, and writing notes for key important information. Paraphrasing and asking clarifying questions drive the point home that one is paying attention and help in ensuring the right information is being delivered.

Feedback

Harms et al. [28] define feedback as "the process of evaluating and discussing the performance of both employees and managers". They declare this is done to raise any improvement points regarding the way the person receiving feedback works, as well as highlighting strong points that are to be maintained [28]. Feedback is supposed to be used as a tool to continuously improve people and help them grow.

Since feedback may deal with a person's flaws, it is important that this feedback is done constructively to help the recipient focus on the act of improvement and not on the existence of their defects [28]. This is not always an easy task, and managers are often poorly trained to do so and are not able to provide effective coaching [44].

In order to make the task of providing feedback easier, several communication models have appeared. Examples of these include the BEAR and BET models, devised by Harms et al., [28] and the BIO model, used by Booking.com employees [56]. The BEAR and BET models are to be used together; the former, which stands for *behaviour*, *effect*, *alternative*, and *result*, is used when addressing an improvement point, while the latter, which stands for *behaviour*, *effect*, and *thank you*, is used to encourage positive situations [28]. Meanwhile, the BIO model stands for *behaviour*, *impact* and *opportunity* [56].

Good feedback allows for continuous improvement in employees and, when used as an encouragement tool, it increases their bond to the organisation [28].

Persuasion

Persuasion is a skill that entails in creating change in the environment by affecting others' beliefs, actions or attitudes. Contrary to what one might expect, as McIntosh et al. [49] say, this is a skill that is not only useful for leaders but throughout the entirety of the organisation, being necessary when cooperation is required.

Since persuasion aims to change people's minds about a certain topic, it requires a solid base on which it can stand on, which is composed of three elements: trust, which comes naturally with a person's trustworthiness; understanding the people being persuaded, so that change can be created more effectively; and building a credible case by checking one's assumptions, having a backup plan in case something goes wrong, and being aware of alternatives [49].

The language used when attempting to persuades also matters; this language focuses on emphasizing the benefits regarding the persuasion subject, directing the speech both to the mind and to the emotions of the recipient, using a positive and definitive approach, and using endorsements from others to create some credibility to one's case [49].

Developing new ideas

In complex projects, it is not uncommon for one to find themself not knowing all information required for a task. In these situations, it is natural to ask for help from someone who might be more knowledgeable. When communicating with others, as McIntosh et al. [49] mention, there are techniques that help in raising new ideas, such as dialogue and brainstorming.

When using dialogue to gather new ideas, the people involved engage in a conversation discussing the available information. All parties should listen to what is being said, respond accordingly, and seek alternatives, with each participant building off of what is being said. A person who is seeking new ideas from colleagues should listen carefully, ask questions, act impersonally, and propose ideas, asking for the other parties' opinions on them [49].

Brainstorming is an alternative better suited to groups, as conducting dialogue with a large number of people does not allow for everyone participating equally in the discussion. The strength this method has is its ability to generate many ideas quickly. People in a brainstorming session are advised to follow four rules, as described by McIntosh et al.: "seek quantity, don't criticize, welcome even far-out ideas, and find ways to combine and improve ideas" [49]. The main theme around brainstorming is to generate ideas using a "quantity over quality" approach and to welcome even the most far-fetched ideas, to avoid alienating participants. In case some participants are not very comfortable with speaking in a group setting, employing Nominal Group Technique (NGT) can be worthwhile: participants write down their ideas and give them to a person who is recording the session. The facilitator reads all ideas, and participants rate each idea, before delivering their ratings to the session recorder. The ideas are then ranked, and then they are discussed. We can see that when using NGT the identities of the authors are not revealed [49].

Another way of generating ideas is through the use of the Six Thinking Hats, a method for discussion facilitation created by Edward de Bono [24]. The titular hats are metaphors for attitudes participants should adopt at that moment in the discussion, and each hat has a colour relating to a specific attitude. The *white hat* calls for an objective and neutral view, which contrasts with the emotion-focused *red hat*. The *black hat* provides a more cautious, pessimistic outlook, while the *yellow hat* is more optimistic. The *green hat* opens the floor to a fertile ground for idea generation, while the *blue hat* focuses on the management of the discussion and the use of the other "hats". Discussions using this method are structured about using each hat in a way where participants can learn the stances and feelings of other people regarding a certain topic, generate new ideas and discuss them.

Overcoming barriers to communication

When two or more people are communicating, they may encounter barriers: elements present in a situation that make it more difficult for people to understand each other. As McIntosh et al. [49] point out, this degree of misunderstanding can range from minor confusion to complete misunderstanding. They also point out eight barriers to communication: framing differences, defensiveness, physical distance, group size and status differences, internal conflict, groupthink, prejudgements, and language issues.

The first one, framing differences, stems from the notion each individual is unique and, as such, has a different worldview from anyone else. In order to overcome this, one should recognise this is happening and try to find common ground [49].

The second one, defensiveness, happens when one is feeling "under attack" during a conversation, and they will end up focusing on defending themselves instead of paying attention to what is being said [49].

The third barrier, physical distance, relates to the concept that the further someone is from another person, they will feel less encouraged to communicate. In order to mitigate this, team members should work in close proximity, or even in an open office [49].

The fourth barrier, group size and status differences, relates to how people feel when they are in a meeting with a large number of people or if there is someone present with a large different status than them; they might feel less confident to intervene in the meeting or even intimidated. To avoid this, meetings have only the necessary people, or any status differences should be addressed so everyone feels at ease [49].

The fifth barrier to communication, internal conflict, can be caused by cutthroat competition between professionals, personal issues, turf warfare between departments or teams, or, related to the first barrier, differences on worldviews. People who are in the middle of conflict should pause, assess the situation, try to gain control of any overwhelming emotions they might be feeling and attempt to listen, which can turn situations like these into productive discussions [49].

The sixth barrier, groupthink, relates to what happens when team members fall into a sense of a hivemind and everyone thinks about things in the same way; this is more prevalent when there is a cohesive group and they all share similar backgrounds and worldviews. This reduces innovation and the team is at risk of being caught by surprise by competitors or by the industry. Solving this requires the team to try to change perspectives every so often and to think outside the box [49].

The seventh barrier, prejudgements, exists when people interpret concepts without listening about them or receiving facts. This can be solved by introducing more facts into the conversation and raising awareness about the reality of the concept [49].

The final barrier to communication, language issues, can appear in three aspects, according to McIntosh et al [49]. These are vagueness and verbosity, jargon, and language differences. Vagueness and verbosity refer to communication that is not precise and that imparts some information but not enough to make it clear, respectively. Jargon is defined by being a set of words members of a profession use but may be incomprehensible by outsiders. Finally, language differences relate to issues that might appear when having non-native speakers, such as trouble in conveying information and misunderstandings. These language issues are subdued by employing a clearer and simpler language to increase accessibility [49].

Mitigating causes of poor communication

McIntosh et al. [49] mention three big causes of poor communication in the workplace. The first one is change; while it is normal and common for things to change in an organisation, having to work with new people implies getting them up to speed in their new environment, as well as build trust, rapport and a business relationship. The second factor is time pressure; tight deadlines and a sense of urgency may make organisation members feel tempted to cut corners, and this applies too to communication. For example, one might overlook details when relaying important information to others, or the message recipients may not ask for details even if they seem necessary. The third cause is interpersonal conflict; people who don't have good relationships with others may engage in practices that may even be counterproductive to the organisation, or not be attentive communicators. While conflicts are common, it is advisable for people engaged in these situations to sort their relationship out to improve productivity and increase the organisation's results [49].

2.1.2 Overcoming team dysfunctions

In addition to these skills, Lencioni [47] discusses the dysfunctions a team may encounter and that may hinder their ability to work productively and deliver good results: absence of trust, fear of conflict, lack of commitment, avoidance of accountability, and inattention to results.

The first dysfunction, absence of trust, relates to the need for a trusting environment so a team can work off of each other. In order for trust to be created among employees, they need to be able to be vulnerable around each other; these vulnerabilities include items such as weaknesses, requiring help in a task, making mistakes, among others. This trust can be built by sharing experiences, showcasing credibility, fostering understanding among team members and following through tasks. A team where its elements trust each other is aware of its weak points and its elements are not afraid to ask each other for help, are more open, and take risks [47].

The second dysfunction, fear of conflict, stems from the stigma present in the professional world to avoid conflict at all costs, particularly as one climbs the corporate ladder [47]. However, doing this can have the opposite effect as one might expect; avoiding conflict may create tension between all parties involved. Lencioni [47] mentions the existence of a good type of conflict, where discussions, instead of focusing on personal attacks, approach only ideas and concepts, ensuring that any present problems are discussed and keeping personal relationships intact. Being able to have this sort of conflict can save time in an organization, as problems are solved more quickly this way.

The third dysfunction, lack of commitment, defines this concept as being the junction between clarity and buying-in. Teams that avoid this dysfunction have clear objectives and everyone commits to them, even if initially opposed. Teams that suffer from this, however, suffer from ambiguity in the environment and lose time, as team members require a perfect amount of information before committing to the goal at hand since they fear failure. A team with committed members is able to move forward more easily, learn from its failures and not be afraid to change directions if it was moving in an undesired way [47].

The fourth dysfunction, avoidance of accountability, relates to the ability of team members to hold their colleagues responsible for the work they do. When accountability is present within a team, team members are fine with having difficult conversations; as such, they establish high standards and are able to identify any problems present within the work environment. If team members avoid accountability, the work quality will be poor, and deadlines may be missed [47].

The fifth and final dysfunction, inattention to results, derives from the thought that team members should focus on the overall progress of the team and not only on their own individual results. Teams that are attentive and pay attention to both types of progress (individual and collective) are able to avoid distractions, enjoy success from their endeavours and benefit from decisions that are done for the sake of the team. On the other hand, inattentive teams are not able to grow as much, cannot beat their competitors, and may lose achievement-oriented team members, as they are much more focused on themselves and not on the team [47].

2.1.3 Effective meetings

The importance of effective meetings arises when workers spend more time than necessary in meetings, which can be translated to wasted time [58]. Avoiding this over-zealousness, workers can better invest their time working towards the project in question.

In their book, Gray et al. [27] propose the 7Ps framework, which aims to improve focus and help the meeting participants to achieve results in a timely fashion. The name comes from the seven terms that constitute the aforementioned framework: *purpose*, which relates to the topic of the meeting and its pertinence; *product*, related to the outcomes and artifacts being produced from the meeting; the attending *people* and their roles; *process*, which is the agenda of the meeting; *pitfalls*, possible risks that may need to be addressed prior to or during the meeting; *prep*, tasks for the attendees to do in advance; and *practical concerns*, which relate to the physical logistics, such as time, date and venue.

2.1.4 Teamwork

Cohen et al. [18] define teams as assemblies of individuals who work in a coordinated fashion as not a group of people, but as a single collective unit. As such, teamwork relates to being able to work as a team.

According to Tuckman [73], a team goes through several phases in its development. In the first one, *forming*, team members get to know each other and discover the boundaries of their tasks and behaviours and show a relationship of dependence with their leaders. This period is also marked by excitement regarding the team and acclimatisation. The second phase, *storming*, is marked by conflict in the team, where team members exhibit defensive and competitive behaviour. This phase is the least productive, as team members may be resisting to the group dynamic and having noncommittal behaviours. The third phase, *norming*, the team overcomes the resistance felt in the previous phase. They become more cohesive, adopt new roles, and feel more at ease with sharing their opinions. The fourth phase, *performing*, is marked by the team seeing its structural problems solved and able to support the tasks at hand; this is aided by more flexible and functional roles. The fifth phase, *adjourning*, was added later on [74] and serves as a termination stage, as the team members conclude their tasks and separate [64].

Katzenbach and Smith [37] define differences between the concepts of teams and groups, as the nature of their work and the situations they are better suited for are not the same. Groups have a leader in a clear, defined role, share their purpose with the broader organisation they are a part of, run efficient meetings and measure their effectiveness through indirect means. Group members' work and accountability are individual, and delegation is prevalent. Contrastingly, teams have shared leadership roles, derive their own purpose, partake in open-ended discussion and try to solve problems during meetings, and measure their effectiveness directly from their work. Team members share accountability and do work together.

The quality of a team's work is related to the goals they have [37], as knowing what to do provides a sense of direction and productivity [48]. Doran [25] outlines a technique to create goals, stating they should be "SMART": specific, measurable, assignable, realistic and time-bound. A goal's specificity relates to how defined it is, requiring measurability to provide an indicator of progress, and assignability to ensure someone will perform the tasks required. Realism is important to keep in mind, as it ensures team members will not be working on a task to which there are not enough resources available. Finally, having a goal be time-bound provides a deadline and some haste to the tasks being done.

2.2 Agile methodologies

This dissertation explores in detail how communication is handled by three methodologies: Scrum, Extreme Programming (XP) and Large Scale Scrum (LeSS). Their structures and history are described in detail in this section.

2.2.1 Scrum

Scrum traces its roots back to the mid-1980s, where Takeuchi and Nonaka [71] presented a then-new methodology for product development. This methodology emphasised speed and flexibility, segmenting and specialising functions instead of using a more traditional approach where the project would move from phase to phase. This methodology was compared throughout their paper to rugby, as the whole team worked together from the beginning to the end of the project, even mentioning the word "scrum", a formation used when the players are all positioned together to gain possession of the ball when restarting play. Several concepts mentioned by Takeuchi and Nonaka in this paper include the usage of self-organising teams, multi-learning, knowledge transfer, and iterative experimentation.

In 1993, Jeff Sutherland, along with his colleagues John Scumniotales and Jeff McKenna, based themselves off of Takeuchi and Nonaka's work and implemented Scrum for the first time at the Easel Corporation [66][67]. Later, in 1995, Sutherland and Ken Schwaber presented a paper [62] at the Object-Oriented Programming, Systems, Languages & Applications Conference '95 (OOPSLA), which detailed Scrum for the first time to the public. This paper mentioned the issues with the waterfall and spiral models and raised the need for a flexible response to unpredictable events. Scrum attempted to provide this flexibility with its iterative approach using sprints, empirical processes where tacit knowledge and trial and error can replace explicit process knowledge when the latter is missing. This early version of Scrum, along with sprints, already used some of the concepts of nowadays, such as having a backlog and doing reviews after sprints.

Scrum was later refined and developed by Schwaber and Sutherland, with them writing the Scrum Guide [69], a document outlining the main concepts, roles, practices and artifacts of the methodology. Scrum has three pillars: transparency, which encourages a common understanding of the system; inspection, which urges users to regularly evaluate themselves and their team; and adaptation, which aims to create flexibility in the face of the changing environment. Schwaber and Sutherland also outline the five values of Scrum: commitment, courage, focus, openness, and respect.

A diagram showcasing Scrum's Sprint, meetings and artifacts can be found in figure 2.1.

The Scrum Team

The Scrum Team, as defined by Schwaber and Sutherland [69], is at the core of the practice of Scrum, being composed of three elements: the Development Team, the Product Owner, and the Scrum Master.

The Development Team is composed of the individuals who, as the name suggests, develop software to deliver new increments at the end of each sprint. They are self-organising, cross-functional and are organised horizontally, sharing accountability and rejecting sub-groups and other titles for these members.

The Product Owner manages the Product Backlog, adding and ordering its items, and has as their goal maximising the value of the product being developed.

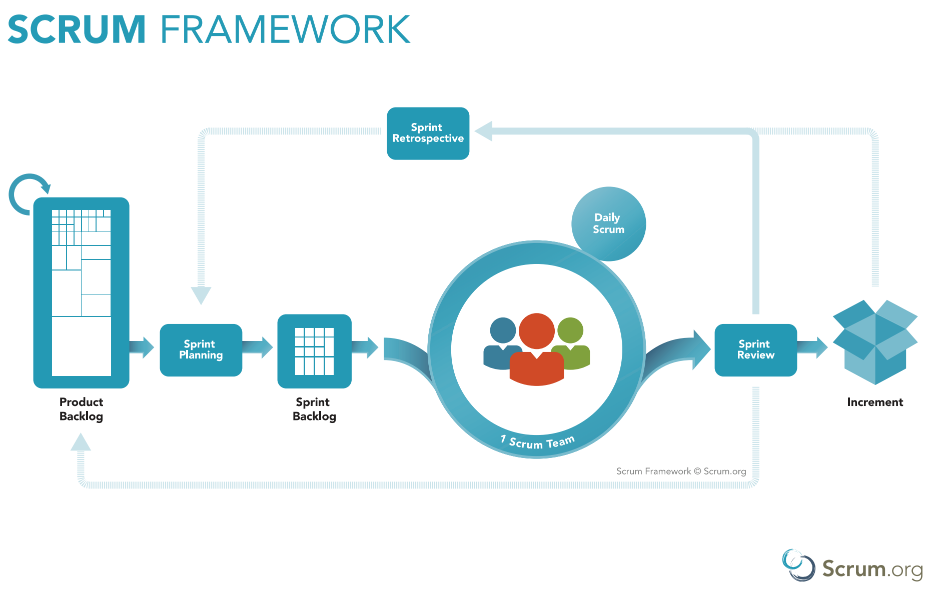


Figure 2.1: The Scrum Framework [63].

Finally, the Scrum Master aids the rest of the Scrum Team with the use of the methodology, aiding the organisation with the adoption of Scrum, coaching the Development Team in being self-organised, cross-functional and able to create high-value products, removing barriers that could hinder the Development Team's work, coaching the Product Owner in managing the Product Backlog effectively, among other tasks.

The Sprint and other events

The iterative aspect of Scrum lies within the Sprint, a uniformly time-boxed duration of time during which an increment of the product is developed. Sprints usually last up to one month; this duration is set by the organization. During this period, the Development Team works on the product, taking care of the items present in the Sprint Backlog. These items represent the current objective, the Sprint Goal, which is established by the Product Owner and is created at the beginning of the Sprint. If the Sprint Goal becomes obsolete, either by the market changing or because the organisation changed its course of action regarding the product, the Sprint can be cancelled, although this event is uncommon. The Sprint is aided by several types of meetings: the Sprint Planning, the Daily Scrum, the Sprint Review, and the Sprint Retrospective [69].

The Sprint Planning meeting, which can last up to eight hours for a one-month Sprint, occurs before the Sprint begins. The Scrum Teams gathers and discusses what can be done during the Sprint and how it will be done. The Product Owner discusses the Sprint Goal with the rest of the team and selects items from the Project Backlog which, after being prioritised, create the Sprint Backlog [69].

During the Sprint, each day starts with a fifteen-minute Daily Scrum meeting. During this event, the Development Team inspects the work done during the previous day and determines which tasks should be done that day. Development Team members also discuss any problems they might be facing to know how to overcome them [69].

At the end of a Sprint, the Sprint Review happens to evaluate the increment developed by the Development Team and, if needed, to adjust the Product Backlog. The Scrum Team discusses what has and has not been done during that Sprint. The status of the marketplace, the Product Backlog, the timeline and the budget are also discussed during this meeting, which can last up to four hours for a one-month Sprint [69].

After the Sprint Review and before the next Sprint Planning, a Sprint Retrospective meeting takes place, where the Scrum Team inspects the quality of the workflow of the team and raises potential improvement points to be worked on during the following Sprints [69].

Scrum Artifacts

Schwaber and Sutherland [69] also point out several artifacts present in this methodology, which have already been mentioned in this section: the Product Backlog, the Sprint Backlog, and the Product Increment.

The Product Backlog exists in the context of the entire project, being a list of everything needed to be implemented across multiple Sprints. It evolves during the project, reacting to the external environment, being the only source of requirements from the stakeholders to the Development Team. As such, it features information related to functions, enhancements and fixes to be done during the project [69].

The Sprint Backlog starts as a subset of the Product Backlog, containing the items that have been chosen to be dealt with during the Sprint. It helps the Development Team keep track of progress and the Sprint Goal, being detailed enough so that the tasks are easily understandable. At the end of the Sprint, the Backlog items that were completed make up the Product Increment [69].

Communication skills handled by Scrum

Scrum approaches verbal communication and listening skills with the meetings defined in the Scrum Guide [69]. The meetings include several discussions about the project, and so these two skills come into play during meetings. Given that these meetings are time-boxed and have clear defined goals, they also fulfil the effective meetings skill.

The feedback skill is approached by one of Scrum's pillars, inspection. This pillar encourages the Scrum Team to continuously evaluate their own work, as well as the work of others, and relay potential improvement points when pertinent [69].

The teamwork skill is embodied by the Scrum Team, as it is composed of professionals with vastly different roles, and as such they must work together in order to achieve progress [69]. The value of respect [69] is also pertinent to teamwork, as team members must respect each other and their work to succeed.

There are multiple ways that Scrum addresses barriers to communication. The pillar of transparency [69] reduces defensiveness, and the value of openness [69] also relates directly to the absence of barriers.

2.2.2 Extreme Programming

In March 1996, Kent Beck joined the Chrysler Comprehensive Compensation (C3) project. It was conceptualised to be a new payroll system for the organisation's employees, and, upon arriving, Beck discovered that the project was going through difficult times and the team was far from ready to have the software go into production, despite that date being two months from then. After meeting with the CIO, the project was restarted with Beck's involvement. The Extreme Programming (XP) methodology was developed during Beck's time with the C3 project, as he took concepts from software engineering and applied them to an extreme degree [4][1]. The resulting method of work included user stories being implemented in three-week iterations, as well as including practices such as pair programming and testing.

XP values

At the core of XP lie five values, which complement each other: communication, simplicity, feedback, courage and respect. Communication relates to the necessity of knowledge transfer between team members and provides enhanced unity among them. Simplicity approaches the necessity to find the best solution for a task, without worrying about doing additional, unnecessary tasks. Feedback helps with continuous improvement in the team, being able to find things that could be better and work on them. Courage relates to being able to act accordingly in every situation, even if it is not pleasant. Finally, respect is necessary among all team members for all values to be effective [4][77].

