The Journal of Systems and Software 219 (2025) 112243

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

The Power of Words in Agile vs. Waterfall Development: Written Communication in Hybrid Software Teams☆

Delina Ly[a,b,*], Michiel Overeem[a], Sjaak Brinkkemper[b], Fabiano Dalpiaz[b]

[a] AFAS Software, Inspiratielaan 1, Leusden, 3833 AV, The Netherlands
[b] Utrecht University, Princetonplein 5, Utrecht, 3584 CC, The Netherlands

ARTICLE INFO

Keywords:
Collaborative software development
Development paradigms
Waterfall development
Agile development
Communication channel formality
Hybrid software teams

ABSTRACT

Software development is constantly evolving, adapting to emerging technologies and development paradigms while leveraging advancements in communication technologies and work modes. We conduct an exploratory case study in a large software organization to investigate how the development paradigm and the formality of communication channels affect written communication within hybrid teams. We perform statistical and content analysis of written conversations from 20 projects involving two software products that use industry adaptations of the Waterfall model and of Scrum, respectively. We found that in agile-developed projects, communication related to the execution-monitoring-control phase of the Project Management Life Cycle is more prevalent, and communication related to the initiation phase occurs more frequently in informal channels. For both project types, communication primarily pertains to the software construction phase of the Software Development Life Cycle. After annotating communication contents using speech acts, representatives are found to be prevalent in informal channels for agile-developed projects, directives are more prevalent in informal channels for waterfall-developed projects, and expressives are more frequent in informal channels for both project types. We provide empirical evidence that development paradigms and communication channel formality impact written communication, with agile-developed projects showing more collaborative interactions in informal channels compared to waterfall-developed projects.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

1. Introduction

As a consequence of the COVID-19 pandemic, software organizations have undergone a shift from traditional co-located work setups to *fully remote* (no central location; employees work from home or co-working spaces) and *hybrid* setups (employees alternate between office and remote work on specific days) (de Souza Santos and Ralph, 2022). As we move beyond the pandemic, software organizations are reevaluating their workplace strategies to align with the changing dynamics of the workforce. Reverting to solely co-located work may lead to dissatisfied employees, while fully remote setups could lead to challenges with onboarding, team breakdowns, developer isolation, and communication gaps (Nolan et al., 2021). In light of these challenges, hybrid work emerges as a solution, offering employees flexibility and autonomy while facilitating in-person collaboration and social interaction. If not adopted properly, this setup can have adverse consequences (Houck et al., 2023), including the lack of spontaneous informal face-to-face interactions, which can hamper the fostering of team maturation, cohesion, and resilience (de Souza Santos et al., 2023).

In this hybrid setting, online communication platforms like Gitter, Slack, and Microsoft Teams play a key role in facilitating team communications, coordination, and collaboration (Lin et al., 2016; Shi et al., 2021). These platforms replaced traditional communication such as Internet Relay Chat, forums, and emails (Storey et al., 2014) as they enhance team collaboration, project coordination, and group awareness by simplifying knowledge exchange and the management and organization of distributed conversations (Shi et al., 2021).

Existing research has acknowledged developers' communication as a valuable source, with many studies focusing on self-reported measures by developers or their communication in public instant messaging forms (Lin et al., 2016; Shi et al., 2021; Silva et al., 2022; Mezouar et al., 2022; Parra et al., 2022). However, these studies do not take

☆ Editor: Dr. Christoph Treude.
* Correspondence to: VX Company, Baarnsche Dijk 8, Baarn, 3741 LR, The Netherlands.
E-mail addresses: dly@vxcompany.com (D. Ly), michiel.overeem@afas.nl (M. Overeem), s.brinkkemper@uu.nl (S. Brinkkemper), f.dalpiaz@uu.nl (F. Dalpiaz).

https://doi.org/10.1016/j.jss.2024.112243
Received 29 May 2024; Received in revised form 23 September 2024; Accepted 5 October 2024
Available online 15 October 2024

into account that software developers within organizations often work with other roles, such as product managers, User Interface and User Experience (UI/UX) designers, and testers in disciplinary teams. In contrast, Stray et al. (2019) analyzed how teams in a disciplinary setting communicate on Slack within the large engineering organization Geosoft. In addition to identifying the factors that influence channel preference (e.g., experienced team members preferred open channels), they found that using Slack increased transparency, team awareness, and informal communication.