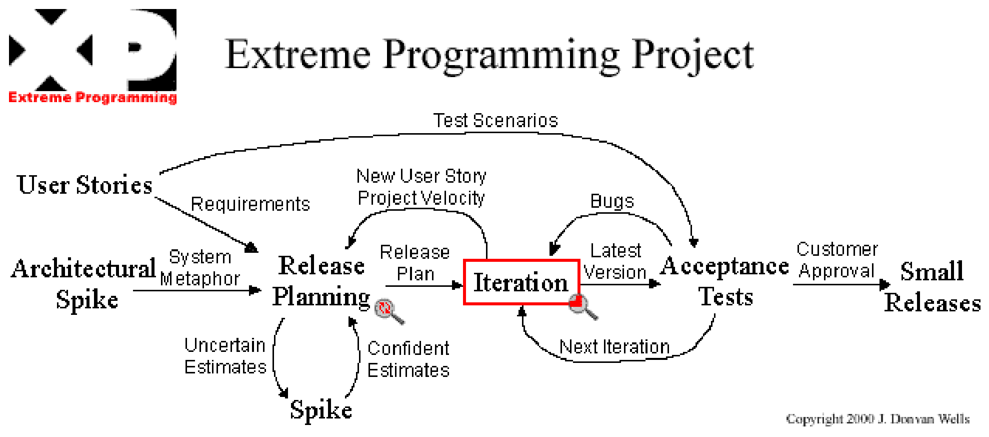


Figure 2.2: The XP methodology [77].

Practices

XP outlines several practices that were first used in the C3 project, such as pair programming, frequent testing, and iterative development, as well as mentioning open space offices and continuous integration.

In pair programming, two programmers sit in front of one computer, with one actively interacting with it and programming (the navigator), while their companion (the driver) thinks about the overall project, provides guidance and reviews the code being developed. This increases shared code ownership, as well as allowing for more inexperienced team members to feel more at ease with the project, learn from their more experienced peers and also helps in reducing the bus factor as not only one person is familiar with the codebase. This is additionally mitigated by the practice of changing pairs every so often, which also helps build trust within the team [77][20][78].

XP also suggests frequent testing when writing code. Testing reduces defects in the software, making it easier to debug, and reduces time spent on bug-fixing, which can be used to further develop the product. XP observes two types of tests: unit tests, which provide insight into the inner workings of the code and how its components interact with each other; and acceptance tests, which are written form the perspective of the users. It is recommended for tests in XP to be automated so that the developers can receive feedback instantly and notice any errors immediately. Testing first before writing code also makes it easier for developers to figure out their approach when developing [4][77].

Similarly to other agile methodologies, XP uses an iterative approach to development, as shown in figure 2.2. Beck [4] suggests working on a weekly cycle to have a good overview of the team's progress over the course of the iteration. At the beginning of each iteration, the team plans their work by choosing user stories, which are ordered by the amount of value to the client they have. Then, these user stories are divided into programming tasks that are digestible by the development team, which provide a detailed view of the work to be done during the iteration. Over the course of the iteration, there are daily stand up meetings, where team members discuss what they accomplished the previous day, what they will do that day, and any hurdles that need to be overcome. This daily stand up meeting ensures the team members are all up to speed regarding the entire project.

A key productivity factor in XP is the physical distance between team members; they should be located close to each other, usually using an open space floor plan. Team members can communicate with each other more easily, as well as making pair programming arrangements easier. The physical space also should common areas for stand up meetings, as well as including room on the walls for sketches and notes.

Another topic approached by Beck [4] is continuous deployment: code should be integrated and submitted into the repository frequently. Frequent integrations take less time than infrequent ones, compatibility issues are more easily addressed, and the product is always deployable if required. Beck also notes that the build, which ideally should last about ten minutes, should be performed synchronously by the system instead of having to trigger that event manually [77].

Communication skills handled by XP

In XP, pair programming addresses multiple skills: verbal communication and listening are required so that the pair can be truly working together, coordinating their efforts at the task at hand. It also helps with feedback, as the navigator is able to provide a live code review of the driver's work. When having issues with a certain task, pairs are encouraged to take a step back and think of new ideas to solve it, particularly if both already have different suggestions that the other one isn't accepting. The rotation of pairs fosters teamwork amongst the developers and makes them trust each other more; this latter point relates to the dysfunction of absences of trust [4][77].

XP has other components of its identity that relate to communication skills. Its meetings, such as the Daily Standup, also have goals in mind just like Scrum [77], which make them more effective, as well as requiring more verbal communication from the team. Two of XP's values, feedback and courage [4], also relate to skills: the former is a direct correspondence with the homonymous skill, and the latter attempts to solve the fear of conflict dysfunction. Another dysfunction XP addresses is the lack of accountability, mitigated by supporting shared code ownership [77]. Finally, XP encourages shared, open offices [4], which remove physical distance, a barrier to communication.

2.2.3 Large Scale Scrum

Large Scale Scrum (LeSS) traces its origins to 2005, being developed by Craig Larman and Bas Vodde. The industry panorama at the turn of the century was that Agile methodologies were appropriate for small teams. However, after Larman and Vodde received requests to apply Scrum at a larger scale, they decided to develop a framework that was able to be applied to large, distributed teams and still be Agile [45]. This resulted in the creation of a methodology that was derived from Scrum and applied its basic principles while being functional and aware of the limitations faced by distributed teams.

LeSS has two variants: LeSS, which can be applied to eight teams of eight people; and LeSS Huge, which can be applied to thousands of people. They function very similarly to each other, however, in Less Huge the scope of the project is divided into Requirement Areas, each with an Area Product Owner, who works with the Product Owner [12].

Since LeSS reimplements Scrum's principles at a larger scale, there are a few constants from Scrum that are also present in LeSS: there is only one shippable product being developed which is done through the management of one single Product Backlog. There is one Product Owner in charge of that artifact, and all teams are synchronised to the same Sprint. This ensures all development is done towards the increment, and that things are ready at the end of a Sprint. Backlogs, designing and planning the work for the coming Sprint [45][12].

A diagram showcasing the main composition of a Sprint in LeSS is shown in figure 2.3.

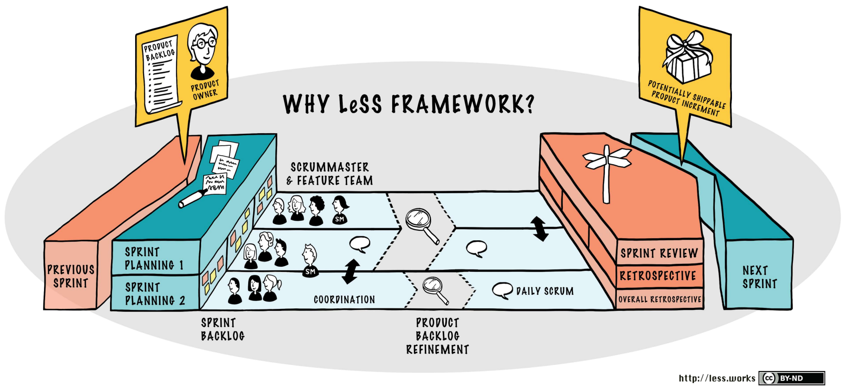


Figure 2.3: An overview of the Sprint in LeSS [12].

Principles

LeSS defines ten principles at the core of its workings: *LeSS is Scrum*, which relates to the application of Scrum's principles at a larger scale; *Transparency*, which, among other practices, urges people to work together, create trust within the team, and creating a better environment where everyone can work fearlessly; *More with LeSS*, which stands by the ideas of more meaningful work and team ownership of the process, without creating more roles and processes that could diminish the quality of the work done; *Whole product focus*, which relates to the fact that there is only one Product, one Product Owner and one Sprint, and everyone is focused towards that same goal; *Customer-centric*, where we see a focus on the client's perspective in order for the development team to understand what is really required and what can be learned from the client's reality; *Continuous improvement*, which urges the team to continuously inspect and evaluate their work to improve it and always seek perfection; *Lean thinking*, which tries to avoid creating unnecessary bloat by streamlining things such as communication practices; *Systems thinking*, which encourages people to inspect the system as a whole and see how the product in its entirety acts, instead of focusing on minute details; *Empirical process control*, which relates to learning from inspecting previous work, including things that did and did not work, instead of following a pre-determined set of practices that may not be appropriate in the context; and *Queueing theory*, which encourages people to understand how queues work in order to work more efficiently. [45][12].

Meetings in LeSS

The meetings in LeSS are mostly the same as in Scrum: at the start of each Sprint, there is a Sprint Planning. However, as shown in figure 2.4, this meeting is divided into two meetings (Sprint Planning 1 and Sprint Planning 2) to avoid having meetings with a large number of people. In Sprint Planning 1, the Product Owner meets with representatives from each Development Team, as well as Scrum Masters. In this meeting, the Product Owner lays out the Product Backlog items ordered by their priority, and the team representatives distribute items amongst themselves. The team representatives also seize this chance to clear up any doubts they might have. Afterwards,

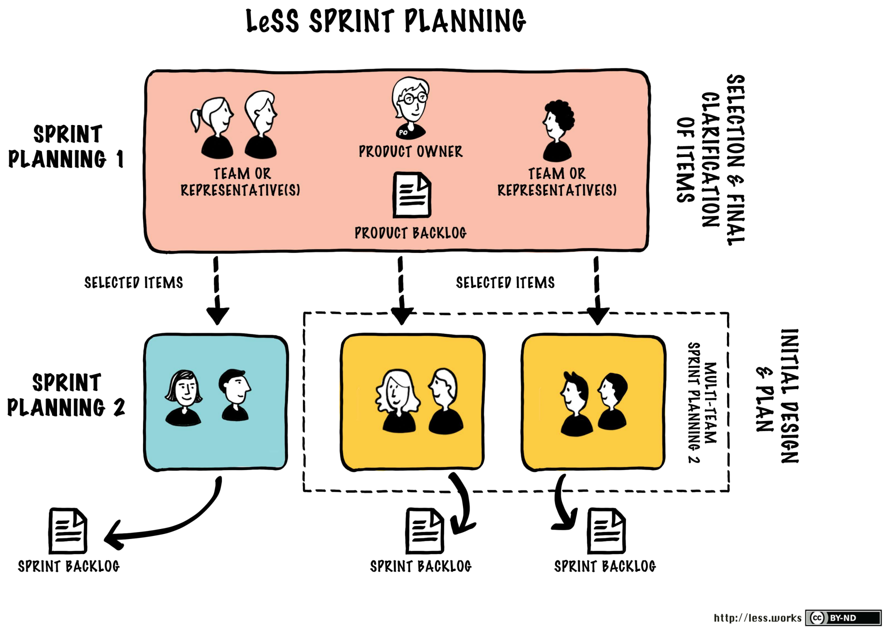


Figure 2.4: Sprint Planning in LeSS [12].

teams divide and each one of them has a Sprint Planning 2 meeting. Each team usually has its own meeting, but two or more teams may decide to have this meeting together if the items on their Sprint Backlog are closely related. In this meeting, the teams take their Product Backlog items, discuss them and create their Sprint Backlogs, designing and planning the work for the coming Sprint [45][12].

Just like in Scrum, Daily Scrums are also present in LeSS. Every day, during this time-boxed meeting of up to fifteen minutes, the team members discuss what they did the previous day, say what they will work on that day and relay any issues to the team. Members from other Development Teams may also join other teams' Daily Scrums as silent observers, to facilitate coordination between teams [45][12].

At the end of a Sprint, as shown in figure 2.5, the Sprint Review happens. During this meeting, the teams are joined by the Product Owner and the stakeholders and/or users in a Sprint Review Bazaar, akin to a science fair. The room is divided into different areas where everyone can discuss the teams' work. Afterwards, all parties involved gather to further discuss and provide feedback and decide how to move forward in the next Sprint [12].

The Sprint Review is followed by the Retrospective meetings. Each team has its own Team Retrospective, which is then followed by an Overall Retrospective. Team Retrospective meetings function like Sprint Retrospectives in Scrum; teams inspect their work and discuss its quality, as well as raising potential improvement points for the organization. These meetings are followed by the Overall Retrospective meeting, which includes the Product Owner, the Scrum Masters, and representatives from each team. In this meeting, the participants explore the way things are

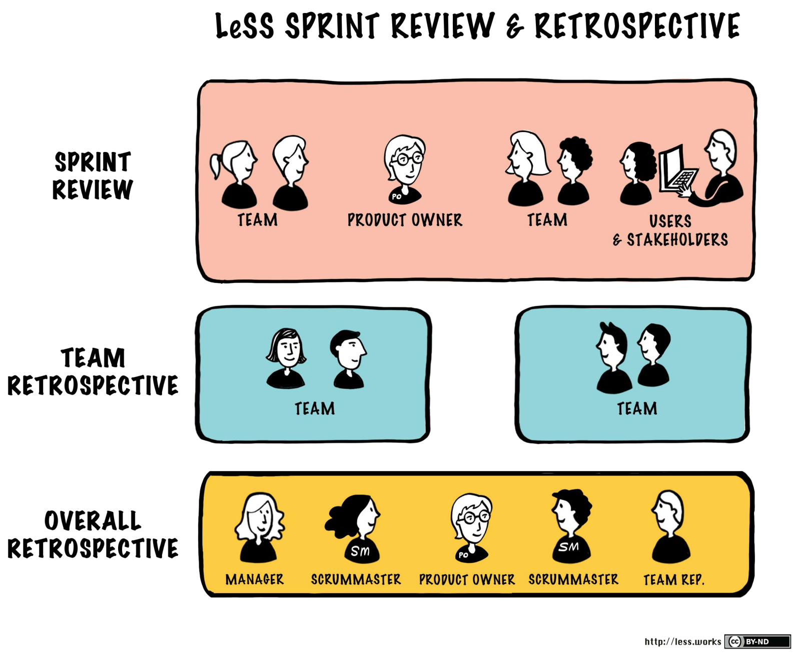


Figure 2.5: Review and Retrospective meetings in LeSS [12].

working in the organisation, discussing topics at a scope superior to a single team's [12].

Communication skills handled by LeSS

LeSS, a scaled approach to agile development, is often more explicit with the communication skills it employs compared to Scrum and XP. Just like in the other methodologies, meetings are time-boxed and have specific goals [45], which helps in their effectiveness. In these events, listening has some additional value, as members from other development teams may join another team's Daily Scrum as a silent observer, which this allows for larger dissemination of information. Meetings also allow team members and stakeholders to receive feedback, such as the Sprint Review meeting.

LeSS encourages informal, verbal communication, as it is deemed to be the fastest way for knowledge transfer. However, in order to keep everyone updated with the teams' progress, the idea of communicating in code, through commits to the repository, is noteworthy; this is a form of virtual communication.

Teams in LeSS are self-organised, cross-functional and long-lived; this requires them to practice teamwork among their members; this also increases trust as team membership is stable over time, which mitigates the absence of trust dysfunction. Teams are co-located, which reduces the distance between members and, in turn, helps in overcoming a barrier to dysfunction. Conflict is also important in teams; if teams learn how to deal with these situations, they will learn from the experience, as well as being better prepared for the following times. This helps with a cause of poor communication, interpersonal conflict, and also reduces the dysfunction of fear of conflict any team members may have.

Given that teams in LeSS are cross-functional, they use Communities of Practice to foster continuous improvement and knowledge sharing regarding different areas of operation in the organization. Members of a community are often committed to their participation and encouraged to review each other's works. This deals with the dysfunctions of lack of commitment and inattention to results.

Organisational structure

Teams in LeSS, similarly to Scrum, are self-organising and cross-functional, being able to manage their work and responsibilities on their own [69][45]. Their cross-functionality allows them to work across the entire technological stack, and focus on a single product feature, taking on a customer-centric approach to product development. In addition to this, team members are exclusive to one team, they work together in the same space and they tend to stay with the same team for a long time. This fosters trust and stability within the team. Team members are also expected to have conflict amongst themselves and to be able to solve that said conflict, as they are able to learn from those experiences [12].

LeSS also includes the concept of Communities of Practice (CoP), informal groups of people who gather voluntarily because they are interested in a topic, have experience or want to learn more about it. CoPs have an informal leader (the CoP coordinator) who helps with managing activities. CoPs allow for the exchange of information between teams, as members share their knowledge about the topic of the CoP with each other [45][12].

The Scrum Masters help the organisation learn about Scrum, providing continuous coaching regarding adoption. They support the Product Owner by teaching them about their role, the Product Backlog and the relationship they have with teams, customers and users. The Scrum Masters also help the teams; each Scrum Master may take upon assisting up to three teams at a time, encouraging them to be self-organising and sharing responsibility. The Scrum Masters' focus on the Product Owner and the teams decreases over time as their focus on the organisation and development practices increases, as the organisation needs to stay sustainable for the use of LeSS, and also keep the teams aware of modern development practices [12].

2.3 Communication within agile contexts

As described in the previous section, the three agile methodologies examined already mention some things regarding the communication skills explored in section 2.1. From this, we can see that most skills were approached by all three methodologies: verbal communication, feedback, listening, effective meetings, and teamwork. The topics of removing barriers to communication and overcoming dysfunctions were also approached by all methodologies to varying degrees. It is expected that teamwork and effective meetings would be approached, as they are required for the basic functions of each methodology.

As the Agile Manifesto mentions the importance of communication and collaboration, one can expect the scientific community has approached the conjunction of communication and agile software development. As such, a literature search for publications that combined these two fields was performed. The resulting publications encountered approach several recurring topics, including meetings, practices, environment and organisation.

2.3.1 Meetings

The importance of meetings was raised several times. Starting with iteration planning [69][4][45], where the development teams plan their next iteration along with the project stakeholders, the opportunity to collect the project's systems requirements is important, as it gives the team openings to clear up any doubts they might have and reduces the chance for miscommunication and the iteration resulting in an increment that didn't follow the client's vision [13][45][57]. In daily meetings, team members are able to provide feedback and keep everyone updated on project status, providing a view of the overall picture and how the team is progressing [4][13][45][57][69]. At the end of an iteration, review and retrospective meetings allow identification of strengths and improvement points, and this sharing of feedback allows the team to continuously develop itself [13][45][57][69]. Retrospective meetings, in particular, benefit from a distance to the workspace to help team members be in a good mindset to focus on the meeting and not on their pending tasks [39]. Kraut [43] also suggests that meetings are effective for routine communication, as it provides a controlled way for the stakeholders to interact with the team.

2.3.2 Engineering Practices

The literature also mentions several practices that are useful for enhancing communication within these environments. With pair programming, developers sit next to each other while one interacts with the workstation and the other one observes, provides feedback and assists their colleague. This can be highly useful for real-time code reviews and sharing of tacit knowledge [57][77][78][13][53]. Murru et al. [53] also mentioned that it increased trust among team members, as the informal communication that happened during pair programming allowed pairs to feel more at ease with each other. However, a case study by Pikkarainen et al. [57] found that while it was useful for code reviews, its usefulness was not as expected when writing code. Another practice found often was the importance of face-to-face communication; this allows team members to discuss sudden events, clear up any uncertainties they might have and share knowledge [13][70]. Melnik and Maurer [50] note that direct communication should be favoured over long communication chains to reduce communication errors. Along with these benefits, Mishra et al. [52] also discuss the importance of face-to-face communication has when it comes to the conveying of emotion, visual cues and other nonverbal communication signals. The practice of continuous integration was also approached in the literature; Pikkarainen et al. [57] mention it facilitates communication about the current project status between the developers and tester groups, and Larman et al. [45] state it allows inter-team communication through commit messages to the repository. Continuous integration also leads to being able to quickly show the client the project status, avoiding further misunderstandings and providing a tighter feedback loop [13].

2.3.3 Environment

Practices related to the environment the teams are set in also influence communication. Information radiators provide information about the project and task status and are best located in easily accessible locations, such as hallways [16][57][77][52]. Taibi et al. [70] also mention that the presence of information radiators reduces the communication overhead required by the team, freeing more time for the development team to dedicate to other activities. Along with information radiators, any written information, such as logs, documentation, and user stories, should be easily accessible; this is often done using web-based collaborative spaces [13][46]. The layout of the space in which the team works is also a key factor regarding its quality of communication; an open space layout reduces the distance between team members and their communication cost [16]. It also facilitates informal communication and fosters trust within the team, in addition to having the potential to make the environment more suitable for pair programming [53][57][77]. Pikkarainen et al. [57] also raise a few problems regarding an open space plan: the room can easily become noisy and distracting, which hinders productivity within the team. This problem was explored by Murru et al., [53] who suggested the use of half-height glass walls to increase focus and privacy without having to sacrifice the small distance between team members. The presence of customers may influence the environment since they are a source of feedback and have a better knowledge regarding the end result. Their presence and availability help the development team in their work, and a lack of involvement ends up being a challenge the team has to solve [33].

2.3.4 Organisation

The final topic at hand relates to the organisation. Conway [19] mentions that the design of a system will mirror that of the communication patterns within the organisation. As such, there is a need for increased organisational flexibility so that different system modules interface in an orderly way with each other. A horizontal or low-depth structure has been explored in order to reduce complexity [53][45], which also connects to the idea of cross-functional teams. These teams allow for some independence and are able to target a specific feature, but are also able to talk to other teams to share knowledge [13][12]. A useful way to perform this knowledge sharing is through the usage of communities of practice, which are informal, voluntary and self-organised groups that exist parallel to the organisation chart and generally consist of organisation members interested or with expertise in a particular topic [12][11][36]. A low-depth organisational structure, along with the iterative nature of agile development, may cause clashes within different parts of a company [57][45][53]. As such, agile practices should be introduced in a careful and gradual way to minimise friction between developers and business people; this introduction can be aided if people are working at a sustainable pace, get executive support, negotiate and set expectations, and have a coach helping them [61]. The organisation's management plays a vital role in agile software development. Highsmith [31] states that it is important that managers are fast and able to deal with change in a short amount of time, such as through using self-organised teams and a collaborative leadership style instead of a command-and-control one. He also states that managers should not attempt to use silver-bullet solutions, as each situation is different.

2.3.5 Individuals and teamwork

In his book *Agile Software Development: The Cooperative Game*, Alistair Cockburn [16] explores the interactions between individuals and how they work in an agile setting. He states that software development is a cooperative endeavour, where team members have to work together and collaborate to achieve good results. Cockburn raises the individuality of organisation members, as each person is different, has their strong and weak suits, and interacts with their environment differently. He also approaches the topics of how someone is motivated, whether it is due to pride in their craft, their successes or their contributions. As people do not work alone, Cockburn explores the nature of teams and how they communicate. This communication is influenced by the environment, such as walls, distance, and the presence of information radiators. Communication has modalities that add additional data to the verbal content, such as sound, gestures, and body language. Cockburn also talks about teamwork and how team members should get along with each other.

2.4 Pattern Theory

The concept of patterns in use today in the software engineering field traces its roots to architecture and the works of Christopher Alexander, such as *The Timeless Way of Building* and *A Pattern Language* [3][2]. In these books, Alexander presents information in a structured way that is supposed to be simple to understand and apply to recurrent situations [68].

This approach to problem-solving was then adapted into the software field, first by Kent Beck and Ward Cunningham in 1987 [6] and subsequently popularised by the 1994 book by Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software* [26], and the authors became known as the Gang of Four (GoF).

To support the development and documentation of patterns, the Hillside Group was founded in 1993, when Kent Beck and Grady Booch sponsored a gathering in Colorado where they, along with Ward Cunningham, Ken Auer, Ralph Johnson, Jim Coplien, and Hal Hildebrand, discussed the creation of patterns according to Alexander's ideas. The Hillside Group organises several conferences focused on writing patterns, workshops and invited talks, such as PLoP (Pattern Languages of Programming), EuroPLoP, and ScrumPLoP [72].

Anatomy of a pattern

A pattern is structured in a way that is supposed to convey the necessary information in the fastest way possible. As described by Meszaros and Doble [51], patterns have mandatory components: a meaningful name for easy recollection, a problem to be solved, a solution that solves the problem, a context in which the subject is inserted and/or the problem appeared, and forces which are present and may influence the choice of the decision.

Patterns may also include optional components, such as indications that the problem might exist, a resulting context from the application of the pattern in question, a rationale that explains the suitability of the pattern in that context, related patterns, illustrative examples and code samples. Patterns may also be known for other names, also called aliases, and also include acknowledgements to contributors to the pattern [51].

In addition to this structure, patterns have thumbnails, usually addressed to in footnotes and in lists of patterns, which relay in a small paragraph what the pattern is about. These pattern thumbnails may also be included in external works if pertinent to provide context to a pattern that is not in the same work [51].

A cohesive set of connected patterns creates a pattern language, which tends to be more than the sum of the patterns within. Pattern languages tend to approach highly complex situations and/or problems that cannot be solved with just one pattern, requiring the use of smaller, more applicable pieces of information [51]. Patterns in a pattern language abide by rules about how they connect to each other and sequences of how they are applied [20].

Takashi Iba and Taishi Isaku [35] outline a process to create pattern languages based on three phases: pattern mining, where information is found from one's own experiences or by collecting it from others; pattern writing, where the patterns are documented, based on the knowledge collected beforehand; and pattern symbolising, where one ensures the name is representative of the pattern, along with having a visual element that relates to the text.

2.5 Other pattern languages

The author also examined other works containing pattern languages that approach similar areas to the topic at hand. This research allowed him to be more familiar with patterns in general and find new relevant information.

Organisational Patterns of Agile Software Development, written by Jim Coplien and Neil Harrison [20], explores several concepts that are present in agile organisations. These concepts range from the roles people play in the daily activities, the ways they should distribute work, how information should flow, the ways teams should function, and how the environment should be like, among other topics.

A Scrum Book: The Spirit of the Game, written by Jeff Sutherland et al. [68], documents several of Scrum's concepts under the form of patterns. In addition to these, the book's pattern languages explore related topics, dealing with topics including teamwork, the organisation's environment, and useful practices and behaviours. It is important to note that the patterns in this book are referred to in this dissertation as the *Scrum Patterns*.

A Pattern Language for Inter-Team Knowledge Sharing in Agile Software Development, written by Santos et al. [60], approaches the ways organisations can foster knowledge sharing between different teams. These ways can affect the physical environment, the developers' day-to-day activities, or even the structure of the teams.

Chapter 3

Proven Communication Practices for Team Communication

This chapter showcases the problem at hand and its solution, along with the methodology being employed during this dissertation.

3.1 Problem

As mentioned in the previous chapters (namely sections 1.1 and 2.1), the presence of effective communication is crucial in agile environments, as its absence often results in time being wasted, mistakes, and misunderstandings [49]. As such, one can state the problem at hand as the question "How can one communicate effectively in agile environments?", as answering it would achieve this work's objectives.

3.2 Planned contribution

As mentioned in chapter 1, this attempt at fostering effective communication within agile contexts is fulfilled by creating a pattern language. As stated in section 2.4, a pattern language is a cohesive set of interlinked common and proven solutions to recurrent problems. The literature review (see section 2.3) done provided with a few recurring practices that aid in enhancing collaboration and teamwork within agile teams and their stakeholders.

The pattern language (see chapter 4) approaches concepts from several areas of interest regarding the presence of an individual working on an agile project, such as the environment, the organisation, the individual itself, and their colleagues.

3.3 Methodology

The process used to create the pattern language was based on the works of Takashi Iba and Taichi Isaku [35], outlining three main phases: pattern mining, pattern writing and pattern symbolising.

The first phase, *pattern mining*, relates to the pursuit of knowledge and information that will be later used in the second phase where the patterns themselves are written [34]. There are several sources of information and approaches to obtaining it, such as deriving knowledge from one's own experiences, collecting it from others or even brainstorming. For this dissertation, the knowledge collected for the patterns was acquired through a literature review of relevant materials, such as academic papers, books and web pages, as well as deriving some information from the author's own experiences.

The second phase, *pattern writing*, relates to the structuring of the pattern language, along with writing the patterns themselves. A pattern has several sections, including a context, a problem, a solution and a set of forces that influence said solution [3], and this process entails in writing these sections using the information gathered in the previous phase [76].

The third phase, *pattern symbolising*, relates to the gist of the pattern being easily understood. As such, it is important to ensure the pattern's name is clear and gets the point across, as well as adding a visual element, such as a photo or an illustration. For this dissertation, each pattern has an illustration drawn by the author that represents (one of) the main ideas of the pattern's solution.

Finally, since interviews were not able to be conducted in the mining phase, the work's validation stage (see chapter 5) entails in surveying agile coaches and similarly experienced professionals to ensure that the patterns ring true to their experiences.

Chapter 4

The AgilECo Pattern Language

The AgilECo (**Agil**e **E**ffective **Co**mmunication, pronounced "agile echo") pattern language contains a set of interconnected patterns that provide information regarding environmental factors, activities and (inter-)personal attributes in order to help increase the effectiveness of the communication found in agile environments. This increase stems from the reduction of existing problems and side effects, while also attempting to enhance knowledge transfer interactions in the organisation.

These patterns are structured using Christopher Alexander's form of *name-context-problem-solution* [3]. Accompanying this structure is a set of forces, which are facts that influence the problem and the resulting solution.

An overview of the pattern language (see figure 4.1) showcases how the patterns are separated into three main areas: environment patterns, activity patterns, and people patterns:

Environment patterns: These patterns (section 4.1) approach both physical and digital factors that exist in the settings where organisation members interact and perform their activities, as they end up influencing the people's behaviour. *Physical environment* patterns describe tangible elements that attempt to remove barriers and reduce the face-to-face communication cost, enhancing communication in these settings. Similarly, *digital environment* patterns describe tools and other software which help to improve communication, as the digital world influences how individuals interact with each other.

Activity patterns: These patterns (section 4.2) describe the activities people engage in during their day-to-day functions and are divided into meetings and practices. *Meeting* patterns describe recurring events where people gather and discuss certain topics with a specific goal in mind. *Practice* patterns relate to actions people take during their daily activities, with various forms and objectives.

People patterns: These patterns (section 4.3) relate to people and the way they behave as individuals in the organisation, as they have some sort of role and a certain skill-set that helps in

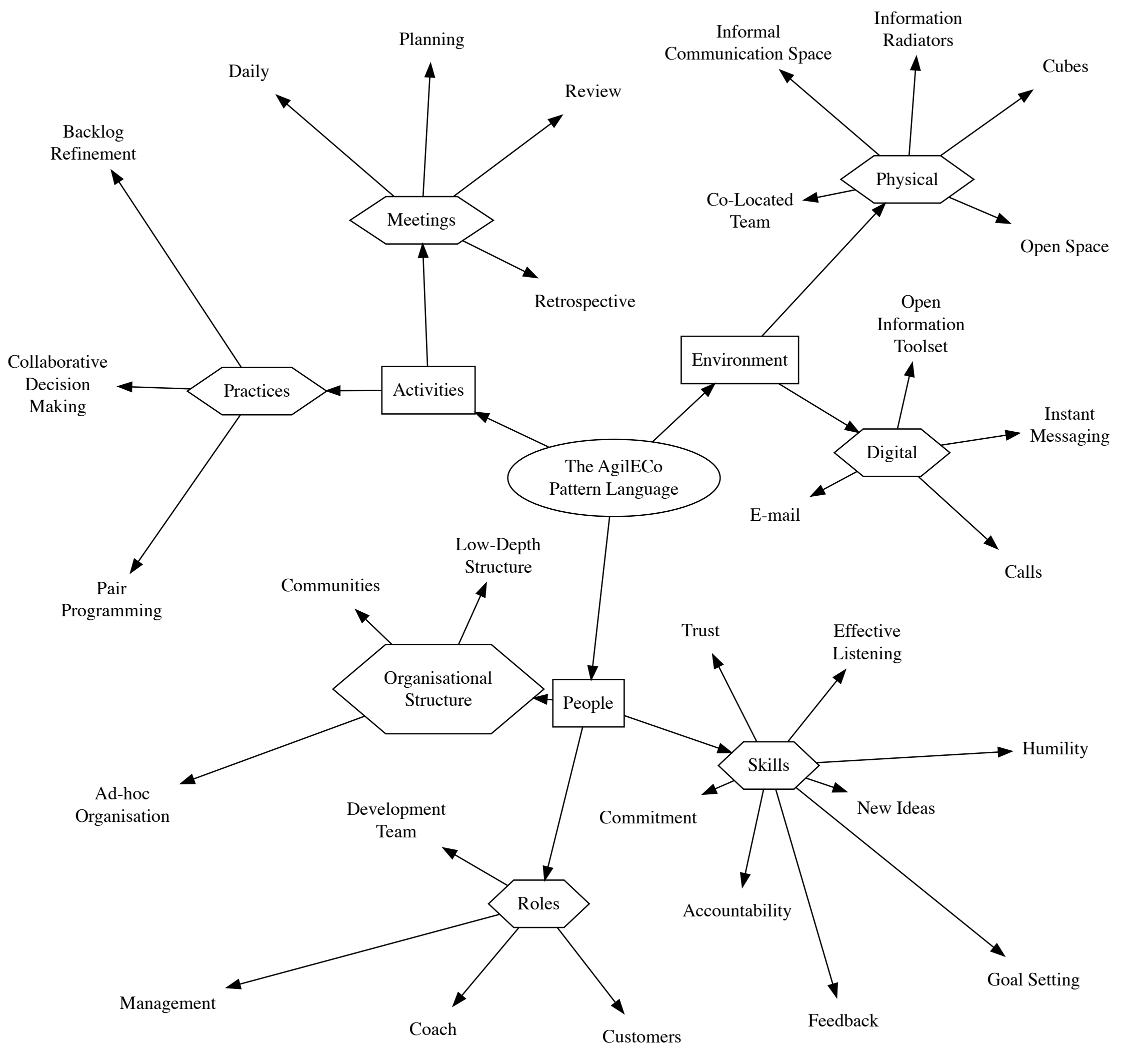


Figure 4.1: An overview of the AgilECo pattern language.

communicating effectively. The organisation itself also has a structure which may influence communication quality. *Role* patterns describe the roles a person can have in the organisation, their responsibilities and their interactions with others. *Organisational structure* patterns approach the way organisation members interact with each other, as these interactions are influenced by the way the organisation is structured. Finally, *skill* patterns describe useful abilities that are useful when communicating; these skills may be related to their behaviour or their interactions with others.

4.1 Environment patterns

Pattern: INFORMATION RADIATORS

In an agile development setting, team members are often sitting in close proximity to each other, sharing an OPEN SPACE office or simply being a CO-LOCATED TEAM. In these environments, the team members are working on the same project, towards the same goal.

Problem

A team cannot achieve its goal without some sort of coordination between its members. The usage of meetings during each iteration, particularly DAILY MEETINGS, can improve the coordination existent between the team members. However, this information is only provided once a day, being static, and it relies on the team members' memory, instead of being available for consultation and being updated as progress occurs.

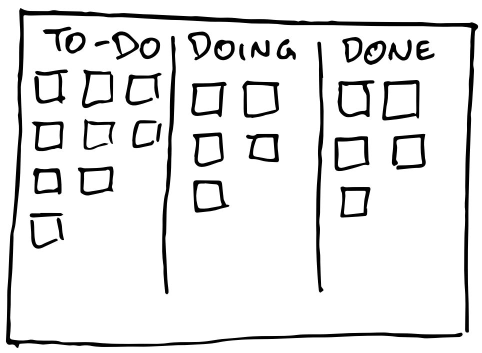
How can we ensure everyone is up to speed regarding the current status of the project?

Forces

- **Effort.** Team members should not have to spend much of their energy, both physical and mental, to stay up to date on the project status.
- **Accessibility.** The information being conveyed to the team members should be easily accessible to them.
- **Information stickiness.** Having things written down ensures they are not lost; Team members should have the option to consult the information available to them if necessary [16].

Solution

Include visual aids, such as whiteboards, flipcharts and similar items with useful information in common areas or in places where people frequently walk by.



These visual aids are often displayed on walls of hallways and shared offices, ensuring accessibility by most, if not all, of the DEVELOPMENT TEAM and the project stakeholders [16].

Depending on the people that use it, the size of an information radiator may vary: two or three people can use a sheet of paper while a whole team should use something bigger. However, one should strive to aim for a bigger size as it is more visible and thus easier to take in information [16].

Another important thing is that the information displayed on an information radiator changes over time; this can be used to track the current progress during an iteration, look up task assignments or even check on the status of auxiliary services [16].

This pattern is related to Kanban, a system that was developed by Taiichi Ohno, who was an engineer in Toyota [55]. This system uses cards are moved from one place to another as the manufacturing process goes on.

Related patterns

The usage of information radiators to keep track of the progress made so far during an iteration is useful when it comes to GOAL SETTING. They can also provide the outcomes of RETROSPECTIVE MEETINGS, so that team members know what to keep in mind to improve their performance [16].

* * *

Pattern: CO-LOCATED TEAM

In agile software development products, the members of the DEVELOPMENT TEAM interact and communicate regularly with each other to ensure they are properly coordinating their efforts.

Problem

The communication between team members, which is inherent to the way teams work, should be easy to achieve. However, people may tend to not talk to each other if they find resistance or if they find that it takes too much effort.

How can we ensure that communication is nearly effortless within the team?

Forces

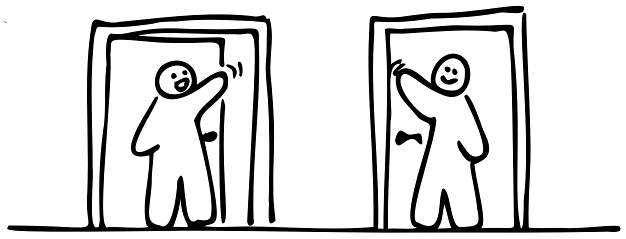
- **Communication cost.** The distance between team members ends up affecting the cost of communication between them. A team member that needs to ask something to a coworker who is working on another floor, they will spend time and energy going to meet their colleague. If the coworker happens to be away from their office, all this effort will have been in vain, and the team member will not be as motivated to try again [16].

- **Conway's law.** Melvin Conway [19] states that a design's structure will be a reflection of the organisation's communication structure. As such, if an organisation wants to have a good, modular design, the interactions between team members should be able to happen accordingly [30].

- **Effectiveness of informal communication.** Informal communication is significant in agile projects to the point that one may not notice that it is there, but its absence makes that importance apparent. Team members being able to communicate face-to-face informally during the day allows them to build trust with each other, discuss any unplanned issues that may appear and absorb tacit knowledge [54] from each other, improving the overall familiarity with the project.

Solution

The team members should work in close proximity to each other, in the same room, adjacent rooms or at least on the same floor.



Out of all these cases, being in the same room is the most desirable option, as it is the option with the lowest distance between team members. However, if such is not possible, working in adjacent rooms or on the same floor can be a worthwhile alternative [16].

Having team members work in close proximity allows them to clear up with the relevant person any issues that may be on their mind in person, build their working relationships and to develop higher-quality code.

The proximity also reduces any potential frustration and subsequent loss of motivation due to missed encounters, as there is a high probability that the people in question will run into each other frequently [16].

This pattern is a stark contrast to having distributed teams, in which team members work separated across different regions.

Related patterns

The team may evolve their proximity regarding working in the same room, developing an OPEN SPACE layout to their offices. Having everyone work in the same space also allows for EXPERT IN EARSHOT [15], making it easier for people to learn by watching and listening to others.

* * *

Pattern: OPEN SPACE

Agile teams work as cohesive units, making use of their highly frequent communication and teamwork to deliver working software in small intervals of time (i.e. iterations).

Problem

The physical space where team members work is of high importance to the resulting work, as the latter is conditioned by the cooperation between team members. This cooperation should be as smooth as possible with the least amount of friction; if the workplace does not foster a collaborative atmosphere, the team will face problems with reaching its goal.

How should a CO-LOCATED TEAM *organise its space to reduce friction and entropy?*

Forces

- **Distance between members.** The motivation team members have to talk to each other is directly related to the distance between them and the energy (both physical and mental) spent to perform that interaction.
- **Feeling at ease.** Team members are bound to engage in conversation with their colleagues regarding work and other subjects depending on how comfortable and familiar they are with each other and the amount of trust within the team.
- **Face-to-face communication.** Face-to-face communication is an effective communication mechanism that comes naturally to human beings. It is able to convey a lot more than just words, showcasing feelings and other sub-textual information in a quick interaction.

Solution

The team should organise their furniture to remove physical barriers between team members, including everyone in the working atmosphere.

This requires that the team is working in the same room [16], and the furniture layout should not exclude anyone, allowing people to see each other and to help team members feel more at ease in their colleagues' presence [57][53].

An open space layout enables team members to have ad-hoc conversations with their colleagues, where they can ask for help, guidance or any other sort of assistance required, which allows for better knowledge sharing and can make the group more cohesive.

Consequences

Despite the more inclusive atmosphere that an open space office provides, team members can feel overwhelmed and distracted due to their apparent constant availability and exposure to the environment [57].

This can be mitigated by the addition of quieter areas where they can relax and have some privacy and alone time [52], and by making use of INFORMAL COMMUNICATION SPACES.

* * *

Pattern: INFORMAL COMMUNICATION SPACE

Agile software development is a collaborative process. Since team members and stakeholders are often working like a CO-LOCATED TEAM would, it is normal that face-to-face communication, a highly useful means of transmitting information, is also commonplace in these environments.

Problem

Despite the usefulness that face-to-face communication enjoys in agile contexts, it is important that it does not take over the entirety of the development team's time. As such, one must be aware that there is a time and a place for everything.

What is the best environment for informal communication?

Forces

- **Focus.** One can describe software development as being akin to building a tower. A developer must tie ideas together, ensuring that the resulting code is coherent and that its logic is sound. This process requires a lot of concentration, and this tower falls apart if the developer loses focus [16], which can happen if they are suddenly roped in a conversation, which is not advantageous to the project.

- **Encouraging interactions.** Informal communication should be encouraged within the organisation because it fosters TRUST and knowledge sharing within the project personnel, and not stifled in order to ensure the developers spend their whole workday developing code.

Solution

Areas that are far enough from workstations to avoid disturbing people who are working but close enough where people are in the vicinity often are good spaces to dedicate to informal communication.

The distance from workstations makes it so that people engaging in conversations can avoid disturbing any team members who might be busy coding. The area in question can also have some INFORMATION RADIATORS, such as whiteboards, to support any conversations happening there.

This area should also be accessible to everyone and, perhaps, a place where people often walk by or that is close enough to the DEVELOPMENT TEAM's workstations. This ensures that if a team member needs to talk to a colleague, there is a high chance they will stumble upon them in that space during the day.

Related patterns

A SNACK SHRINE [68] and THE WATER COOLER [20] are good examples of INFORMAL COMMUNICATION SPACES, as they provide central gathering places around which informal conversations can happen.

***

Pattern: CUBES

The software development world is getting more and more connected across borders and continents. As such, it is not uncommon to find situations where project members have to participate in meetings using CALLS.

Problem

When participating in a conference call, good acoustic conditions are a requirement, so that all parties can understand what is being said. If the call includes a video feed, then that should be of quality as well.

How can we create good conditions for conference calls or small spontaneous meetings?

Forces

- **Subject sensitivity.** Meeting participants may want to avoid outside parties listening in on meetings that are dealing with sensitive or confidential subjects.
- **Background noise.** The place where the call participants are in should be as quiet as possible to maximise everyone's understanding of the discussion taking place.

- **Hardware.** Depending on how conference calls are set up in the organisation, specific hardware may be necessary, even if it is just a screen and a camera for video conferences.

Solution

Using acoustically-sound rooms or spaces, that can fit either a couple of people or a large group, provide good conditions for calls and meetings.

These acoustically isolated rooms (CUBES) can be fruitful to ensure that project members can participate in conference calls without having to worry about noise, which can be distracting and may lead to communication issues.

If a large group of people (such as a DEVELOPMENT TEAM) is participating in a call, a meeting room may be suitable for them. However, for individuals or pairs, one can go with commercially available "phone booths" or with small rooms.

Examples

There are several companies offering soundproof booths as commercial solutions to enteprises, such as ROOM[1], Zenbooth[2] and Urban Office[3].

Related patterns

As mentioned earlier, CUBES provide a support role to CALLS. However, if a CUBE is large enough for a small group of people, they can also be useful rooms for meetings such as RETROSPECTIVE MEETINGS.

* * *

[1] https://room.com/
[2] https://zenbooth.net/
[3] https://urban-office.com/office-phone-booths.html

Pattern: CALLS

In agile projects, interactions between the members of the DEVELOPMENT TEAM are common and even required to happen. However, in today's connected and globalised world, distributed teams are not as rare as they used to be, and there may be occasions where some of the work done is remote.

Problem

In the event that some team members need to meet, there should be a way that any magnitude of distance between them is not an issue, allowing them to meet and interact synchronously.

How can we contact with a person in a synchronous manner who is far away?

Forces

- **Meeting urgency.** Depending on the topic at hand, there may be a need for a better back-and-forth between involved parties. For example, this reduces the validity of e-mails as a communication mechanism, as replies happen less frequently, and the synchronous nature of the interaction disappears.

- **Communication temperature.** The temperature (or richness) of a communication channel is related to the amount of information conveyed by the participants during their interaction. Communication channels tend to be more effective if their temperature is also higher, as one can obtain information from the tone of voice, facial expressions, along with being able to interact directly with the other party [16].

- **Hardware.** Given that the meeting parties are physically distant, one can infer that it will happen through the use of technology. As such, the participating parties should have the hardware, software and other infrastructure required for the meeting.