Based on the foundations provided by previous research, we explore the impact of two factors that may affect communication within teams: the formality of the communication channel and the adopted development paradigm. First, communication within software organizations can be classified as formal or informal. Formal communication is characterized by its structured nature, requiring extensive planning, predetermined participants, and an agenda (Kraut et al., 1990). It is often used for routine and repetitive tasks that require clear instructions and adherence to procedures. In contrast, informal communication is characterized by minimal planning, no predefined structure, and interactive exchanges that are rich in content and change based on team members' current interests, needs, and understanding (Kraut et al., 1990). Informal communication is expected to cope better with new, uncertain, or unexpected events as it facilitates effective coordination and decision-making. Software organizations often use different channels to facilitate formal or informal communication. The formality of these channels may impact the written communication content (Kraut et al., 1990; Törlind and Larsson, 2002; Dorairaj et al., 2011): formal channels may be preferred for conveying official information, such as release notes, while informal channels may be preferred for problem-solving as they allow for more dynamic interactions. Second, software products are often developed through projects that are assigned to specific teams (van de Weerd et al., 2006; Kittlaus and Fricker, 2017). Each project team may adopt a different development paradigm, which may influence communication within teams. For instance, Agile project teams might communicate more frequently due to daily stand-ups.

To obtain an in-depth understanding of written team communication within a real-world context, we conduct an exploratory case study at AFAS Software, a privately held software development organization based in Leusden, the Netherlands. Our case company uses its issue management system, *Insite*, for formal communication through tickets, and it uses the online communication platform Microsoft *Teams* to facilitate informal communication through oral calls, one-on-one written chats (not part of our dataset), and project-related channels (see Section 2.4 for further explanation). AFAS Software develops two software products that use industry adaptations of different development paradigms in the category of enterprise applications: PrWf (Waterfall model, non-agile) and PrAg (Scrum framework, agile). Our main research question is: *How do the employed development paradigm and the formality of the communication channel impact the written communication content within hybrid software development teams?*

As for our research method, we conduct statistical and content analysis by manually coding the formal and informal conversations for 20 projects: 9 for PrWf and 11 for PrAg, which were conducted between 2019 and 2023. The project-related communication content consists of 680 messages, amounting to 15,939 words for PrWf, and 2690 messages, totaling 44,454 words for PrAg. Based on the literature, we construct a coding scheme that characterizes the conversations in terms of (i) the Project Management Life Cycle (PMLC), (ii) the Software Development Life Cycle (SDLC), and (iii) speech acts (refer to Section 2.7 for justification). We analyze the data to determine if the development paradigm, the formality of the communication channel, or both influence the code frequency within these three categories. We employ data triangulation by extracting communication content from the internal communication channels and conducting a supplementary survey with team members involved in the selected projects to validate or refute findings derived from our analysis.

By answering the main research question, this paper makes the following three contributions:

- Our study is among the first to analyze the communication within project teams, extending the focus beyond studies that solely center on software developers. While doing so, we extract communication from the internal channels of a software organization, rather than depending on publicly accessible data.
- We investigate the effect of two factors on the communication content: the adopted development paradigm and the formality of the communication channel.
- We analyze the communication content of 20 hybrid project teams within a large software organization. Specifically, we provide insights into how the communication content pertains to the phases of the PMLC, the phases of the SDLC, and which speech act types are involved.

The remainder of the paper is organized as follows. In Section 2, we refine the main research question into six sub-research questions and hypotheses and describe the research setting and methodology. We present and interpret the results in Section 3. We address the research questions, provide further discussion on the results, compare our findings to previous studies, discuss the threats to the validity of our study, and present the implications and recommendations in Section 4. Finally, in Section 5, we provide a summary.

2. Research setting and methodology

We present our refined research questions and hypotheses in Section 2.1. Our research setting, including the case company, their software products, development paradigms, and communication channels, is described in Sections 2.2, 2.3, and 2.4. Subsequently, we discuss our methodology, detailing project sampling criteria (Section 2.5), communication content extraction (Section 2.6), coding scheme, and data analysis procedures (Section 2.7). We conclude with a survey to validate our statistical and content analysis findings.

2.1. Research questions and hypotheses

To address the main research question presented in the introduction, we operationalize communication content through a coding scheme comprising codes derived from the PMLC, the SDLC, and speech acts types, which leads to the following sub-research questions:

- *How do the employed development paradigm and the formality of the communication channel impact the content of written communication pertaining to the Project Management Life Cycle phases? (RQ1 A & RQ1B)*
- *How do the employed development paradigm and the formality of the communication channel impact the content of written communication pertaining to the Software Development Life Cycle phases? (RQ2 A & RQ2B)*
- *How do the employed development paradigm and the formality of the communication channel impact the use of speech acts types in written communication? (RQ3 A & RQ3B)*

To answer these research questions, we test the following hypotheses regarding the PMLC, the SDLC, and speech acts types:

The content of written communication within a development team

H$_1$: is affected by the development paradigm.