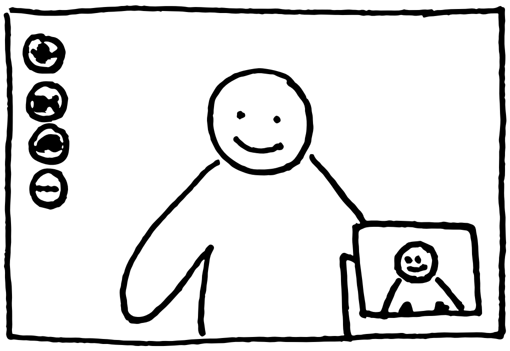
Solution

Two (or more) people far apart from each other can call each other using a phone, a computer or any similar device to have a synchronous conversation.

When a meeting between people that are not physically together must happen, they can opt for having a conference call, ensuring that the synchronous aspect of the communication is present.

These calls can be audio-only or accompanied by a video feed and are usually based on Internet protocols, although using telephone networks is not uncommon.

Depending on the service in question, calls may be one-on-one meetings or may have several participants. Some services also allow for sharing one's screen, which can be useful for presentations or product demonstrations.



While participating in a call may only require a phone or a computer, organisations often have meeting rooms with a workstation and a camera suited to having a group of people participating in one end of a call.

Examples

People can use phone calls when the presence of a video feed is not imperative or use computer software, such as, for example, Skype[4], Google Meet[5], Microsoft Teams[6], and Zoom[7].

Related patterns

The usage of CUBES may enhance team members with an environment that is better suited for calls to take place.

* * *

Pattern: INSTANT MESSAGING

Agile teams rely on communication and teamwork between their members. However, in today's globalized environment, there may be instances where the DEVELOPMENT TEAM members are not working in the same physical space and, as such, digital, far-reaching means of communication are necessary.

Problem

While working on projects, either with distributed teams, with someone who is not in the same place as their colleagues, or even with someone who just seems to be currently busy, a need for quick, short bursts of information between colleagues may arise.

How can one reach another person with quick information?

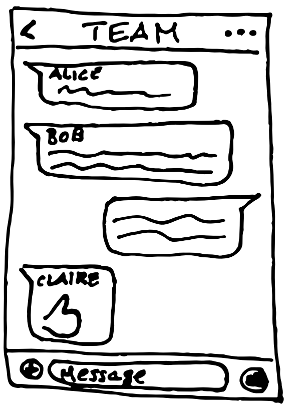
[4]https://www.skype.com/
[5]https://meet.google.com/
[6]https://teams.microsoft.com
[7]https://zoom.us/

Forces

- **Speed.** In some occasions, it is vital to send the information as fast as possible to the recipient. The reasons for this increased haste may be related to windows of opportunity or relevance, fear of forgetting what is being transmitted or due to approaching deadlines.
- **Convenience and ease of use.** It is important to be able to reach another person with certainty that they will receive the information being transmitted. It is common for people to carry a smartphone or another sort of internet-connected device with them, and, as such, these are viable means to reach another person.
- **Distance between parties.** It is important to be able to use a means of communication that can work regardless of distance, as latency and reliability can hinder the communicative process. As such, using something versatile enough to the point that it can work across a room, a building or even across different regions ends up being a powerful ally to communication.

Solution

Using a computer or another internet-able device, a person can send short text messages that are sent instantly to their recipient.



This can be used to quick information regardless of distance, and is done through instant messaging (IM) services. These services allow users to send text messages in real-time, and some even allow sending other types of media, such as documents and images.

IM services allow users to have direct, one-on-one conversations with other people, as well as providing the ability to create chat rooms where multiple users can interact with each other. Some IM services also provide voice and video chat, which may also be useful for CALLS. Conversation logs and search features are also commonplace, allowing for easier consultation regarding previous interactions.

Some IM services include integration with OPEN INFORMATION TOOLSETS, which can enhance the communication aspect between users by making it easier to provide additional context and relevant documents. They may also be available both on computers and on mobile devices, to make it easier for users to reach their colleagues if necessary, wherever they may be.

Examples

There are several enterprise-oriented IM services available in the market, such as Slack, Microsoft Teams and Riot.im. However, users may also use services that are more oriented towards the consumer market, such as WhatsApp, Discord or even using programs that use open-source protocols such as XMPP or Internet Relay Chat (IRC).

Related Patterns

This pattern contrasts with E-MAILS, which are used for larger, more substantial messages that are sent at a lower pace.

* * *

Pattern: E-MAILS

Agile software development is a collaborative endeavour that works thanks to the effort coming from people from all roles, including DEVELOPMENT TEAMS, COACHES, CUSTOMERS and the MANAGEMENT.

Problem

There may be instances where people need to communicate, but either face-to-face communication is not possible, a written message is required, or there is a need to store any information conveyed.

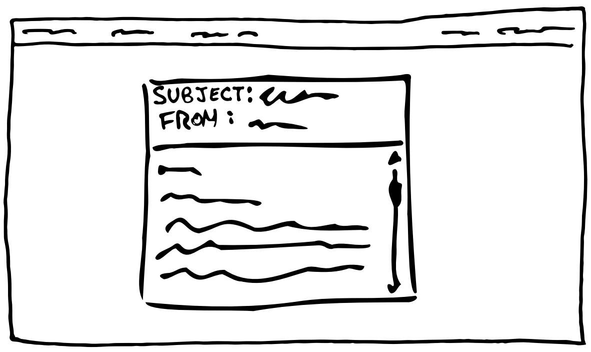
How can we send information that is lasting to team members, regardless of distance?

Forces

- **Mass communication.** Collaboration on certain topics may entail the involvement of several different people, who can easily receive information via mass communication methods.
- **Accessibility.** Collaboration with different people is easier when they are accessible, regardless of any physical distance between those parties.
- **Information stickiness.** Writing things down in a way that they can be consulted ensures the information is not lost [16]. Additionally, the act of transferring information into an environment that can be useful for archival also encourages it to be more substantial.

Solution

Use e-mails to address team members, which provide pondered, substantial information and that can be archived and referred to later on.



E-mails are one of the most ubiquitous communication methods of today's current corporate world, with almost all organisations supplying their employees with a corporate e-mail address. This results in increased accessibility, allowing for the relevant people to be contacted.

Given their resemblance to letters, e-mails provide users with the chance to write longer, more pondered and substantial messages and at a slower pace. Additionally, they can be archived and consulted later on if necessary.

Related Patterns

This pattern contrasts with INSTANT MESSAGING, which is more useful when communication needs to be fast but its archival is not needed.

* * *

Pattern: OPEN INFORMATION TOOLSET

Agile software development projects are often extensive, being carried out by several people with vastly different backgrounds and roles who coordinate their tasks and share their knowledge.

Problem

Although the Agile Manifesto prioritizes "working software over extensive documentation" [5], the DEVELOPMENT TEAM, along with the project stakeholders (the MANAGEMENT and the USERS/CUSTOMERS), may write down records of their activities.

How should the project personnel archive any written documentation and other records they produce throughout the project?

Forces

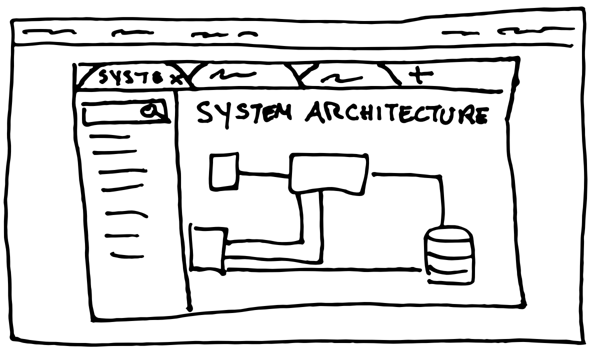
- **Accessibility.** Project personnel should be able to access archived information regardless of where they might need to do so, as it is possible that one may find themselves working away from the usual work area.

- **Ease of use.** Any tools used for document archival should provide the fewest resistance possible to data input [13], as documentation is often sidelined in agile projects due to the need to maintain it.

- **Longevity.** Project personnel may need to access older documents (or even from previous projects), and they should be available for consultation if required.

Solution

Adopt web-based tools where users can easily search for and add new information to provide a good platform for information archival and knowledge sharing.



Platforms where users can consult information about the work they have at hand should be accessible wherever the need for consultation may arise. As such, web-based solutions are a worthwhile choice, as they can be accessed from any device with a web browser.

The platform in question should be easy to use, regarding browsing archived information and adding new data. Documents should be able to link to each other, and different file types should be permitted in the system [13].

These features allow for the project personnel to archive system documentation, architectural documents, meeting notes, diagrams, backlog items, iteration plans, among other data [13].

Examples

Company intranets are common applications of OPEN INFORMATION TOOLSETS, as well as wikis and cloud-based storage drives.

Related Patterns

These can be used as complements to INFORMATION RADIATORS, as both elements provide ways for the DEVELOPMENT TEAM to stay up to date and informed.

4.2 Activity patterns

Pattern: DAILY MEETINGS

Agile development processes rely on teamwork; as such, projects are large enough to the point they are split into sub-tasks, which are distributed among team members. As iterations in agile projects often have goals the team works towards, it is necessary to keep everyone up to speed on the current status.

Problem

Face-to-face communication is regarded as the most effective way to transmit information within agile contexts [5]. However, a logistical hurdle appears when it is not viable for everyone to relay their current status within the project to other team members individually.

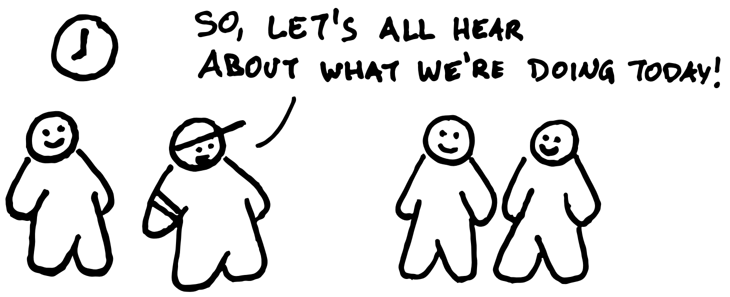
How can we keep everyone on the team updated regarding the current status of the project and re-planning?

Forces

- **Goal control.** Agile methodologies often implement some sort of goal to keep the team members focused on the tasks that need to be done, as is the case with Scrum and LeSS [69][45].
- **Short iterations.** Iteration lengths are usually short (commonly between one and three weeks) and since the project status can change very often, there is a need for frequent updates to the team and re-planning.
- **Plurality of involved parties.** The DEVELOPMENT TEAM, along with the stakeholders, comprise a plural number of organisation members, which all must be updated regarding the project status.

Solution

Every day, the DEVELOPMENT TEAM should meet and discuss their progress on the project, their current tasks and any problems they might be facing, and re-plan if needed.



The Daily Meeting presents itself as a way to keep everyone informed in a regularly-scheduled fashion. These meetings are held at the start of every day before everyone starts working on their tasks. As such, all parties can impart and receive information about their influence on the project status [69][45][4].

Each development team member discusses what they have achieved so far and their current plans, which aids in ensuring everyone is mindful of the current iteration goal, as well as what has changed since the last meeting. Additionally, team members also tell if there have been any hurdles needing overcoming; this helps in creating awareness for potential problems and how to eliminate them.

Examples

XP implements this pattern using the Daily Stand Up Meeting [77], while Scrum and LeSS do so through the Daily Scrum [69][45].

* * *

Pattern: PLANNING MEETINGS

In agile projects, the DEVELOPMENT TEAM works in iterations, building the product in increments. This iterative approach to building software allows for the USERS/CUSTOMERS to provide relevant FEEDBACK, reducing wasted time.

Problem

During the course of a project iteration, the DEVELOPMENT TEAM should have in mind the tasks to be done, so that the increment is consistent with itself and with has already been done previously. The agile principle of having working software being the primary measure of progress [5] furthers this point, urging team members to work towards the same path.

How can we lay a proper foundation to the next iteration's work?

Forces

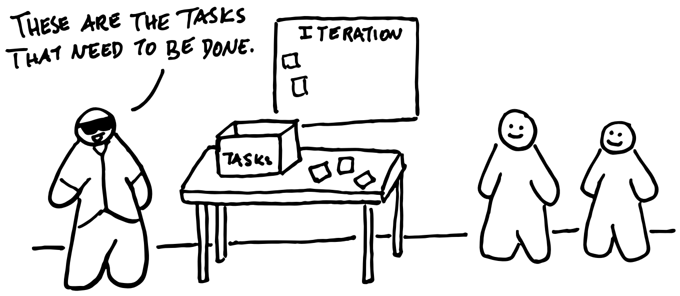
- **Client Involvement.** Agile methodologies rely heavily on having the CUSTOMERS as involved parties in the development process, allowing for requirements that are effectively representative of the desired result.

- **Adaptability.** The iterative nature of agile methodologies allows for a more quick and swift reaction to shifting market demands, which may or may not be conveyed to the team by the CUSTOMERS. These reactions tend to occur between the end of an iteration and the start of the next one.

- **Staying on track.** In order to reduce wasted efforts, the team should be working towards the same goal [16]. Having planned out what to do for an iteration can reduce the number of tasks done with no pertinent reason or that end up unfinished, which hinders the iteration's resulting increment.

Solution

Before an iteration starts, the DEVELOPMENT TEAM and the CUSTOMERS should decide which tasks should be done during the iteration.



During this meeting, the CUSTOMERS should determine what needs to be done and lay out the tasks that the DEVELOPMENT TEAM should tackle during the iteration to establish a clear course of action.

Having these objectives laid out from the beginning reduces wasted efforts, as the team ends up knowing what they should focus on with their work.

Examples

The Release Planning meeting is a version of this pattern applied to Extreme Programming [4], while the Sprint Planning meeting can be found in Scrum [69] and LeSS [45], fulfilling the same general role.

Related patterns

This pattern helps with the team's GOAL SETTING and its relationship with the CUSTOMERS. Additionally, the SPRINT PLANNING pattern of the Scrum Patterns [68] is a version of the way this meeting works in Scrum.

* * *

Pattern: REVIEW MEETINGS

In agile software development projects, the people involved in them employ an iterative approach to develop the products. So, at the end of each iteration, the DEVELOPMENT TEAM will have produced an increment of the product.

Problem

The Agile Manifesto states the importance to respond to change [5] as one of its core values. Additionally, the team needs to evaluate their work to ensure that they are delivering working software without issues.

How should the team evaluate and review the work done during the iteration?

Forces

- **Changing circumstances.** Agile projects provide greater flexibility to plan about future steps when compared to more traditional, plan-driven paradigms. As such, in the event that the market shifts or unexpected events happen, one can guide the project a lot better and adapt to unforeseen circumstances.

- **Stakeholder involvement.** Agile processes rely heavily on the interaction between the DEVELOPMENT TEAM and stakeholders such as the CUSTOMERS. These interactions with outside parties are good sources of FEEDBACK and provide opportunities to think outside the box.

- **Staying on track.** As agile projects are composed of several iterations, it is possible for one to start losing track of the overall goal of the project, focusing only on the individual iterations. Having moments where one can focus and look at the bigger picture is necessary.

Solution

At the end of an iteration, the DEVELOPMENT TEAM should meet with the CUSTOMERS and other stakeholders to present the increment built during said iteration.

This meeting is an opportunity to evaluate the work done and to gather FEEDBACK that may be useful to use during the next iteration. It is also an opportune time to assess any changes to the market and to review the overall project status [69].

The outcomes of this meeting are useful for the next PLANNING MEETINGS, which should occur before the next iteration begins.

Examples

Both Scrum and Large Scale Scrum make use of Sprint Review meetings to assess the work done during Sprints [69][45].

Related Patterns

These meetings are a good opportunity to develop the working relationship between the DEVELOPMENT TEAM and the CUSTOMERS.

***

Pattern: RETROSPECTIVE MEETINGS

One of the principles of the Agile Manifesto states that the DEVELOPMENT TEAM should regularly reflect on their work and adjust how they do it [5], improving their effectiveness and strengthening the developers as a team.

Problem

Amidst all the work the DEVELOPMENT TEAM has to do, along with the iterative nature of agile processes, the team should have some time allocated to reflect on their work and where it can improve.

How should the DEVELOPMENT TEAM *reflect on their work in order to grow?*

Forces

- **Need for improvement.** Human beings are not perfect and, as such, are always gaining experience with the work they do. Being able to harness this experience will yield better results in the future.
- **Timing.** The iterations the DEVELOPMENT TEAM goes may end up being dense in terms of workload, as well as the team needing some allocated time to be together during these times of reflection.
- **Freeing up one's mind.** Moments of reflection require a proper state of mind in order to function as intended. One must be able to take their mind off any tasks they might have at hand so they can focus on reflecting on what they have learned.

Solution

Between the end of an iteration and the beginning of the next one, the DEVELOPMENT TEAM should meet maybe with a facilitator (e.g. the COACH) somewhere quiet and reflect on the team's work and on how to improve it.

The meeting participants should gather at a place where they might feel at ease, preferably outside of the work area. The resulting distance from their workspace allows the team members to not worry about their current tasks as much, and having a more neutral ground may help with getting into the right mindset [39].

This meeting must happen after the iteration ends, usually preceded by REVIEW MEETINGS. The COACH serves as a facilitator for the members of the DEVELOPMENT TEAM, who will think about both strong and improvement points concerning their work and discuss potential ideas to try during the next iteration. This meeting is also a great way to gather FEEDBACK within the team.

Examples

Scrum and LeSS implement this meeting as Sprint Retrospectives [69][45].

Related patterns

These meetings require TRUST between team members and knowing the ins and outs of giving and receiving FEEDBACK.

The Scrum Patterns implement this meeting in the form of the SPRINT RETROSPECTIVE [68].

* * *

Pattern: PAIR PROGRAMMING

During the development of a project, one may wonder what the best way for the DEVELOPMENT TEAM to work is. In agile contexts, teams are often CO-LOCATED, perhaps working even in an OPEN SPACE.

Problem

It is important to maintain the trust and teambuilding team members have in each other, along with their knowledge and awareness regarding the product being developed. This need is intensified when the team has newer members, who may not feel at ease or know much about the system.

How can the DEVELOPMENT TEAM *better distribute its knowledge?*

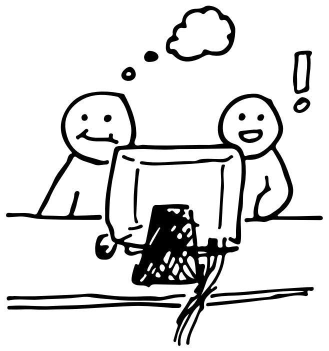
Forces

- **Knowledge sharing.** There is a need for a good distribution of knowledge in the team; this makes it so that more than one team member is familiar with specific details of the system, decreasing the bus factor [20].

- **Camaraderie.** The act of achieving results together, regardless of its size or relevance to the project, helps in making people feel more comfortable around each other [16].

Solution

DEVELOPMENT TEAM members should pair up and work together at the same workstation, while one codes and the other provides support, sharing knowledge and building a working relationship.



Pair programming consists of having two development team members working side by side at the same workstation. In this situation, one developer, the "driver", controls the mouse and keyboard, while their pair, the "navigator", provides support to the task at hand [4].

While pairs can stay together and work regularly with each other, team members can also partake in "promiscuous pair programming" [7]. In this variant, pairs rotate with each other after a set period of time.

These practices allow for team members to learn about the system off of each other while feeling increasingly more comfortable within the group [17].

Related patterns

Making progress in the project through pair programming allows for DEVELOPMENT TEAM members to feel more at ease with each other, increasing the TRUST between themselves.

* * *

Pattern: BACKLOG REFINEMENT

In agile projects, the DEVELOPMENT TEAM receives the requirements either directly from the CUSTOMERS or through a surrogate before dividing and converting them into more accessible tasks or user stories, easing their implementation. These items are then organised into a backlog of things to be done during the project.

Problem

Depending on the project's scale and duration, it is possible that the market may undergo through changes, the CUSTOMERS' priorities may shift, or even the project may evolve organically into something different.

How can we stay adaptable and keep up with the needs of an evolving project?

Forces

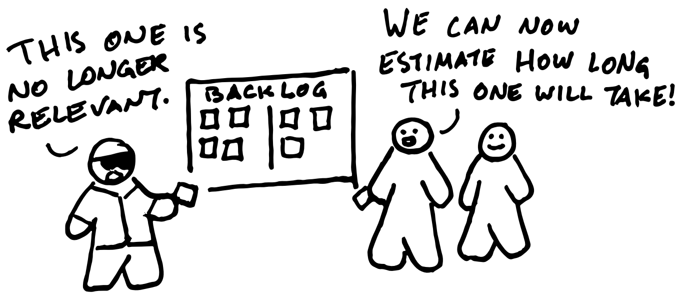
- **Iterative work.** Work in agile projects is done iteratively; each iteration should be preceded by some sort of PLANNING MEETING where the main tasks for the iteration are laid out so the team can focus on delivering working software by the time the iteration is over.

- **Direct contact.** Agile projects benefit from direct contact between the DEVELOPMENT TEAM and the CUSTOMERS; this allows the team to know the latter's current priorities regarding which requirements the DEVELOPMENT TEAM should implement in the first place.

- **Goal maintenance.** Unlike traditional, plan-driven software development projects, which stay rigid regarding any defined objectives, it is normal for the roadmap to shift and change in agile projects [5]. As the project evolves, it is important to do maintenance on what is already planned to see if it still makes sense for the project.

Solution

The DEVELOPMENT TEAM, along with the CUSTOMERS, should assess the backlog, remove or sort existing items depending on their relevance, and add new ones if necessary.


THIS ONE IS NO LONGER RELEVANT.
BACKLOG
WE CAN NOW ESTIMATE HOW LONG THIS ONE WILL TAKE!

Reviewing the backlog entails removing items that are no longer relevant to the current state of the project, dividing larger items into smaller attainable tasks, re-evaluating item priorities, assessing and adding estimates, and creating new items [69].

Performing this maintenance task, either spontaneously or by allocating some time to it, helps the DEVELOPMENT TEAM have a unified perspective regarding the current state of the project. Having the CUSTOMERS involved in this process also ensures that the end result stays true to their vision.

Examples

Scrum and LeSS usually allocate up to a tenth of the team's capacity to Backlog Refinement [69][45].

Related patterns

This process of backlog refinement benefits from GOAL SETTING practices.

* * *

Pattern: COLLABORATIVE DECISION MAKING

Agile software development environments involve plenty of interactions and decisions to take place.

Problem

DEVELOPMENT TEAMS, being self-organised in nature, may find difficulties when having to make decisions about their work.

How should decisions be made in agile settings?

Forces

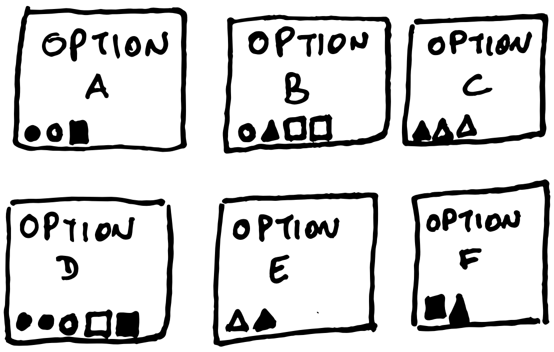
- **Customer involvement.** Agile methodologies rely on direct communication between the DEVELOPMENT TEAM and the CUSTOMERS, removing potential information losses in the process that the requirements travel from the interested parties to the people who will implement them.

- **Self-organisation.** DEVELOPMENT TEAMS are often self-organised in nature [5], sharing leadership between its members, who share relationships of interdependence with each other. The team derives its purpose from within, rather than receiving it from the organisation.

- **Commander's intent.** The commander's intent is a military concept that states the desired mission outcome. Military situations are high-risk scenarios and, as such, the troops may not be able to receive orders that adapt according to the circumstances. The commander's intent provides the team with the power to take action and be bold, as long as they reach a successful outcome [65].

Solution

The DEVELOPMENT TEAM should make use of collaborative decision-making methods, such as finding consensus, trying to reach a compromise, or applying a voting method, in order to make decisions.



Collaborative decision making allows for the team to reduce any cognitive bias they may have regarding the matters at hand, bringing the advantages of providing new perspectives and additional insight, as well as improving people's COMMITMENT by providing them with motivation to support something in which they were involved. It is essential to establish a clear objective and find the root cause of the issue, ensuring that the ideas discussed are pertinent [79].

The facilitator (or the team) should choose a method to employ after they find NEW IDEAS. When there is consensus within the people involved, they all find that the idea is acceptable and agree with its implementation. Consensus may appear spontaneously, but it is often built. In this process, people engage in open discussion, voice their concerns, collaborate in developing proposals, choose a direction to follow and, after drafting a final proposal, it should be chosen with everyone's support [29].

If a consensus cannot be reached, a compromise may be an acceptable alternative, particularly when the group is divided between two opposing options. When attempting to reach a compromise, each party concedes part of their idea, eventually reaching a point where everyone can agree on what is being discussed.

Finally, voting methods take people's choices and then derive a final choice from them. There are several types of voting methods, including plurality voting, majority voting and multi-voting. Plurality voting, also known as first-past-the-post, states that the option with the most votes is the winner, even if it did not get over half of them [40]. Majority voting states that the winner is the option that has over fifty per cent of the votes cast [40]. Finally, in multi-voting, people get a set number of votes to hand out, which is usually around one-third to one-half of the options available.

Then, participants can distribute their votes as they please. This can be used to narrow the viable options or to choose the final choice [14].

Related patterns

Team members should pay attention to the level of COMMITMENT within the team, as low levels can lead to team members being stuck (whether due to a lack of information or a lack of consensus) and not being able to make a decision.

4.3 People patterns

Pattern: COACH

Software projects are often team efforts, requiring a plurality of people to create the intended project. When employing agile methodologies in these projects, we find the DEVELOPMENT TEAM interacting with the CUSTOMERS.

Problem

The adoption of agile methodologies is not an easy task, as they are different compared to what the clients and maybe even the developers are used to encountering. The people involved may need some guidance on how to behave and on what to expect from the process.

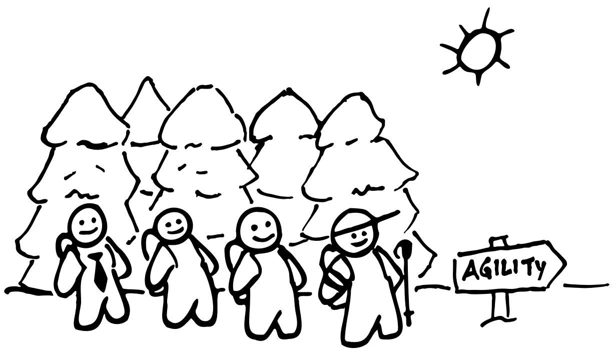
How can we support the team and the clients in the adoption of agile methodologies?

Forces

- **Perpetuity.** The adoption of agile methodologies is a process that does not end, with its practitioners adopting a process of continuous improvement.
- **Paradigm shift.** Agile methodologies are vastly different from traditional "plan-driven" methodologies, and it requires the people involved to change the way they face their work [61].
- **Environment.** The working environment should have the necessary conditions for agile practices to take place; without them, the team and the stakeholders may not harness the advantages of these methodologies.

Solution

The COACH provides a support role to the DEVELOPMENT TEAM from within the organisation, assisting them and the MANAGEMENT in the adoption and learning process of agile methodologies, removing obstacles, and facilitating meetings.



The adoption of agile methodologies is a process that incurs a learning curve, as they are different from traditional, plan-driven methods and require a paradigm shift [61]. The Coach should be a person who is well-versed in agile methodologies, and that can guide people regarding new practices and skills to develop [23].

The coach also partakes in other activities, such as collaborating with the Management to remove any obstacles the DEVELOPMENT TEAM may be facing and facilitating meetings like DAILY MEETINGS or RETROSPECTIVE MEETINGS [23].

Examples

Scrum and LeSS have the Scrum Master, who supports the team in the adoption and usage of the methodology in question [69][45].

Related Patterns

The Scrum Patterns approach the concept of the COACH through the SCRUM MASTER [68].

* * *

Pattern: DEVELOPMENT TEAM

In an enterprise environment, one does not build software alone. Applications and programs have been becoming increasingly complex and extensive over the past years, turning their conception into a collaborative endeavour.

Problem

Given that there is a need for multiple people working on software, an organisation should know what the best way is to ensure their developers can work together productively and effectively, ensuring high product quality.

How should the developers organise themselves to work?

Forces

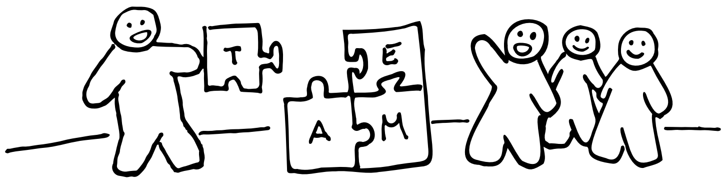
- **Knowledge management.** Seeing that developers are individual beings, they possess some own knowledge, which can grow throughout the project as they learn about the system they are developing and the tools they are using. This knowledge may be then shared throughout the organisation by the developers.

- **Cooperative work.** Building software requires being able to deal with several areas of functionalities, such as front-end and back-end development, which end up being very different from each other. Thus, since developers often tend to specialise in an area over other ones, they need to be able to work with others in a way that they complement each others' weaknesses.

- **Group vs. team dynamics.** While groups and teams can be both defined as being sets of individuals, they have fundamental differences which may make one more viable over the other depending on the circumstances. Working groups tend to have a defined leader, a mission that reflects the organisation's own, a lack of sharing accountability resulting from each person alone, efficient meetings, a bigger presence of delegation, and have their effectiveness measured indirectly. On the other hand, in teams, their members work together and share leadership, there is both individual and shared accountability, their purpose is created from within, meetings are directed towards discussion and problem-solving, and its performance is measured by assessing the work done [37].

Solution

The developers should organise themselves into a DEVELOPMENT TEAM, where they work together and support each others' activities, share leadership, hold each other accountable, share a strong bond, and work towards a common goal.



This organisation into teams happens as building software requires interdependence by the part of the developers, and the individuals should organise themselves according to the description of teams present in the third force above.

Development teams should be cross-functional, being able to implement features from beginning to end, covering all necessary architectural areas. As such, team members complement each other's weaknesses.

In order to reduce the overhead generated by shifting personnel and so that the developers can establish working relationships with each other, teams should be long-lived. This factor also helps in building trust between team members, as they start to know how their colleagues work and what to expect.

Knowledge sharing is also important within the team. Individual tacit knowledge about the product in development can be made explicit and shared within the team and combined with other explicit knowledge available, which is later on internalised into tacit knowledge by the developers [54]. As such, the team can improve their perception of their work thanks to what their teammates have learned themselves.

Related patterns

DEVELOPMENT TEAMS rely heavily on the existence of TRUST between their members as well as their COMMITMENT. Their long-lived nature is also an example of STABLE TEAMS, one of the Scrum Patterns [68].

* * *

Pattern: CUSTOMERS

One of the values behind the Agile Manifesto [5] is the importance of customer collaboration in agile software development.

Problem

The interactions the customers have with the organisation and its personnel should be productive and not hinder the DEVELOPMENT TEAM's work.

How should the project's recipients interact with the organisation?

Forces

- **Communication efficiency.** Melnik and Faurer [50] state that there is a significant loss of knowledge in Tayloristic knowledge sharing, as information flows through a chain of roles and details are warped or missed before reaching the final recipient. In contrast, agile knowledge sharing focuses on direct communication between sender and recipient, reducing information losses.

- **Receiving necessary information.** The people who know the most about the desired end results are the customers, as they know which features and user stories have the highest priority and can provide any necessary feedback [33].

Solution

The CUSTOMERS should provide timely feedback to the DEVELOPMENT TEAM, clearing up any questions they may have.

The CUSTOMERS are usually the people who ordered the software or who are overseeing its creation; they might also be, in other situations, end-users who provide FEEDBACK.

They define the requirements for the software to be developed, establishing priorities and providing FEEDBACK on the result of each iteration. This direct contact with the DEVELOPMENT TEAM helps in reducing misunderstandings and losses of information, as the team can ask clarifying questions to ensure that everyone is on the same page.

While they may be present in the DEVELOPMENT TEAM's day-to-day activities, the CUSTOMERS' presence is more common and expected in PLANNING MEETINGS and REVIEW MEETINGS, as those events provide opportunities for the CUSTOMERS to give inputs about the current state of the project.

Examples

Extreme Programming has the CUSTOMERS as elements of the DEVELOPMENT TEAM, playing a highly involved role in the rest of the team's daily operations [77].

In Scrum, customers may not interact directly with the DEVELOPMENT TEAM; instead, the Product Owner manages the backlog and maintains a consistent vision for the project [69].

Related patterns

When the DEVELOPMENT TEAM is working on a project where CUSTOMERS are absent or inexistent, the organisation can use a SURROGATE CUSTOMER [20] to fill in the CUSTOMERS' shoes.

The Scrum Patterns [68] approach this topic with the PRODUCT OWNER and the DEVELOPMENT PARTNERSHIP.

* * *

Pattern: MANAGEMENT

Agile software development projects often function in an enterprise setting, where the organisation has a defined administrative body (the management) that manages the company's strategy and efforts.

Problem

The organisation's management has a considerable influence, even if indirectly, on the organisation's results. The management deals with staffing, budget and tools, among other factors, which has repercussions on the teams' abilities to get things done.

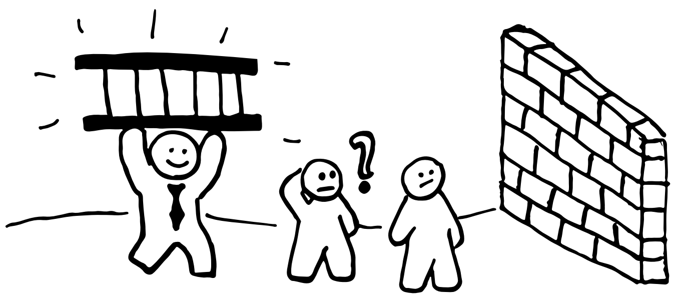
How should the behaviour of the organisation's management be facing agile projects?

Forces

- **Agility and change.** One of the principles of the Agile Manifesto [5] states that changes to the project are welcome, even if it is already late in development. Agile methodologies can handle these changes, while traditional, plan-driven ones will face difficulties in doing so.
- **Organisations as ecosystems.** One can compare organisations to living ecosystems [31]: both these concepts include a plurality of individuals interacting with each other and with their environment. These interactions cause individuals to react to changes and other events, making the current state of the system an ever-evolving concept.
- **Informal environments.** Agile organisations may also be AD-HOC ORGANISATIONS, where organisation members interact freely with each other and collaborate on their work, not worrying about rank or hierarchy.

Solution

The management should support their teams, forecasting needs, removing external obstacles, and working with the COACHES to remove internal ones.



In agile projects, DEVELOPMENT TEAMS work as self-organised entities, deriving their purpose from their own work instead of it coming from the organisation. As such, the MANAGEMENT should play a supporting role and not a directorial one.

The support the MANAGEMENT provides starts with anticipating any needs the @Development Teams may have, as it helps with maintaining their productivity and a sustainable pace.

The DEVELOPMENT TEAM's agility and ability to react to changes is reduced when they encounter obstacles; as such, the management should remove them as soon as possible, having the means to remove external obstacles (such as ones related to logistics and external relationships).

When it comes to internal obstacles, however, collaborating with the @Coaches is a good course of action, as they tend to be aware of the necessities the team may have.

Related patterns

The MANAGEMENT should place their TRUST in the DEVELOPMENT TEAM and their work, which they can evaluate by participating in REVIEW MEETINGS.

***

Pattern: AD-HOC ORGANISATION

The people working in agile projects are not restricted to interacting in the context of their project. One can compare organisations to living ecosystems [31] and, as such, people exist within the entirety of an organisation and not just in a team or a project.

Problem

The organisation's personnel establish a working relationship, which should be compatible with their interactions, fostering a collaborative environment that is necessary for agile development.

How should the working relationships of the organisation personnel function?

Forces

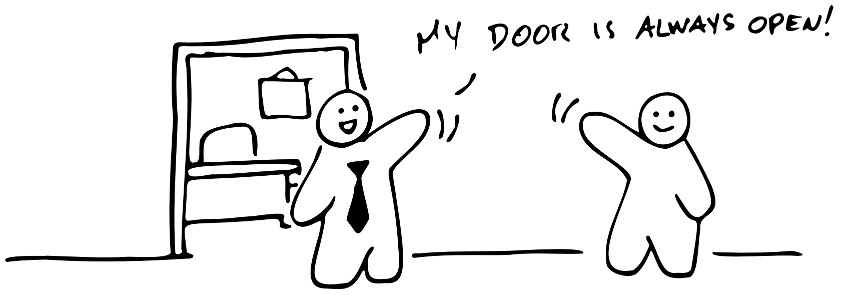
- **Red tape.** Speed is a vital factor to be mindful of when reacting to change. As requirements and the market shift, having to wait for the MANAGEMENT to give the DEVELOPMENT TEAM the go-ahead they require hinders their productivity and reduces their agility, potentially costing more money to the organisation in the long run.

- **Communication and middlemen.** Humans are not perfect and, as such, it is expectable that if someone has to transmit information they received to someone else, not all information will go through, as the content changes and loses details [50]. Removing middlemen and attempting to communicate directly helps in reducing these communication errors.

- **Team self-organisation.** DEVELOPMENT TEAMS are cross-functional groups of people that, in addition to sharing their leadership and working and making decisions as a single unit, they derive their purpose from their work [37]. This self-organised nature of teams, in addition to the complexity and uncertainty of agile software development projects, make it so that decisions should come from who is working on the project instead of someone that is more removed from the situation.

Solution

The organisation members should feel free to interact with any other person if the need arises, regardless of rank or area of operation.



Agile software development is a cooperative and collaborative endeavour and, as such, organisation members should not be afraid to ask each other for advice, even if one is asking a colleague from another team. Additionally, COMMUNITIES can aid this inter-team collaboration.

The teams' self-organisation and COMMITMENT, in addition to the TRUST people have in each other, are factors that are vital for this to work. However, organisation members should be mindful of the pertinence of their interactions, as one being overzealous of their freedom of interaction can hinder productivity instead of being a source of FEEDBACK and NEW IDEAS.

Related patterns

INFORMAL COMMUNICATION SPACES can enhance this informal communication atmosphere. As mentioned earlier, TRUST and COMMITMENT are important for this to work.

* * *

Pattern: COMMUNITIES

When an organisation has a large project, its work is often divided among several teams, which work in a coordinated way to complete their goal. On some occasions, these teams may dedicate their work to one or more program features, being cross-functional in nature.

Problem

The members of a cross-functional team may tend to specialise in the area focused on by their work. This makes it harder for team members to discuss the intricacies of their work with their colleagues and to share their discoveries in a way that can be seized to advance the project even further.

How can we foster cross-team communication and knowledge sharing?

Forces

- **Cross-functional teams.** The members of these teams tend to be more specialised in some or other area compared to their teammates. As such, learning new things about their area of expertise may be difficult if they rely solely on their team.
- **Inter-team communication.** Large projects with several teams working on them require cooperation and coordination between those parties, in order to achieve a cohesive and working result.
- **Sharing culture.** In organisations that employ traditional, command-and-control management styles, information flows vertically and very rarely horizontally, as the people in the top positions make the decisions. Contrastingly, organisations that employ a collaborative leadership style decentralise the decision-making practices [31] and, as such, useful knowledge, such as new techniques and good practices, should be distributed within the company.

Solution

COMMUNITIES are groups that exist parallel to the organisational chart and allow organisation members to exchange knowledge between teams, increase inter-team collaboration and build a sense of unity within the organisation.

These groups are informal, voluntary and self-organised, consisting of organisation members who are interested in a particular topic [36]. They tend to arise due to the cross-functional and independent nature of teams, which can lead to team members not being able to share their knowledge or trying to find a tool or a process that a colleague from another team has already found.

Communities tend to not adhere to organisational boundaries, existing parallelly to the organisation chart and allowing for an increased horizontal knowledge sharing [11]. This factor allows team members to interact and collaborate with other organisation personnel that have similar interests and specialisations, increasing the unity within the organisation.

Examples

LeSS applies this pattern through its Communities of Practice [45].

Related patterns

Communities foster inter-team communication, as well as allowing for team members to gather FEEDBACK about their work. This also allows for organisation members to hold each other accountable for their work, fostering ACCOUNTABILITY.

* * *

Pattern: LOW-DEPTH STRUCTURE

The iterative nature of agile software development projects, where the team receives feedback regularly and develops incrementally, is a stark contrast from traditional, plot-driven approaches, where one only receives inputs at the beginning and the end.

Problem

An organisation does not only contain DEVELOPMENT TEAMS, having people from many roles which require different approaches to their work who end up having to work together.

How should an agile organisation be structured?

Forces

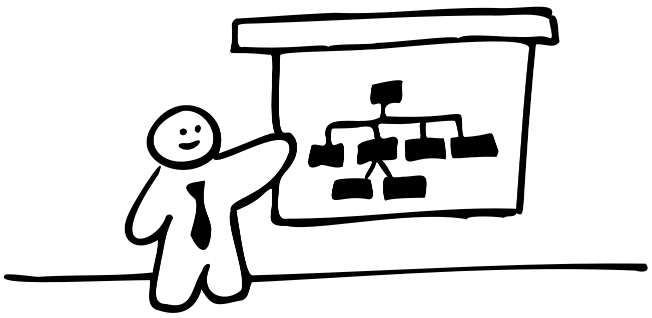
- **Organisation size.** Organisations that use a traditional command-and-control management style acquire more layers and become more complex as they grow in size. This complexity happens due to their vertical information flow, which makes it harder for managers to be able to keep track of everything [31]. On the other hand, organisations that employ a collaborative leadership style give their teams more autonomy require a smaller bureaucratic overhead, as orders do not have to come from the top of the organogram.

- **Direct vs indirect knowledge transfer.** In traditional organisations, knowledge flows indirectly, going from role to role until it reaches its destination. However, one can expect that each person will not retain the information they received in its entirety, losing or changing details as time goes on. Direct knowledge transfer, however, minimises information losses since the recipient interacts directly with the source of information [50].

- **Reacting in the face of changes.** The Agile Manifesto values the ability to react and respond to changes [5]. With traditional, command-and-control management styles, reaction times may be higher compared to collaborative leadership styles, as decisions must come from the top of the organogram instead of the teams just deciding themselves the next course of action.

Solution

The organisational chart should have a low number of layers, encouraging a more direct approach to communication in the organisation.



DEVELOPMENT TEAMS are self-organised, cross-functional entities that are usually able to handle their work on their own [37], having shared leadership and even partaking in COLLABORATIVE DECISION MAKING. The presence of CUSTOMERS in the development process aids this autonomy, as they are able to provide information and FEEDBACK directly to the DEVELOPMENT TEAM. The MANAGEMENT and the COACHES play support roles, performing tasks such as removing obstacles, forecasting needs and facilitating meetings.

These role interactions make it so that decision-making capabilities shift from being concentrated at the top to being spread out throughout the organisation. As such, there is a reduced need for intermediate layers in the organisation chart, and people should feel encouraged to speak directly with whoever may be necessary, depending on the situation.

Examples

Aside from the organisation management, LeSS implements a low number of layers, with a person supervising the teams and the Product Owner(s), who work as peers [45].

Related patterns

This reduction of the complexity in the organisational chart is approached by Coplien and Harrison [20] in the FEW ROLES pattern.

The reduced number of layers encourages the organisation to be more of an AD-HOC ORGANISATION.

* * *

Pattern: TRUST

In agile environments, the projects that are undertaken are often too large and require separation and distribution of tasks. These tasks are divided amongst the team members, who have different experiences in each occupation area. In the end, the work done for the tasks should combine to form a cohesive product.

Problem

The difference in expertise in different areas within agile teams, along with the large size and scope of projects, make it so that team members need to depend on one another to succeed, both in the accomplishment of each person's tasks and in assistance with other ones.

How can team members complement each other and work together towards the goal?

Forces

- **Strengths and weaknesses.** No human is perfect and, as such, each individual on the team has their own strengths and weaknesses, both personal and professional, which create a unique combination of skills in the team.
- **Task distribution.** Agile teams, which are often cross-functional [5][69][45], end up distributing tasks to the team members who feel the most comfortable in working in the task's operational area. The work done for these tasks is then combined to create the product, and everything must fit together.
- **Short iterations.** Agile methodologies employ iterations that are not very long, usually lasting between one to three weeks [69][45]. This makes teamwork a necessity, as only one person could not feasibly do all the work alone with the expected quality within the time-box of an iteration, making it so people need to depend on each other.

Solution

Building trusting relationships between team members allows them to feel more comfortable around each other and to support each other's weaknesses.

Team members should foster a trusting atmosphere within themselves. Being able to trust their colleagues allows team members to not be afraid to ask for help and to recognise one's vulnerabilities [47].