H$_{1.1}$: in formal communication channels is affected by the development paradigm.

H$_{1.2}$: in informal communication channels is affected by the development paradigm.

The content of written communication within

H$_2$: a development team is affected by the formality of the communication channel.

H$_{2.1}$: a non-agile development team is affected by the formality of the communication channel.

H$_{2.2}$: an agile development team is affected by the formality of the communication channel.

2.2. Case company

As defined by Wohlin (2021) and Wohlin and Rainer (2022), a case study is *"an empirical investigation of a case, using multiple data collection methods, to study a contemporary phenomenon in its real-life context, and with the investigators(s) not taking an active role in the case investigated"*. We conducted a case study on AFAS Software, a privately held Dutch software development organization based in Leusden, the Netherlands, with branches in Belgium, Curacao, and Aruba. The company employs over 650 employees, including 150 employees allocated to the research and development department.

We selected AFAS Software for its suitability in facilitating a comparison of communication content between two software products; each developed using industry adaptations of different development paradigm - PrWf (Waterfall model, non-agile) and PrAg (Scrum framework, agile). Additionally, we employed convenience sampling since two authors were embedded in the company — the first author associated with PrWf and the second author with PrAg. The authors started this study after the selected projects had been completed and did not actively engage in the investigated case.

In real-world contexts, project teams often adjust development paradigms to align with their team practices, the complexity of their work, and organizational structures (Mortada et al., 2020). This also applies to our case company. Product managers have formally defined roles and clear responsibilities in the Waterfall model. However, in the Scrum framework, this role is not explicitly defined; instead, the responsibilities are typically assigned to the Product Owner (West, 2016). In our case company, product managers are still involved in agile-developed projects as this is inherited from the existing organizational structure.

Furthermore, the roles of software architects and developers are not strictly separated (Galster et al., 2016); depending on the workload, team members may perform tasks associated with both roles. Software architects typically handle three primary responsibilities: (1) architectural design, (2) internal communication, and (3) external communication (Kruchten, 2008). In our case company, software architects are also tasked with writing code. Both PrAg and PrWf software architects write code, but only PrAg software architects write automated tests.

2.3. Products & associated development paradigms

Xu and Brinkkemper (2007) define a software product as *"a ready-to-use package of software components or services, along with supporting materials, that is made available for sale within a specific market"*. We collected communication content from two software products developed by AFAS Software: PrWf and PrAg. PrWf is a mature enterprise resource planning system that offers the functionality needed to automate administrative processes in medium to large organizations. The more recent product, PrAg, offers the functionality to automate accounting for small organizations and is used by accounting firms.

The releases of both products are developed through a series of projects. Projects are unique ventures aimed at producing a set of deliverables within a specified time, cost, and quality constraints (Westland, 2007). Projects have a specific goal to research, develop, enhance, or remove one or more functionality modules. These functionality modules are broken down into more granular aspects that, in our case company, are called features. PrWf project teams work on developing pre-defined features and addressing maintenance tickets leading up to scheduled releases, which occur three times per year (see Section 2.3.1). PrAg project teams work on a specific set of features and maintenance tickets in two-week sprints, with releases scheduled every four weeks due to commercial considerations (see Section 2.3.2). Both PrWf and PrAg address any bugs that disrupt the production environment immediately, regardless of the release schedule.

Although there are differences between PrWf and PrAg (see Table 1), such as product type and maturity, the products are still suitable for comparison when analyzing communication content. Specifically, despite differing product types, both products fall under the category of *enterprise software*. Notably, the launch age of PrAg (date: June 17, 2021) is different from its actual age, as the product has been in development for 11 years. Thus, *both organizational units are mature* in the context of communication. Table 2 presents an overview of the development paradigms adopted by PrWf and PrAg. The development paradigm conceptually encompasses principles shared among a group of methods, and these methods establish the process for developing applications and specify the role of techniques within this process (Brinkkemper, 1996).

2.3.1. PrWf's Waterfall development approach

PrWf uses an industry-adapted version of the Waterfall model. Insite tickets are primarily created for feature requests, modifications, or issues related to existing features, often initiated by a client. The product manager and UX/UI designer identify, define, and determine which features to implement in the next release. The prioritization is influenced by the number of client requests for specific features, substantial financial contributions from clients, the amount of development effort required, insights obtained from the market, and any relevant legal changes.