Additionally, a trusting environment increases the quality of the work done as the team is more invested in the project and each member acquires a sense of responsibility, acknowledging their own mistakes and always striving to do a better job [47].

Creating a trusting atmosphere can be done by implementing an OPEN SPACE layout, keeping them as CO-LOCATED TEAMS, ensuring members are present in MEETINGS and even by doing practices such as PAIR PROGRAMMING.

Related Patterns

This pattern touches upon the subject of trust in agile environments, similarly to patterns in other pattern languages: the COMMUNITY OF TRUST, a part of Coplien and Harrison's Organizational Patterns [20], and the CIRCLE OF TRUST, part of the Scrum Patterns [68].

* * *

Pattern: HUMILITY

In agile software development projects, progress happens thanks to the DEVELOPMENT TEAM's teamwork as well as the support from the COACH and the CUSTOMERS.

Problem

The collaborative aspect of agile projects ends up requiring a specific mindset from everyone involved, in a way that fosters trust, cooperation and a proper working environment.

How should people act among their peers and before their team's accomplishments?

Forces

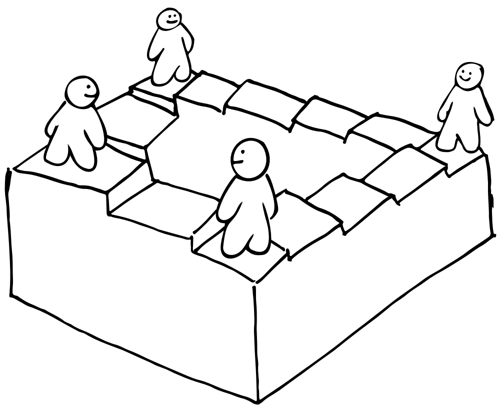
- **Shared accountability.** Team members should share ACCOUNTABILITY regarding their work, as well as holding themselves accountable about the team's performance [47]. This factor stems from the nature of a team, where members establish a relation of interdependence and rely on each other to make things happen [37], as well as agile methodologies advising the existence of collective code ownership [77].

- **Cross-functional teams.** Teams in agile projects tend to be cross-functional, where the developers have different specialities [37]. As such, the members of a cross-functional team are able to take on more extensive tasks.

- **Nobody's perfect.** All organisation workers are human and, as such, with the occasional bad day happening or encountering unforeseen circumstances, there is always the possibility of failure when working on a task.

Solution

People should be humble and think about others as equals instead of holding themselves in higher regard over their peers.



Team members should have a humble behaviour regarding their interactions with others. Agile software development is a collaborative effort and, as such, success happens thanks to the good work that was done by everyone involved.

While it is normal (and even healthy) to be proud of one's accomplishments, since that can be a source of motivation in certain occasions, one must be careful to avoid going overboard with it. People tend to work better with people they have a positive impression of, and being humble also helps with being aware of one's shortcomings and receiving FEEDBACK, which in turn helps with improving one's work.

Related patterns

Humility helps when receiving FEEDBACK and is a key component in the maintenance of ACCOUNTABILITY in the team.

* * *

Pattern: EFFECTIVE LISTENING

Agile software development is a collaborative endeavour, requiring communication and knowledge transfer between all involved parties.

Problem

Face-to-face communication is a powerful method regarding knowledge sharing in agile projects, as people work in close proximity to each other, such as in CO-LOCATED TEAMS. However, while face-to-face communication is rich and full of information, its ephemeral nature may lead to certain things being lost or forgotten over time.

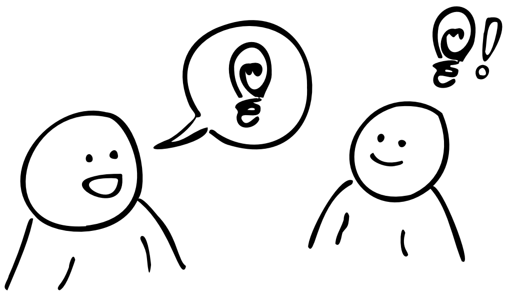
How can we maximise the information we retain from face-to-face communication?

Forces

- **Hearing vs listening.** While both words relate to sound, hearing pertains to the act of receiving sound, and it is done effortlessly and automatically by the brain, which filters out irrelevant sounds. On the other hand, listening is an act that requires focus and voluntary effort, where the brain actively processes the sounds received and decodes them into information [80].
- **Building trust.** People feel more at ease around people who pay attention to them and listen. Building relationships is useful in agile contexts, as people tend to work better when they are comfortable in their social environment.
- **Face-to-face communication.** Face-to-face communication is one of the most common ways of communicating in agile contexts, particularly with CO-LOCATED TEAMS [13]. As such, team members can easily have conversations with each other about anything needing discussing and can perceive, along with the words being said, other cues such as tone of voice, physical gestures and posture.

Solution

Make use of "effective listening" techniques, such as reducing distractions, showing interest, paraphrasing and asking clarifying questions.



The act of reducing distractions, both external and internal, helps in absorbing the information conveyed. Closing doors, moving to quieter areas, and setting aside any work that is nearby are good ways to remove external distractions, while avoiding fleeting thoughts helps in removing internal distractions [49].

Showing interest helps in ensuring the other person perceives the interaction as being a worthwhile investment of their time. Establishing eye contact, adopting an appropriate body posture, such as leaning forward, avoiding note-taking too much, and displaying fitting facial expressions are actions that convey interest in the conversation [49].

Paraphrasing shows the other participant in the conversation that the listener effectively paid attention to what was said, while having the added benefit of serving as a confirmation mechanism, avoiding misinterpretations [49].

Finally, asking clarifying questions, while allowing the speaker to know that the information is being retained by the listener, also allows the latter to know more about the topic at hand [49].

Examples

Aside from the organisation management, LeSS implements a low number of layers, with a person supervising the teams and the Product Owner(s), who work as peers [45].

Related patterns

Team members that listen effectively to their peers tend to find higher levels of TRUST in their relationships.

* * *

Pattern: NEW IDEAS

Software development is a process that requires having to solve problems occasionally, as they may appear every now and then.

Problem

When going about their daily tasks, one may find themselves stumped regarding a problem and not knowing how to proceed in that situation.

How can we come up with new solutions to problems?

Forces

- **Feeling at ease.** It is not uncommon to find information about one topic divided among team members. Having an environment where people can feel at ease and TRUST each other reduces friction when asking for help.

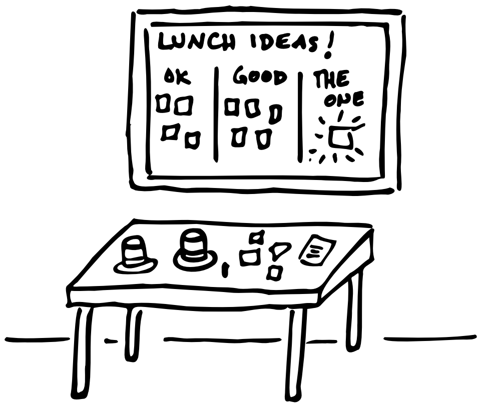
- **Quantity and quality.** There are often trade-offs between quantity and quality; focusing efforts on finding only just a few high-quality things is very different than focusing on finding a high number of things, regardless of their quality.

Solution

With the help of another person or a group, employ methods that generate new ideas, such as dialogue, brainstorming, Nominal Group Technique (NGT) or the Six Thinking Hats.

Engaging in dialogue is useful when interacting with only one other person. Both parties should take turns speaking, bouncing off ideas with each other and building on what is being said. It is of note that the person asking for help should listen closely, ask questions, act impersonally and ask for opinions when proposing ideas [49].

The other methods are group-focused. Brainstorming consists of generating ideas focusing on quantity over quality, welcoming even the most farfetched ideas since this technique works best if all participants feel welcomed. Participants can also combine and improve existing ideas [49].



If there are people who do not feel comfortable with sharing their ideas publicly, making use of NGT is an option. With this technique, participants can write down their ideas and give them to the facilitator, who will then read them out loud. The participants rate the ideas before handing their ratings to the facilitator. This process ensures the author's anonymity. After the ideas are ranked, they are free to be discussed [49].

Another method of coming up with new ideas and conducting a productive discussion is through the use of the Six Thinking Hats. These hats describe attitudes to adopt during the discussion, as one can shift their perspective just like one can switch the hat they are wearing. Each hat is identified by a colour, which identifies the perspective to have: the white hat calls for an objective and neutral view; the red hat focuses on emotions; the black hat provides a cautious, more pessimistic view; the yellow hat focuses on optimism and the good side of things; the green hat provides an outlet for generating new ideas; the blue hat focuses on the overall process and the use of the other hats. The discussions are then structured around the usage of the six hats, as they make it easier for participants to engage in a way that is more beneficial to the current status [24].

Related patterns

The methods here describe benefit from good EFFECTIVE LISTENING skills.

* * *

Pattern: COMMITMENT

Agile software development projects involve making decisions and performing several tasks throughout their courses.

Problem

The DEVELOPMENT TEAM may face having to make hard decisions or having to deal with tough tasks.

How should team members behave before difficult situations?

Forces

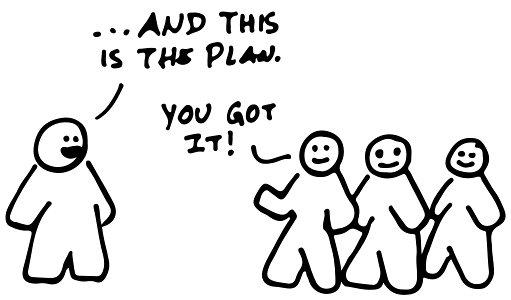
- **Risk and uncertainty.** There are situations where it is impossible for the team to make a perfect decision and that the team members have to deal with uncertainty or other factors that are impossible to know. A team that is afraid of risk and unwilling to deal with the unknown may find itself paralysed in situations where taking action is needed [47].

- **Reaching consensus.** When faced with a situation where a decision is needed, the team members may not all agree on what their course of action should be, and they may not be able to reach a consensus [47].

- **Continuous improvement.** Agile methodologies encourage people to learn from their work, reflecting on what works and what does not, improving their performance as time goes on and the project progresses [5].

Solution

The team should strive for clear, well-defined decisions, which everyone should accept and support.



Not all human beings are alike and, as such, they may often disagree when facing tough decisions. Even when the team cannot reach a consensus, everyone should understand that everyone's views are valid and they should buy into whichever course of action they chose, as a team works together as a single unit [47].

These decisions should also be clear, and the team members should easily understand their priorities. If these decisions should prove to fail, the team is then able to learn from their mistakes and move forward [47].

Related patterns

Even when consensus is not possible, the team should employ COLLABORATIVE DECISION MAKING techniques, ensuring they take everyone's opinion into account.

* * *

Pattern: ACCOUNTABILITY

Agile projects require collaboration and cooperation from all parties involved [5], whether they are part of the DEVELOPMENT TEAM, the MANAGEMENT or even the CUSTOMERS.

Problem

Given that there are people from different backgrounds and realities working together, their approaches to their work may vary, which in turn makes ensuring quality complicated.

How should the team behave in order to ensure quality?

Forces

- **Teams as single units.** Teams are groups of people that work together sharing leadership, establishing relationships of interdependence. They work and make decisions together, measuring their results by assessing what the team made as a collective, instead of focusing on the individuals [37].
- **Usefulness of feedback.** People are not always aware of both their strong and weak points. As such, providing FEEDBACK on their colleagues' work may help in them enhancing their overall performance, as they try to develop their improvement points, in addition to keeping what works.
- **Importance of trust.** TRUST is essential in a team's working environment as it allows people to feel comfortable regarding showing their vulnerabilities since there is a belief that everyone believes in each other's good intentions. This comfort propels people to focus on the task at hand, and one can find that people believe in the team and their work if there is a trusting environment [47].

Solution

Team members should hold their colleagues responsible for their work, without any fear of having difficult conversations about it.

HEY, I FEEL YOU HAD SOME TROUBLE WITH THIS TASK. DO YOU WANT SOME POINTERS?

It is normal for team members to build relationships with each other as they get to know more about each other, spend time together and overcome any adversities their work might lay down in their path. When a team member is not working as well as one would expect, their colleagues should speak up regardless of how hard it may be, as these situations may provide valuable FEEDBACK for the person in question.

Contrastingly, bottling it up and not saying anything ends up deteriorating the people's relationships, as they start holding grudges and trusting each other less and less [47].

Ensuring the team is on the same page regarding what standards to follow is of high importance, as the team members know what they are supposed to do and how they should behave. Giving FEEDBACK is a valuable tool, as it improves quality. Finally, if there are any rewards awarded, they should be given to the team as a whole instead of singling out individual team members, as it encourages team members to keep each other in check about their performance [47].

Related patterns

This concept relies heavily on the TRUST and HUMILITY team members have for it to work. Additionally, REVIEW MEETINGS and RETROSPECTIVE MEETINGS benefit from team members holding each other accountable.

* * *

Pattern: FEEDBACK

When working with agile processes, one may find these focus on continuous improvement. As such, team members find themselves in several situations that aim to improve themselves and their work methods.

Problem

Although the members of the DEVELOPMENT TEAM may want to improve, there may be some uncertainty regarding how to start or which areas may need work.

How can people improve their work?

Forces

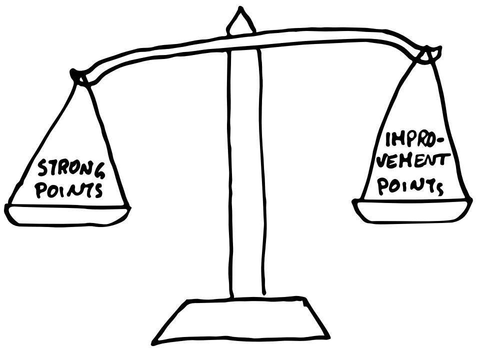
- **Trust within the team.** It is vital for TRUST to exist between the members of the DEVELOPMENT TEAM. Having to acknowledge one's shortcomings and flaws may put them in a vulnerable position, and their colleagues should be supportive [47].

- **Shared accountability.** Team members share ACCOUNTABILITY for each other's work [47]. By being able to help each other out with improving one another, they are indirectly improving the quality of the team's overall work, which is something team members should strive to do.

- **Views from the outside.** Some things are better noticed when we are not too familiar or too close to the subject in question. An example of this may happen when coding when one has spent too long looking for a bug and thus needs a second opinion to try to find it or to just take a break and come back afterwards. When dealing with someone's improvement points, some are better noticed by others, since they have a view that is different from one's own.

Solution

The members of the DEVELOPMENT TEAM should provide FEEDBACK to each other, by pointing out each other's strong points, as well as any potential improvement points that can be corrected.



There are several models to provide feedback, such as the BET/BEAR model, which provides a set of guidelines to follow depending on what is to be highlighted: BET stands for "behaviour, effect, and thanks" and should be used to heighten positive behaviour. One starts by stating the behaviour in question and its effect, and then by thanking the recipient. On the other hand, BEAR stands for "behaviour, effect, alternative, and result" and should be used to highlight improvement points. One starts by stating the behaviour in question and its effect, and then by suggesting an alternative path and the expected results of the alternative suggested [28].

The usage of models such as the BET/BEAR one helps in directing the recipient's focus on the act of improvement itself rather than the existence of the improvement points. If properly delivered and addressed, feedback can increase one's bond to the team and the organisation [28].

Related patterns

Feedback is often employed in both DAILY MEETINGS, REVIEW MEETINGS and RETROSPECTIVE MEETINGS, as well as helping with coming up with NEW IDEAS and being related to the team members' ACCOUNTABILITY and HUMILITY.

* * *

Pattern: Goal Setting

Given their iterative and collaborative nature, where several people work together to make a specific product reality, one can assume that agile projects are extensive undertakings.

Problem

When dealing with a considerably-sized project, it is important to be aware of the tasks, goals and other objectives that will have to be accomplished.

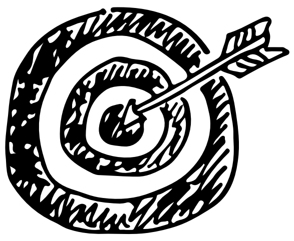
How can we divide the project into attainable objectives?

Forces

- **Sense of direction.** Knowing what to do helps us stay productive [48] and feel less lost, and having a set of tasks should allow us to direct our energy towards completing them, which in turn brings us closer to our goal.
- **Motivation.** Having tasks laid out for the team members helps them stay motivated in two ways. The first one deals with not feeling overwhelmed with which tasks will help or hinder the team's progress, reducing any fears of failure. The second one is the feeling of accomplishment one gets when they manage to complete a task. These two factors help in keeping team members productive and motivated.
- **Increased teamwork.** Being sure about which tasks are to be done helps a lot with a team's coordination, as they will have the same goals in mind and progress will happen more easily. This is similar to having a team push a boulder to a specific location [16]; if everyone knows where to go, the forces applied to the boulder will make it so that the team arrives at their destination more quickly, which does not happen if there are team members who are pushing the boulder the wrong way.

Solution

A project can be divided into goals, ensuring their specificity, time attribution, and clear end conditions.



When dealing with large undertakings, it is important to be able to establish a course of action regarding what needs to be done, dividing the overall objective into smaller goals.

These goals should be SMART in nature: specific, measurable, assignable, realistic and time-bound [25]. This provides goals with a clear-cut purpose that should be fulfilled within a specific timeframe, with a distinct end-condition. Finally, being able to assign people to a goal provides them with some ownership of their accomplishments.

Examples

Scrum's sprints focus on the idea of the Sprint Goal, which is to be fulfilled by the items in the Sprint Backlog [69].

Related patterns

DAILY MEETINGS, PLANNING MEETINGS and RETROSPECTIVE MEETINGS help with maintaining the overall project's goals. These are supported by any BACKLOG REFINEMENT done during the iterations.

Chapter 5

Validation

This chapter approaches the process of validation of the work performed, discussing the methodology, the recipients of the process and the results.

5.1 Goals

The information present in the pattern language presented in chapter 4 came mostly from a literature review, without any interviews performed directly to industry professionals. As such, in order to provide validity to the tasks performed, specialists in the field were contacted after-the-fact to shed some light on whether the patterns documented are in agreement with the industry reality.

This contribution by the professionals happened through a survey, which has the goal to determine answers to the following research questions:

- **RQ1: How can an organisation adapt their environment to better suit its members?**
- **RQ2: Which skills are the most important to have in an agile setting?**
- **RQ3: Are organisations adopting the practices and concepts documented in the pattern language?**

5.2 Methodology and design

The survey was directed at professionals in the industry, more specifically at agile coaches either working in organisations or as freelancers. The reasons for choosing this demographic relate to the awareness coaches have regarding their organisation, along with a higher consciousness of the concepts described in the pattern language.

Due to time restraints and the inability to visit industry organisations due to the COVID-19 pandemic, the professionals were surveyed through the use of a questionnaire. This method was chosen as, unlike interviews, questionnaires do not require synchronous communication, allowing for the person being surveyed to fill them when they find the time, at their own pace.

The questionnaire (reproduced in appendix C) was designed to be simple and easy to answer, taking little time from the questionees. A first section includes questions about the person's organisation, approaching topics such as the size, number of teams, and percentage of teams that employ agile methods, among others. The questions in this section are not mandatory, as the person answering the questionnaire could not be allowed to discuss certain aspects. The following sections are composed of questions inquiring the questionee about the implementation of each pattern in their organisation. These questions are mandatory and include the pattern's patlet (see appendix A) to refresh one's memory about the contents of the patterns. Most of these questions have four possible answers: a first, positive one, stating that the pattern is applied in the entire organisation or in its full capacity; a second, also positive one, referring to a partial implementation; a third, negative one, stating the pattern is not applied; and a final one, used whenever the questionee does not know or the question does not apply. The exceptions to this four-option structure are the questions relating to role patterns which, along with the other options, have two negative answers instead of just one. One of these choices states that the role does not exist, while the other affirms the role exists but not as described in the pattern.

To ensure its accessibility, the questionnaire was created using Google Forms and included links to a web-version of the pattern language. The form was then sent directly to several agile coaches working in the industry via e-mail after requesting their collaboration in the survey.

5.3 Results

Out of the six specialists contacted, four were able to fill out the survey (referred to as R1, R2, R3 and R4). Their answers to the first part of the questionnaire are present in table D.1, while the answers to the other sections are in table D.2.

All four respondents work in different types of organisations; R1 is part of a large organisation where a large majority of the teams employ agile methods. The second respondent, R2, works in a medium-sized organisation where all teams operate using agile methods. Despite R3's organisation having a large number of development teams and employees, it has a low percentage of agile teams. Finally, R4 works with a very small organisation, where every team is agile.

Despite the respondents' backgrounds being different, their answers still managed to provide answers to the research questions mentioned in section 5.1. Starting with **RQ1**, we can learn that digital environment patterns were implemented at a higher level compared to physical environment ones. From this, we can infer that the teams' digital environment is easier to modify, as implementation costs are lower and there are not as many possible restrictions as with the physical environment.

The respondents show us that all skills mentioned by the pattern language are important; however, as an answer to **RQ2**, we can see that COMMITMENT was the skill with the highest implementation rate, followed by TRUST. These skills are highly valuable for good teamwork to happen, particularly in contexts where high change is a factor.

Examining the overall answers to the questionnaire allows us to answer to **RQ3**. Table 5.1 shows us the percentage of each type of reply from each respondent, as well as the average of the four. As we can see, the average percentage of positive answers is over 95%, with over half of that being from fully implemented patterns. From this, we can conclude that the pattern language rings true to the industry reality and that organisations are adopting the documented concepts and practices.

Table 5.1: Survey answer percentages.

Percentage	**R1**	**R2**	**R3**	**R4**	**Average**
Full implementation	52%	68%	26%	58%	51%
Partial implementation	48%	29%	65%	35%	44%
Negative answers	0%	3%	3%	6%	3%
Unknown/Not Applicable	0%	0%	6%	0%	2%
Total	**100%**	**100%**	**100%**	**100%**	**100%**
Positive (Full + Partial)	**100%**	**97%**	**90%**	**94%**	**95%**

The answers to the survey also allow us to derive other conclusions. Looking at table 5.1, we see that R3's organisation, which has a large size but a small portion of agile teams, has the lowest percentage of patterns implemented in their full capacity. The higher number of teams using traditional software development methods may indicate that the techniques, meetings and even the skill-set closely associated with agile teams are not as present compared to the other, more agile organisations.

Despite R4 being from an organisation where all teams are agile, there are a few patterns not being implemented, such as COMMUNITIES. These absences may be related to the smaller size of the organisation, where information can flow more easily without additional components being added to the organisation.

Finally, table D.2 shows us that PAIR PROGRAMMING is the pattern with the lowest presence in the respondents' organisations. Agile teams tend to be cross-functional and, as such, developers may not feel like they would derive much value from working together at the same workspace, despite being a great way to increase trust within the team and to share knowledge.

5.4 Threats to validity

Despite all of the author's efforts when conducting the survey, some factors may have influenced the questionnaire's outcomes. Firstly, the number of replies could always be higher; although the questionnaire yielded inputs from professionals from different organisations, having more respondents would mitigate the influence of any potential outliers existing in the replies received. The questionnaire consisted mostly of multiple-choice questions, which could have made it hard for the respondents to express a specific idea.

The survey's subject itself is a factor to keep in mind, as not everyone is fully aware of the nature of patterns and how they work. As such, one could have always misunderstood something when reading the pattern language, influencing their reply in the questionnaire. Finally, the skill-set questions deal with something that may not be fully perceptible from an outsider's view, potentially making it hard for the respondent to be sure about the prevalence of a specific skill in their organisation.

5.5 Additional validation

The pattern language, along with the physical environment patterns, were presented at EuroPLoP 2020, an online conference that focuses on patterns [32]. The author participated in a writer's workshop where other participants provided feedback to the document, adding another source of validity for the work done.

The workshop participants praised the pertinence of the topic, the clear pattern form, the illustrations and the pattern organisation. They also suggested improvement points to the submitted paper in regard to adding mentions to specific concepts, other patterns, forces and more examples to the patterns, making the overview diagram easier to read, and general wording suggestions.

Chapter 6

Conclusions and Future Work

In this chapter, we find concluding remarks about this document and the work done, while also discussing any difficulties encountered and potential future work. Finally, this chapter ends with a personal reflection regarding the author's journey and growth during the process of writing this dissertation.

6.1 Concluding remarks

The agile manifesto states that team members should cooperate and communicate with stakeholders throughout the project. Effective communication happens when the recipients receive the message clearly without issues. As such, to reduce the chance of mistakes and misunderstandings and to foster a better work environment, this communication should be done as effectively as possible.

A literature review shed some light on several different areas, starting with communication. The communication skill-set contains several skills useful in the workplace: verbal, non-verbal and virtual communication, listening, feedback, persuasion, developing new ideas, overcoming barriers to communication, reducing causes of poor communication, overcoming team dysfunctions, meetings and teamwork. The author researched agile methodologies, focusing on Scrum, XP and LeSS. Scrum is a methodology whose principal concept is the Scrum Team, composed by a Scrum Master, a Development Team and a Product Owner. The Scrum Team works iteratively during Sprints, and work is outlined in four time-boxed meetings. XP provides focus on other concepts such as information radiators and pair programming. Information radiators are a form of relaying information, usually using walls in common areas, where team members can find information passively by looking at them. Pair programming involves two people sitting at the same computer, one interacting with the workstation and the other one providing guidance. This practice provides opportunities to share knowledge and improve team building. LeSS is based on Scrum but applied to a larger scale and, although there is only one Sprint, one Product Owner and one

Product Backlog, each team works on their own backlog, collaborating with other teams if necessary. The conjunction between communication and agile software development was explored. The articles and books found mention several recurring tactics that improve communication within the team, including pair programming, open space offices, information radiators, effective meetings, frequent face-to-face communication, accessible information, cross-functional and self-organised teams, direct interaction between developers and clients, and the ability to react to change. Alistair Cockburn's book, "Agile Software Development: The Cooperative Game", shed some light on how individuals behave and stay motivated, along with approaching how teams communicate and which factors influence said communication. The topic of pattern theory was also approached. Patterns are easily-digestible pieces of information that include common and proven solutions to recurring problems. All patterns have a title, a context, a problem, a solution and forces that drive its choice as a pattern, but some may have other additional fields. Finally, a review of three related pattern languages provided some insight into how patterns work, as well as some additional useful knowledge.

The problem this dissertation attempted to solve, asking how a person could communicate effectively in agile environments, was tackled through the writing of a pattern language. This set of thirty-one patterns and proto-patterns approaches several topics such as the environment, the activities the people engage in, the individuals themselves, and how they behave and interact with each other.

To validate the work present in the pattern language, the author surveyed industry professionals regarding the adoption of the patterns in their organisations. The questionnaire results show that the patterns were mostly implemented in their full capacity, while a good percentage indicated partial implementations. These were favourable results, in spite of having a small sample size. The pattern language was also presented at EuroPLoP 2020, an online conference focused on pattern languages. The author's participation allowed him to receive feedback about the pattern language as a whole and the physical environment patterns, functioning as an additional validation technique.

As such, we can summarise the key contributions of this work as the following items:

- Literature review on communication, agile methods, pattern theory and related pattern languages;
- Pattern language containing thirty one patterns and proto-patterns relating to effective communication;
- Survey of industry professionals;
- Presentation of the pattern language at a patterns conference.

6.2 Difficulties encountered

The author ran into some difficulties and unexpected obstacles when working on this dissertation, mostly due to the COVID-19 pandemic. The initial work plan included visiting agile organisations and interacting directly with industry professionals to acquire knowledge to document in the patterns, which became inviable due to the shift to remote work and social distancing. The lockdown and the switch to remote work also impacted the author's motivation and productivity.

6.3 Future work

The work conducted in this dissertation can still be further developed in a few areas. Starting with the validation, we can see that the survey could be replicated with a larger sample size to see how the results found so far would be affected. One could also conduct interviews with professionals to potentially bridge any gaps that may have resulted from the literature review

The pattern language itself could and will be revised facing these new interactions with industry professionals, as well as being published on its own as a booklet.

6.4 Personal reflection

This work allowed me to learn a lot about something I had only known about superficially: patterns. I had the chance to learn a lot more about what goes into documenting a pattern, including the processes of pattern mining and writing. Working on a pattern language also allowed me to start getting involved in the scientific community, and participating in EuroPLoP 2020 was a unique experience I am happy to have had.

Although the patterns were the centrepiece of my dissertation, I was also able to learn a lot about different concepts related to communication and deepen my understanding of agile methodologies.

Finally, I had the pleasure of working with some of the most wonderful people I got to know, and I feel like I have grown as a person thanks to this experience.

Appendix A

Patlets

Table A.1: Pattern language patlets

Pattern name	Patlet
INFORMATION RADIATORS	Visual aids such as whiteboards, flipcharts and similar items with useful information help ensuring everyone is up to speed when placed in common areas or in places where people frequently walk by.
CO-LOCATED TEAM	Having team members work in close proximity to each over, in the same room, adjacent rooms or at least on the same floor helps communication to be nearly effortless within the team.
OPEN SPACE	Organising the team's furniture to remove physical barriers between team members and include everyone in the working atmosphere helps reducing friction and entropy.
INFORMAL COMMUNICATION SPACE	Areas far enough from workstations but close enough where people are in the vicinity often are good spaces to dedicate to informal communication.
CUBES	Acoustically-sound rooms or spaces are places that create good conditions for conference calls.
CALLS	A group of people who are far apart from each other use a phone, a computer or any similar device to participate in a call, allowing them to interact synchronously.
INSTANT MESSAGING	Using a computer or another internet-accessible device, a person can send short text messages that are sent instantly to their recipient through the use of an IM service.
E-MAILS	E-mails can be used to address team members and provide them with pondered, substantial information that can be archived and referred to later on.
OPEN INFORMATION TOOLSET	Web-based tools where users can easily search for and add new information provide a good platform for information archival and knowledge sharing.
DAILY MEETINGS	The Development Team can stay updated about the project by meeting at the start of each day and discussing their progress on the project, their current tasks and any problems they might be facing.

Table A.1 continued from previous page

Name	**Patlet**
PLANNING MEETINGS	Before an iteration starts, the Development Team and the Customers should decide which tasks should be done during the iteration, establishing a clear course of action.
REVIEW MEETINGS	At the end of an iteration, the Development Team should meet with the Customers and other stakeholders to present the increment built during said iteration, evaluating it and reflecting on the work done.
RETROSPECTIVE MEETINGS	In order to grow, the Development Team should meet with the Coach between the end of an iteration and the beginning of the next one somewhere quiet and reflect on the team's work and on how to improve it.
PAIR PROGRAMMING	In order to make knowledge distribution easier, development team members should pair up and work together at the same workstation, while one codes and the other provides support.
BACKLOG REFINEMENT	The team's adaptability is aided by periodically assessing the backlog, modifying existing items depending on their relevance, and adding new ones if necessary.
COLLABORATIVE DECISION MAKING	Decisions should be made through the use of collaborative decision-making methods, such as finding consensus, trying to reach a compromise, or applying a voting method, in order to make decisions.
COACH	The Coach supports the development team and the management in the adoption and learning process of agile methodologies, along with the removal of obstacles, and facilitation of meetings.
DEVELOPMENT TEAM	The developers should organise themselves into teams, entities where people work together and support each others' activities, share leadership, hold each other accountable, share a strong bond, and work towards a common goal.
CUSTOMERS	The Customers should have a close working relationship with the Development Team, providing feedback and clearing up any questions they may have.
MANAGEMENT	When dealing with agile projects, the management should support their teams by forecasting needs and removing both internal and external obstacles.
AD-HOC ORGANISATION	The organisation members should feel free to interact with any other person if the need arises, regardless of rank or area of operation.
COMMUNITIES	Communities are groups that exist parallel to the organisational chart, allowing people to exchange knowledge between teams, increase inter-team collaboration and build a sense of unity within the organisation.
LOW-DEPTH STRUCTURE	The organisational chart should be structured in order to have a low number of layers, encouraging a more direct approach to communication and decision-making in the organisation.
TRUST	Building trusting relationships between team members allows them to feel more comfortable around each other and to support each other's weaknesses, permitting them to complement each other.

Table A.1 continued from previous page

Name	**Patlet**
HUMILITY	Team members should be humble and think about others as equals instead of holding themselves in higher regard over their peers, even when feeling accomplished.
EFFECTIVE LISTENING	Effective listening techniques, such as reducing distractions, expressing interest, asking clarifying questions and paraphrasing are useful to maximise information retained from face-to-face communication.
NEW IDEAS	Generating new ideas be done through techniques such as dialogue, brainstorming, Nominal Group Technique (NGT) or the Six Thinking Hats.
COMMITMENT	Even in the most difficult of situations, team members should establish clear courses of action and buy into them.
ACCOUNTABILITY	Team members should hold each other responsible for their work by establishing high standards and not being able to give feedback, regardless of how hard it may be.
FEEDBACK	Team members should provide feedback to each other, by pointing out each other's strong points, as well as any potential improvement points that can be corrected.
GOAL SETTING	Dividing a project into SMART goals can ensure the resulting objectives are specific, time-bound and have clear end conditions.

Appendix B

Pattern Problem-Solution Relationships

Table B.1: Environment patterns: Physical

Pattern	Problem	Solution
INFORMATION RADIATORS	How can we ensure everyone is up to speed regarding the live progress of the project?	Include visual aids, such as boards, flipcharts and similar items with useful information in common areas or in places where people frequently walk by.
CO-LOCATED TEAM	How can we ensure that communication is nearly effortless within the team?	The team members should work in close proximity to each over, in the same room, adjacent rooms or at least on the same floor.
OPEN SPACE	How should a CO-LOCATED TEAM organise its space to reduce friction and entropy?	The team should organise their furniture to remove physical barriers between team members, including everyone in the working atmosphere.
INFORMAL COMMUNICATION SPACE	What is the best environment for informal communication?	Spaces that are far enough from workstations to avoid disturbing people who are working but close enough where people are in the vicinity often are good spaces to dedicate to informal communication.
CUBES	How can we create good conditions for CALLS?	Using acoustically-sound rooms or spaces, that can fit either a couple of people or a large group, provide good conditions for calls and meetings.

Table B.2: Environment patterns: Digital

Pattern	Problem	Solution
CALLS	How can we contact with a person who is far away in a synchronous manner?	Two (or more) people far apart from each other can call each other using a phone, a computer or any similar device to have a synchronous conversation.
INSTANT MESSAGING	How can one reach another person with quick information?	Using a computer or another internet-able device, a person can send short text messages that are sent instantly to their recipient.
E-MAILS	How can we send information that is lasting to team members, regardless of distance?	Use e-mails to address team members, which provide pondered, substantial information and that can be archived and referred to later on.
OPEN INFORMATION TOOLSET	How should the project personnel archive any written documentation and other records they produce throughout the project?	Web-based tools where users can easily search for and add new information provide a good platform for information archival and knowledge sharing.

Table B.3: Activity patterns: Meetings

Pattern	Problem	Solution
DAILY MEETINGS	How can we keep everyone on the team updated regarding the current status of the project?	Every day, the DEVELOPMENT TEAM should meet and discuss their progress on the project, their current tasks and any problems they might be facing.
PLANNING MEETINGS	How can we lay a proper foundation to the next iteration's work?	Before an iteration starts, the DEVELOPMENT TEAM and the CUSTOMERS should decide which tasks should be done during the iteration.
REVIEW MEETINGS	How should the team evaluate and reflect on the work done during the iteration?	At the end of an iteration, the team should assess the work done and evaluate the task backlog, timeline and budget for the next iterations.
RETROSPECTIVE MEETINGS	How should the Development Team reflect on their work in order to grow?	Between the end of an iteration and the beginning of the next one, the DEVELOPMENT TEAM should meet with the COACH somewhere quiet and reflect on the team's work and on how to improve it.

Table B.4: Activity patterns: Practices

Pattern	**Problem**	**Solution**
PAIR PROGRAMMING	How can the Development Team better distribute its knowledge?	DEVELOPMENT TEAM members should pair up and work together at the same workstation, while one codes and the other provides support, sharing knowledge and building a working relationship.
BACKLOG REFINEMENT	How can we stay adaptable and keep up with the needs of an evolving project?	The DEVELOPMENT TEAM, along with the USERS/-CUSTOMERS, should assess the backlog, remove or sort existing items depending on their relevance, and add new ones if necessary.
COLLABORATIVE DECISION MAKING	How should decisions be made?	The DEVELOPMENT TEAM should make use of collaborative decision-making methods, such as finding consensus, trying to reach a compromise, or applying a voting method, in order to make decisions.

Table B.5: People patterns: Roles

Pattern	**Problem**	**Solution**
COACH	How can we support the team and the clients in the adoption of agile methodologies?	The COACH provides a support role to the DEVELOPMENT TEAM from within the organisation, assisting them and the MANAGEMENT in the adoption and learning process of agile methodologies, removing obstacles, and facilitating meetings.
DEVELOPMENT TEAM	How should the developers organise themselves to work?	The developers should organise themselves into a DEVELOPMENT TEAM, where they work together and support each others' activities, share leadership, hold each other accountable, share a strong bond, and work towards a common goal.
CUSTOMERS	How should the project's recipients interact with the organisation?	The CUSTOMERS should provide timely feedback to the DEVELOPMENT TEAM, clearing any questions they may have.
MANAGEMENT	How should the behaviour of the organisation's management be facing agile projects?	The management should support their teams, forecasting needs, removing external obstacles, and working with the COACHES to remove internal ones.

Table B.6: People patterns: Organisational Structure

Pattern	Problem	Solution
AD-HOC ORGANISATION	How should the working relationships of the organisation personnel be like?	The organisation members should feel free to interact with any other personnel if the need arises, regardless of rank or area of operation.
COMMUNITIES	How can we foster cross-team communication and knowledge sharing?	Communities are groups that exist parallel to the organisational chart and allow organisation members to exchange knowledge between teams, increase inter-team collaboration and build a sense of unity within the organisation.
LOW-DEPTH STRUCTURE	How should an agile organisation be structured?	The organisational chart should have a low number of layers, encouraging a more direct approach to communication in the organisation.

Table B.7: People patterns: Skills

Pattern	Problem	Solution
TRUST	How can team members complement each other and work together towards the goal?	Building trusting relationships between team members allows them to feel more comfortable around each other and to support each other's weaknesses.
HUMILITY	How should people act among their peers and before their team's accomplishments?	People should be humble and think about others as equals, instead of holding themselves in higher regard over their peers.
EFFECTIVE LISTENING	How can we maximise the information we retain from face-to-face communication?	Make use of "effective listening" techniques, such as reducing distractions, showing interest, paraphrasing and asking clarifying questions.
NEW IDEAS	How can we come up with new solutions to problems?	With the help of another person or a group, employ methods that generate new ideas, such as dialogue, brainstorming, Nominal Group Technique (NGT) or the Six Thinking Hats.
COMMITMENT	How should team members behave before difficult situations?	The team should strive for clear, well-defined decisions, which everyone should accept and support.
ACCOUNTABILITY	How should the team behave in order to ensure quality?	Team members should hold their colleagues responsible for their work, without any fear of having difficult conversations about it.
FEEDBACK	How can people improve their work?	Team members should use feedback to point out positive points as well as improvement points to their colleagues.
GOAL SETTING	How can we divide the project into attainable objectives?	A project can be divided into SMART goals, ensuring their specificity, time-attribution, and clear end conditions.

Appendix C

Survey Questionnaire

This appendix contains the questions present in the survey conducted for the validation of the work performed (see chapter 5).

Section 1: Organisational environment

These questions allow us to categorise your organisation in terms of size, agility, and project distribution, which will be used to attempt to find correlations with the presence of patterns.
These questions are not mandatory (due to the potential existence of NDAs and trade secrets), but answering them is highly appreciated.

What is the dimension of your organisation's personnel?

◯ 20 people or under
◯ 21 - 50 people
◯ 51 - 100 people
◯ 101 - 250 people
◯ 251 - 500 people
◯ 501 - 1000 people
◯ Over 1000 people

How many development teams are working in your organisation?

◯ Just one
◯ 2 - 5 teams
◯ 6 - 10 teams
◯ 11 - 20 teams
◯ Over 20 teams

What is the estimated percentage of teams that are working using agile methods?

0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

Which agile methodologies are employed in your organisation?

Please select all that apply. If your organisation employs a hybrid of these, please tick the "Hybrid" box and the methodologies that build said hybrid. If your organisation uses its own methodology that was developed in-house, please use the "Other" field.

◯ Scrum
◯ Extreme Programming (XP)
◯ Crystal
◯ Kanban
◯ Large Scale Scrum (LeSS)
◯ Scrum at Scale (S@S)
◯ Scaled Agile Framework (SAFe)
◯ Lean Startup
◯ Scrum of Scrums
◯ Hybrid
◯ Other: ____________________

How are projects distributed among development teams?

◯ All teams are working on the same project
◯ All teams share projects, but there are several projects
◯ Some teams are working alone on projects while others collaborate on theirs
◯ Each team works alone on their project

Section 2: Environment Patterns

These patterns relate to the factors, both physical and digital, that exist in the environment where people are situated and, therefore, influence how they behave. These questions are mandatory.

Physical Environment Patterns

These patterns express tangible, physical elements that enhance communication in agile contexts, attempting to remove barriers and reduce the face-to-face communication cost.
For this set of patterns, please state if the following patterns are implemented in your organisation.

[Physical] Information Radiators

Visual aids such as whiteboards, flipcharts and similar items with useful information help ensuring everyone is up to speed when placed in common areas or in places where people frequently walk by.

◯ Yes, this pattern is implemented in every team.
◯ Yes, this pattern is implemented in some teams.
◯ No, this pattern is not implemented in any teams.
◯ Unknown/Not applicable

[Physical] Co-Located Team

Having team members work in close proximity to each over, in the same room, adjacent rooms or at least on the same floor helps communication to be nearly effortless within the team.

◯ Yes, this pattern is implemented in every team.
◯ Yes, this pattern is implemented in some teams.
◯ No, this pattern is not implemented in any teams.
◯ Unknown/Not applicable

[Physical] Open Space

Organising the team's furniture to remove physical barriers between team members and include everyone in the working atmosphere helps reducing friction and entropy.

◯ Yes, this pattern is implemented in every team.
◯ Yes, this pattern is implemented in some teams.
◯ No, this pattern is not implemented in any teams.
◯ Unknown/Not applicable

[Physical] Informal Communication Space

Areas far enough from workstations but close enough where people are in the vicinity often are good spaces to dedicate to informal communication.

◯ Yes, this pattern is implemented in every team.
◯ Yes, this pattern is implemented in some teams.
◯ No, this pattern is not implemented in any teams.
◯ Unknown/Not applicable

[Physical] Cubes

Acoustically-sound rooms or spaces are places that create good conditions for conference calls or meetings.

◯ Yes, this pattern is implemented in every team.
◯ Yes, this pattern is implemented in some teams.
◯ No, this pattern is not implemented in any teams.
◯ Unknown/Not applicable

Digital Environment Patterns

The digital environment also influences how people communicate in agile contexts as different communication methods have different goals and yield different results.
For this set of patterns, please state if the following patterns are implemented in your organisation.

[Digital] Calls

A group of people who are far apart from each other use a phone, a computer or any similar device to participate in a call, allowing them to interact synchronously.

◯ Yes, this pattern is implemented in every team.
◯ Yes, this pattern is implemented in some teams.
◯ No, this pattern is not implemented in any teams.
◯ Unknown/Not applicable

[Digital] Instant Messaging

Using a computer or another internet-accessible device, a person can send short text messages that are sent instantly to their recipient through the use of an IM service.

◯ Yes, this pattern is implemented in every team.
◯ Yes, this pattern is implemented in some teams.
◯ No, this pattern is not implemented in any teams.
◯ Unknown/Not applicable

[Digital] E-mails

E-mails can be used to address team members and provide them with pondered, substantial information that can be archived and referred to later on.

◯ Yes, this pattern is implemented in every team.
◯ Yes, this pattern is implemented in some teams.
◯ No, this pattern is not implemented in any teams.
◯ Unknown/Not applicable

[Digital] Open Information Toolset

Web-based tools where users can easily search for and add new information provide a good platform for information archival and knowledge sharing.

◯ Yes, this pattern is implemented in every team.
◯ Yes, this pattern is implemented in some teams.
◯ No, this pattern is not implemented in any teams.
◯ Unknown/Not applicable

Section 3: Activity Patterns

These patterns relate to the activities people engage in during their day-to-day activities, which can be divided into meetings and practices. These questions are mandatory.

Meeting Patterns

Meetings are recurring events where people gather to discuss topics relevant to the organisation's activities. Each meeting has a specific goal in mind.
For this set of patterns, please state if the following patterns are implemented by the teams in your organisation.

[Meeting] Daily Meetings

The Development Team can stay updated about the project by meeting at the start of each day and discussing their progress on the project, their current tasks and any problems they might be facing.