After defining a new release, which includes selecting projects — each possibly comprising one or more features — the projects are allocated to a team. The UI/UX designer undertakes discussions with the client, software architects, and other organizational units to interpret and validate the requirements. Subsequently, the designer creates a comprehensive document encompassing the feature's functional and technical aspects and a graphical interface design. Upon completing this document, the designer notifies the responsible software architect via the ticket that the feature is ready for construction. The software architect implements the feature according to the specifications outlined in the design document without undergoing any code review. In instances where applicable, the team has members who modify the configuration, page content and layout, and templates.

Following these updates, the feature is manually tested. Once testing (including acceptance testing) is successfully completed and has obtained written approval from the test & quality team member, the feature ticket is forwarded to a team member responsible for documentation, including updating release notes, online manuals, courses, and videos. The tickets follow a linear progression. However, if any unforeseen issue arises, the tickets have to revisit any preceding steps; for instance, if an issue occurs during construction, the ticket is reassigned to the designer, who reviews and addresses any uncertainties before the software architect resumes progress. PrWf adheres to a

Table 1
Product and development details of PrWf and PrAg.

Product details	PrWf	PrAg
Type	Software product to automate administrative processes	Software product to automate accounting
Category	Enterprise application	Enterprise application
Product Life Cycle stage (Cao and Folan, 2012)	Maturity stage	Introduction stage
Number of employees	Around 100 employees	Around 50 employees
Number of clients[a]	Over 12,000 organizations, with variations in size from medium to large	200 organizations, consisting of entrepreneurs with small businesses and freelancers[b]
Development details	PrWf	PrAg
Development years	25 years	11 years
Employed paradigm	Waterfall model (non-agile) (see Section 2.3.1)	Scrum framework (agile) (see Section 2.3.2)
Technology	VB6, C# and SQL	PrAg's Low-code platform (van der Schuur et al., 2017)
Version control system	TFS	Git

[a] In this study, the term *'organizations'* denotes business entities wherein multiple establishments across various locations are classified as a single entity.
[b] During the studied period from May 23, 2019, to June 7, 2023, PrAg was used by 200 organizations comprising small business entrepreneurs and freelancers. The client base has demonstrated remarkable growth, expanding to 7000 organizations, indicating that PrAg is currently in the growth stage.

Table 2
Overview of the development approaches of PrWf and PrAg.

Aspect	PrWf	PrAg
Development approach	Industry adaptation of the Waterfall model.	Industry adaptation of the Scrum framework.
Flexibility	If unforeseen issues occur, adjustments can be made throughout the process, but it is necessary to revisit previous steps.	Features can be modified during sprints in response to stakeholder feedback as long as they align with the sprint goal.
Ticket organization	*Insite* tickets without a defined structure.	*Insite* tickets are linked to feature tickets in *Azure DevOps*.
Progress monitoring	Tickets follow a sequential progression. Weekly team meetings take place to provide updates on project progress and address emerging concerns.	During daily stand-ups, team members share project progress, discuss upcoming tasks, address impediments, and coordinate efforts.
Requirements	The product manager and UI/UX designer identify, define, prioritize requirements, and decide which features to implement in the next release. The prioritization is influenced by factors such as client requests, market trends, and legal changes.	The product manager and business analyst identify any missing features and formulate requirements, while the software architects assess the scope and feasibility of features and partly decide whether requirements are implemented.
Planning	The project is assigned to a team. The UI/UX designer has discussions with clients and organizational units to interpret and validate requirements.	Team members develop a technical plan for the items on the backlog and estimate the workload. Every two weeks, backlog items with the highest priority are placed on the sprint backlog through a collaboration of team members and the product owner.
Design	UI/UX designers create a comprehensive document encompassing a graphical interface design, functional and technical aspects, and incorporate input from software architects.	UI/UX designers create a graphical interface design and serve a dual role as software architects engaging in modeling and development.
Construction	Features are implemented according to the document created in the design phase. Code reviews are not performed.	During the two-week sprint, team members work on the implementation of the sprint backlog items. Team members conduct code reviews in *Azure DevOps*.
Testing	Manual testing is conducted by a dedicated test department. The implemented features undergo acceptance testing.	Unit, integration, and end-to-end tests are written, and manual testing is performed when required. The implemented features undergo acceptance testing.
Release	Features are documented in the release notes. Release notes, online manuals, courses, and videos are updated by documentation employees. Releases occur three times a year.	After the sprint, the team meets and demonstrates the implemented features. The team also meets to reflect on the sprint, identify areas for improvement, assess risks and concerns, and devise strategies to improve the next sprint. Features are documented in the release notes. Although the features implemented in each sprint are shippable, the project teams decided to schedule releases every four weeks due to commercial considerations.
Maintenance	Maintenance is typically carried out via *Insite* tickets and only involves development efforts. Maintenance-related tickets are planned for an upcoming release. Production-disrupting bugs are addressed immediately.	Maintenance is typically carried out via *Insite* tickets and only involves development efforts. Maintenance-related tickets are planned into a sprint if the priority allows it. Production-disrupting bugs are addressed immediately.