◯ Yes, this pattern is implemented by every team.
◯ Yes, this pattern is implemented by some teams.
◯ No, this pattern is not implemented by any teams.
◯ Unknown/Not applicable

[Meeting] Planning Meetings

Before an iteration starts, the Development Team and the Customers should decide which tasks should be done during the iteration, establishing a clear course of action.

◯ Yes, this pattern is implemented by every team.
◯ Yes, this pattern is implemented by some teams.
◯ No, this pattern is not implemented by any teams.
◯ Unknown/Not applicable

[Meeting] Review Meetings

At the end of an iteration, the Development Team should meet with the Customers and other stakeholders to present the increment built during said iteration, evaluating it and reflecting on the work done.

◯ Yes, this pattern is implemented by every team.
◯ Yes, this pattern is implemented by some teams.
◯ No, this pattern is not implemented by any teams.
◯ Unknown/Not applicable

[Meeting] Retrospective Meetings

In order to grow, the Development Team should meet with the Coach between the end of an iteration and the beginning of the next one somewhere quiet and reflect on the team's work and on how to improve it.

◯ Yes, this pattern is implemented by every team.

◯ Yes, this pattern is implemented by some teams.
◯ No, this pattern is not implemented by any teams.
◯ Unknown/Not applicable

Practice Patterns

Practices are activities that are done during the daily activity of the organisation, with various forms and objectives.
For this set of patterns, please state if the following patterns are implemented by the teams in your organisation.

[Practice] Pair Programming

In order to make knowledge distribution easier, development team members should pair up and work together at the same workstation, while one codes and the other provides support.

◯ Yes, this pattern is implemented by every team.
◯ Yes, this pattern is implemented by some teams.
◯ No, this pattern is not implemented by any teams.
◯ Unknown/Not applicable

[Practice] Backlog Refinement

The team's adaptability is aided by periodically assessing the backlog, modifying existing items depending on their relevance, and adding new ones if necessary.

◯ Yes, this pattern is implemented by every team.
◯ Yes, this pattern is implemented by some teams.
◯ No, this pattern is not implemented by any teams.
◯ Unknown/Not applicable

[Practice] Collaborative Decision Making

Decisions should be made through the use of collaborative decision-making methods, such as finding consensus, trying to reach a compromise, or applying a voting method, in order to make decisions.

◯ Yes, this pattern is implemented by every team.
◯ Yes, this pattern is implemented by some teams.
◯ No, this pattern is not implemented by any teams.
◯ Unknown/Not applicable

Section 4: People Patterns

These patterns relate to people and how they behave as individuals within the organisation, which has some sort of structure. Individuals often have a role within the environment and should possess a set of skills adequate for communicating effectively. These questions are mandatory.

Role Patterns

An individual has a designated role within the organisation. These roles have clearly defined responsibilities and established interactions between themselves.
For this set of patterns, please state if the following roles are implemented in your organisation as described in the patterns.

[Role] Coach

The Coach supports the development team and the management in the adoption and learning process of agile methodologies, along with the removal of obstacles, and facilitation of meetings.

- ◯ Yes, this role exists and the pattern is implemented in its full capacity.
- ◯ Yes, this role exists and the pattern is implemented partially.
- ◯ No, this role exists but the pattern is not implemented as described in any way, shape or form.
- ◯ No, this role does not exist.
- ◯ Unknown/Not applicable

[Role] Development Team

The developers should organise themselves into teams, entities where people work together and support each others' activities, share leadership, hold each other accountable, share a strong bond, and work towards a common goal.

- ◯ Yes, this role exists and the pattern is implemented in its full capacity.
- ◯ Yes, this role exists and the pattern is implemented partially.
- ◯ No, this role exists but the pattern is not implemented as described in any way, shape or form.
- ◯ No, this role does not exist.
- ◯ Unknown/Not applicable

[Role] Customers

The Customers should have a close working relationship with the Development Team, providing feedback and clearing up any questions they may have.

- ◯ Yes, this role exists and the pattern is implemented in its full capacity.
- ◯ Yes, this role exists and the pattern is implemented partially.
- ◯ No, this role exists but the pattern is not implemented as described in any way, shape or form.
- ◯ No, this role does not exist.
- ◯ Unknown/Not applicable

[Role] Management

When dealing with agile projects, the management should support their teams by forecasting needs and removing both internal and external obstacles.

- ◯ Yes, this role exists and the pattern is implemented in its full capacity.

◯ Yes, this role exists and the pattern is implemented partially.
◯ No, this role exists but the pattern is not implemented as described in any way, shape or form.
◯ No, this role does not exist.
◯ Unknown/Not applicable

Organisational Structure Patterns

An organisation needs to have some sort of structure so that its day-to-day activities can happen as they should. These patterns explain several organisation types, as well as detailing some additional characteristics about the interpersonal relationships between personnel.
For this set of patterns, please state if your organisation implements following organisational structure patterns.

[Organisational Structure] Ad-hoc Organisation

The organisation members should feel free to interact with any other person if the need arises, regardless of rank or area of operation.

◯ Yes, this pattern is implemented in its full capacity.
◯ Yes, this pattern is implemented partially.
◯ No, this pattern is not implemented as described in any way, shape or form.
◯ Unknown/Not applicable

[Organisational Structure] Communities

Communities are groups that exist parallel to the organisational chart, allowing people to exchange knowledge between teams, increase inter-team collaboration and build a sense of unity within the organisation.

◯ Yes, this pattern is implemented in its full capacity.
◯ Yes, this pattern is implemented partially.
◯ No, this pattern is not implemented as described in any way, shape or form.
◯ Unknown/Not applicable

[Organisational Structure] Low-Depth

The organisational chart should be structured in order to have a low number of layers, encouraging a more direct approach to communication and decision-making in the organisation.

◯ Yes, this pattern is implemented in its full capacity.
◯ Yes, this pattern is implemented partially.
◯ No, this pattern is not implemented as described in any way, shape or form.
◯ Unknown/Not applicable

Skill Patterns

Organisation members should possess skills that can help in communicating with others; these skills can be related to their behaviour and how they view the world or related to their interactions with others.
For this set of patterns, please state if the people in your organisation employ the following skill patterns.

[Skill] Trust

Building trusting relationships between team members allows them to feel more comfortable around each other and to support each other's weaknesses, permitting them to complement each other.

- ◯ Yes, this pattern is implemented by most, if not all, people.
- ◯ Yes, this pattern is implemented by some people.
- ◯ No, this pattern is not implemented at all by anyone.
- ◯ Unknown/Not applicable

[Skill] Humility

Team members should be humble and think about others as equals instead of holding themselves in higher regard over their peers, even when feeling accomplished.

- ◯ Yes, this pattern is implemented by most, if not all, people.
- ◯ Yes, this pattern is implemented by some people.
- ◯ No, this pattern is not implemented at all by anyone.
- ◯ Unknown/Not applicable

[Skill] Effective Listening

Effective listening techniques, such as reducing distractions, expressing interest, asking clarifying questions and paraphrasing are useful to maximise information retained from face-to-face communication.

- ◯ Yes, this pattern is implemented by most, if not all, people.
- ◯ Yes, this pattern is implemented by some people.
- ◯ No, this pattern is not implemented at all by anyone.
- ◯ Unknown/Not applicable

[Skill] New Ideas

Generating new ideas be done through techniques such as dialogue, brainstorming, Nominal Group Technique (NGT) or the Six Thinking Hats.

- ◯ Yes, this pattern is implemented by most, if not all, people.
- ◯ Yes, this pattern is implemented by some people.
- ◯ No, this pattern is not implemented at all by anyone.
- ◯ Unknown/Not applicable

[Skill] Commitment

Even in the most difficult of situations, team members should establish clear courses of action and buy into them.

◯ Yes, this pattern is implemented by most, if not all, people.
◯ Yes, this pattern is implemented by some people.
◯ No, this pattern is not implemented at all by anyone.
◯ Unknown/Not applicable

[Skill] Accountability

Team members should hold each other responsible for their work by establishing high standards and not being able to give feedback, regardless of how hard it may be.

◯ Yes, this pattern is implemented by most, if not all, people.
◯ Yes, this pattern is implemented by some people.
◯ No, this pattern is not implemented at all by anyone.
◯ Unknown/Not applicable

[Skill] Feedback

Team members should provide feedback to each other, by pointing out each other's strong points, as well as any potential improvement points that can be corrected.

◯ Yes, this pattern is implemented by most, if not all, people.
◯ Yes, this pattern is implemented by some people.
◯ No, this pattern is not implemented at all by anyone.
◯ Unknown/Not applicable

[Skill] Goal Setting

Dividing a project into goals can ensure the resulting objectives are specific, time-bound and have clear end conditions.

◯ Yes, this pattern is implemented by most, if not all, people.
◯ Yes, this pattern is implemented by some people.
◯ No, this pattern is not implemented at all by anyone.
◯ Unknown/Not applicable

Section 5: Final remarks

Do you have any suggestions or closing remarks?

____
____
____

Appendix D

Survey Replies

In table D.2, a black circle denotes the pattern is implemented fully, a white circle denotes the pattern is implemented partially, a cross denotes the pattern is not implemented and "Unknown/Not Applicable" answers are denoted by a question mark.

Table D.1: Survey respondents' organisation data.

Question	**R1**	**R2**	**R3**	**R4**
Dimension of organisation personnel	501 - 1000 people	251 - 500 people	Over 1000 people	21 - 50 people
Number of development teams	Over 20 teams	11 - 20 teams	Over 20 teams	2 - 5 teams
Percentage of teams using agile methods	80%	100%	30%	100%
Agile methodologies employed	Scrum, XP, Kanban, Scrum at Scale, Scrum of Scrums, Hybrid	Scrum, XP, Kanban	Hybrid of Scrum and Kanban	In-house developed methodology
Project distribution among teams	All teams share projects, but there are several projects	Some teams are working alone on projects while others collaborate on theirs	All teams share projects, but there are several projects	Some teams are working alone on projects while others collaborate on theirs

Table D.2: Pattern implementation according to survey respondents.

Patterrn	**R1**	**R2**	**R3**	**R4**
INFORMATION RADIATORS	●	●	○	●
CO-LOCATED TEAM	○	○	?	○
OPEN SPACE	●	○	✗	●
INFORMAL COMMUNICATION SPACE	●	○	○	○
CUBES	●	○	●	○
CALLS	●	●	○	●
INSTANT MESSAGING	●	●	●	●
E-MAILS	●	●	●	●
OPEN INFORMATION TOOLSET	●	●	●	○
DAILY MEETINGS	●	●	○	●
PLANNING MEETINGS	○	●	○	●
REVIEW MEETINGS	○	○	○	●
RETROSPECTIVE MEETINGS	○	○	○	●
PAIR PROGRAMMING	○	○	?	✗
BACKLOG REFINEMENT	○	●	○	●
COLLABORATIVE DECISION MAKING	●	●	○	○
COACH	●	✗	●	●
DEVELOPMENT TEAM	○	●	○	●
CUSTOMERS	○	○	○	○
MANAGEMENT	●	●	○	○
AD-HOC ORGANISATION	●	●	●	○
COMMUNITIES	○	○	●	✗
LOW DEPTH STRUCTURE	●	●	○	●
TRUST	●	●	○	○
HUMILITY	○	●	○	●
EFFECTIVE LISTENING	○	●	○	○
NEW IDEAS	○	●	○	●
COMMITMENT	●	●	●	●
ACCOUNTABILITY	○	●	○	○
FEEDBACK	○	●	○	●
GOAL SETTING	○	●	○	●

References

[1] Kevin Aguanno. *Managing Agile Processes*. Multi-Media Publications Inc., 2005.

[2] Christopher Alexander. The Timeless Way of Building, 1979.

[3] Christopher Alexander, Sara Ishikawa, and Silverstein Murray. *A Pattern Language*. 1977.

[4] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. addison-wesley professional, 2 edition, 2004.

[5] Kent Beck, Mike Beedle, A Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for Agile Software Development, 2001.

[6] Kent Beck and Ward Cunningham. Using Pattern Languages for Object-Oriented Programs. *http://c2.com/doc/oopsla87. html*, 1987.

[7] Arlo Belshee. Promiscuous pairing and beginner's mind: embrace inexperience [agile programming]. In *Agile Development Conference (ADC'05)*, pages 125–131. IEEE Comput. Soc, 2006.

[8] Board of European Students of Technology. BEST - Identity, 2020.

[9] Board of European Students of Technology. Departments of BEST, 2020.

[10] Trinadh Bonam. Using Feedback Loops to Boost Development Lifecycles.

[11] John Seely Brown and Paul Duguid. Organizational Learning and Communities-of-Practice: Toward a Unified View of Working, Learning, and Innovation. *Organization Science*, 2(1):40–57, feb 1991.

[12] The LeSS Company B.V. Large Scale Scrum (LeSS), 2019.

[13] Thomas Chau and Frank Maurer. Knowledge sharing in agile software teams. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3075:173–183, 2004.

[14] Craig Cochran. *The continual improvement process: from strategy to the bottom line*. Paton Professional, 2003.

[15] Alistair Cockburn. Expert In Earshot. *Portland Pattern Repository*, 2005.

[16] Alistair Cockburn. *Agile software development: the cooperative game*, volume 113. 2006.

[17] Alistair Cockburn and Laurie Williams. *The Costs and Benefits of Pair Programming*, pages 223–243. Addison-Wesley Longman Publishing Co., Inc., USA, 2001.

[18] Philip R. Cohen and Hector J. Levesque. Teamwork. *Noûs*, 25(4):487–512, sep 1991.

[19] Melvin E. Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.

[20] James O Coplien and Neil B Harrison. *Organizatinal Patterns of Agile Software Development*. Pearson Prentice Hall Upper Saddle River, 2004.

[21] Frank E. X. Dance. The "Concept" of Communication. *Journal of Communication*, 20(2):201–210, jun 1970.

[22] Marcel Danesi. Metaphor and figurative meaning in verbal communication. In *Handbook of Communication: Verbal Communication*, chapter 8, pages 142–161. De Gruyter, 2016.

[23] Rachel Davies and Liz Sedley. *Agile Coaching*. Pragmatic Bookshelf, 1st edition, 2009.

[24] Edward de Bono. *Six Thinking Hats*. Penguin. Penguin Life, 2017.

[25] George T. Doran. There's a S.M.A.R.T. way to write management's goals and objectives. *Management review*, 70(11):35–36, 1981.

[26] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education India, 1995.

[27] Dave Gray, Sunni Brown, and James Macanufo. *Gamestorming: A playbook for innovators, rulebreakers, and changemakers*. O'Reilly Media, Inc, 2010.

[28] Patricia L. Harms and Deborah Britt Roebuck. Teaching the art and craft of giving and receiving feedback. *Business Communication Quarterly*, 73(4):413–431, 2010.

[29] Tim Hartnett. *Consensus-oriented decision-making: The CODM model for facilitating groups to widespread agreement*. new society publishers, 2011.

[30] James D. Herbsleb and Rebecca E. Grinter. Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software*, 16(5):63–70, 1999.

[31] James A. Highsmith. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, volume 12. Addison-Wesley, 2000.

[32] Hillside Europe. Call for papers EuroPLoP 2020, 2020.

[33] Rashina Hoda, James Noble, and Stuart Marshall. The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology*, 53(5):521–534, may 2011.

[34] Takashi Iba and Taichi Isaku. Holistic pattern-mining patterns a pattern language for pattern mining on a holistic approach. 2012.

[35] Takashi Iba and Taichi Isaku. A Pattern Language for Creating Pattern Languages: 364 Patterns for Pattern Mining, Writing, and Symbolizing. In *ACM Reference Format*, volume 63, pages 1–63, 2016.

[36] T. Kahkonen. Agile Methods for Large Organizations - Building Communities of Practice. In *Agile Development Conference*, pages 2–11. IEEE, 2004.

[37] Jon R Katzenbach and Douglas K Smith. The discipline of teams. *Harvard Business Review*, pages 111–120, 1993.

[38] Catherine Kerbrat-Orecchioni. Conversation and interaction. In *Handbook of Communication: Verbal Communication*, chapter 9, pages 165–180. De Gruyter, 2016.

[39] Norman Kerth. *Project retrospectives: a handbook for team reviews*. Dorset-House Publishing, 2013.

[40] Charles King. Electoral systems. *Georgetown University*, 1:4, 2000.

[41] Mark L. Knapp, Judith A. Hall, and Terrence G. Horgan. *Nonverbal Communication in Human Interaction*. Wadsworth, 8 edition, 2014.

[42] Henrik Kniberg and Mattias Skarin. *Kanban and Scrum-making the most of both*, volume 1. 2010.

[43] Robert E Kraut and Lynn A. Streeter. Coordination in software development. *Communications of the ACM*, 38(3):69–81, mar 1995.

[44] Jim Krug. PEOPLE SKILLS: Improving the Performance Appraisal Process. *Journal of Management in Engineering*, 14(5):19–20, sep 1998.

[45] Craig Larman and Bas Vodde. *Large-scale Scrum: More with LeSS*. Addison-Wesley Professional, 2016.

[46] Lucas Layman, Laurie Williams, Daniela Damian, and Hynek Bures. Essential communication practices for Extreme Programming in a global software development team. *Information and Software Technology*, 48(9):781–794, 2006.

[47] Patrick Lencioni. *The Five Dysfunctions of a Team*. John Wiley & Sons, 2006.

[48] Edwin A. Locke, Karyll N. Shaw, Lise M. Saari, and Gary P. Latham. Goal setting and task performance: 1969-1980. *Psychological Bulletin*, 90(1):125–152, 1981.

[49] Perry McIntosh, Richard Luecke, and Jeffery H. Davis. *Interpersonal Communication Skills in the Workplace, Second Edition*. AMACOM Div American Mgmt Assn, 2008.

[50] Grigori Melnik and Frank Maurer. Direct verbal communication as a catalyst of agile knowledge sharing. *Proceedings of the Agile Development Conference, ADC 2004*, pages 21–31, 2004.

[51] Gerard Meszaros and Jim Doble. A Pattern Language for Pattern Writing. *Writing*, pages 1–36, 2005.

[52] Deepti Mishra and Alok Mishra. Effective communication, collaboration, and coordination in eXtreme programming: Human-centric perspective in a small organization. *Human Factors and Ergonomics In Manufacturing*, 19(5):438–456, sep 2009.

[53] Orlando Murru, Roberto Deias, and Giampiero Mugheddu. Assessing XP at a European Internet company. *IEEE Software*, 20(3):37–43, 2003.

[54] Ikujiro Nonaka and Noboru Konno. The Concept of "Ba": Building a Foundation for Knowledge Creation. *California Management Review*, 40(3):40–54, apr 1998.

[55] Taiichi Ohno. *Toyota Production System*. Productivity Press, dec 1988.

[56] Tomasz Pieta. The importance of feedback, 2017.

[57] Minna Pikkarainen, Jukka Haikara, Outi Salo, Pekka Abrahamsson, and Jari Still. The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13(3):303–337, jun 2008.

[58] Steve Preston. Meetings bloody meetings or making meetings CLEAR?, 2015.

[59] Andrea Rocci and Margherita Luciani. Semantics and verbal communication. In *Handbook of Communication: Verbal Communication*, chapter 4, pages 57–75. De Gruyter, 2016.

[60] Viviane Santos, Alfredo Goldman, Eduardo Guerra, Cleidson De Souza, and Helen Sharp. A Pattern Language for Inter-Team Knowledge Sharing in Agile Software Development. In *Proceedings of the 20th Conference on Pattern Languages of Programs*, PLoP '13, USA, 2013. The Hillside Group.

[61] Bob Schatz and Ibrahim Abdelshafi. Primavera Gets Agile: A Successful Transition to Agile Development. *IEEE Software*, 22(3):36–42, may 2005.

[62] Ken Schwaber. SCRUM Development Process. In *Business Object Design and Implementation*, pages 117–134. Springer London, London, 1997.

[63] Scrum.org. What is Scrum?, 2020.

[64] Judith Stein. Using the Stages of Team Development.

[65] Chad Storlie. Manage Uncertainty with Commander's Intent. *Harvard Business Review*, (November), nov 2010.

[66] Jeff Sutherland. Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies. *Cutter IT Journal*, 14(12):5–11, 2001.

[67] Jeff Sutherland. Agile development: Lessons learned from the first scrum. *Cutter Agile Project Management Advisory Service: Executive Update*, 5(20):1–4, 2004.

[68] Jeff Sutherland, James O Coplien, Lachlan Heasman, Mark den Hollander, and Cesário Ramos. Scrum Patterns.

[69] Jeff Sutherland and Ken Schwaber. The Scrum Guide. *Software in 30 Days*, 268:133–152, 2017.

[70] Davide Taibi, Valentina Lenarduzzi, Muhammad Ovais Ahmad, and Kari Liukkunen. Comparing Communication Effort within the Scrum, Scrum with Kanban, XP, and Banana Development Processes. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE'17*, volume Part F1286, pages 258–263, New York, New York, USA, 2017. ACM Press.

[71] Hirotaka Takeuchi and Ikujiro Nonaka. The new new product development game. *Polymeric Materials Science and Engineering, Proceedings of the ACS Division of Polymeric Material*, 54:199, 1986.

[72] The Hillside Group. History, 2018.

[73] Bruce W. Tuckman. Developmental sequence in small groups. *Psychological Bulletin*, 63(6):384–399, 1965.

[74] Bruce W Tuckman and Mary Ann C. Jensen. Stages of Small-Group Development Revisited. *Group & Organization Studies*, 2(4):419–427, dec 1977.

[75] Paul Watzlawick, Janet Helmick Beavin, and Don D. Jackson. *Pragmatics of Human Communication*. Number 1. 1967.

[76] Tim Wellhausen and Andreas Fiesser. How to write a pattern? In *Proceedings of the 16th European Conference on Pattern Languages of Programs - EuroPLoP '11*, pages 1–9, New York, New York, USA, 2011. ACM Press.

[77] Don Wells. Extreme Programming: A Gentle Introduction, oct 2013.

[78] Laurie Williams, Robert R. Kessler, Ward Cunningham, and Ron Jeffries. Strengthening the case for pair programming. *IEEE Software*, 17(4):19–25, 2000.

[79] Mark A. Wilson. Collaborative decision making building consensus group decisions for project success. In *Proceedings of the Project Management Institute Global Congress North Americas (2003). S*, number September, pages 25–26, 2003.

[80] Jason S. Wrench, Anne Goding, Danette Ifert Johnson, and Bernardo A. Attias. *Stand Up, Speak Out: The Practice and Ethics of Public Speaking*. FlatWorld, oct 2011.