release frequency of three times a year. Maintenance-related tickets are planned for an upcoming release. Production-disrupting bugs, however, are addressed immediately.

PrWf's approach aligns with the Waterfall model, known for its plan-driven methodology where processes, final deliverables, project costs, and completion dates are determined at the project's outset (Royce, 1970). The approach also underscores the Waterfall model's limitation in addressing unexpected outputs from intermediate processes, as tickets have to revisit preceding steps to resolve unforeseen issues. Additionally, PrWf's approach differs from the V-Model, which emphasizes the relationship between development and testing activities, with testing occurring in parallel with development at each stage (Weilkiens et al., 2022), as their approach follows a linear progression, and testing occurs after development. Furthermore, PrWf's approach differs from RUP, which is iterative and incremental and embraces continuous feedback, changes, and evolving documentation. In contrast, PrWf's approach is linear and sequential, with limited feedback after the requirements are finalized, costly changes once a phase is completed, and an emphasis on comprehensive documentation.

2.3.2. PrAg's Agile development approach

PrAg adopts an industry-adapted version of the Scrum framework. The teams create a single feature ticket for a specific functionality module. This ticket is subdivided into several smaller tickets that represent the verticals within the functionality module (user stories). A vertical is a separate part of the functionality module that the team can implement and deliver independently. Such a vertical is then divided into even smaller tickets (tasks). In most cases, tasks related to identifying missing functionality, planning, designing, and releasing software are conducted on the feature level. In contrast, the other tasks, such as software construction, testing, and maintenance, are addressed on the vertical level. PrAg operates with cross-functional and self-sufficient project teams, where all team members possess the essential skills and expertise required to deliver a shippable product without relying on external teams. While team members each have their strengths, they are flexible and capable of supporting various tasks as needed. For example, software architects contribute not only through design but also by writing code and unit tests.

The product manager and business analyst identify missing features and formulate requirements, while the software architects partly determine the scope and feasibility of features and whether requirements are implemented. Subsequently, the business analyst and software architects identify the smallest step required to satisfy certain requirements. This identified step is placed as an item on the backlog in *Azure DevOps* and prioritized by team members. The UI/UX designers not only create a graphical interface design for this item but also serve a dual role as software architects, actively engaging in modeling and development. The team members proceed to develop a technical plan for the items on the backlog and estimate the workload.

The project teams have opted for a two-week sprint as it allows them to make necessary adjustments while providing enough time to achieve meaningful progress. Every two weeks, backlog items with the highest priority are placed on the sprint through a collaboration between the team and the product owner. Maintenance-related tickets are included in a sprint if the priority allows it, while production-disrupting bugs are addressed immediately. Throughout this period, team members work on these items. The team members also write unit, integration, and end-to-end tests, and when required, a test & quality team member performs manual testing on the items. Team members conduct code reviews in *Azure DevOps*. When a pull request is ready for review, the author of the pull request notifies the team via the project-related *Teams* channel. Once the appropriate team member(s), depending on their knowledge, have reviewed the pull request, the pull request is forwarded. The pull request comments were not included in our data set as they were not within the primary focus of our study. Upon completion of a pull request, a message is automatically generated in Insite. During daily stand-ups, team members share progress, discuss upcoming tasks, address impediments, and coordinate efforts. The team is responsible for implementing features, allocating members to these tasks, and determining the timing of their inclusion in the sprint. After a two-week interval, the team meets and demonstrates the implemented features. These features undergo acceptance testing by test & quality team members, and once approved, they are documented in the release notes. Although the features implemented in each sprint are shippable, the project teams decided to schedule releases every four weeks due to commercial considerations.

This approach aligns with Scrum (Schwaber, 1997; Sutherland and Sutherland, 2014; Schwaber and Sutherland, 2020). The framework starts with planning and defining a new release based on the current backlog, along with providing estimates. Additionally, a design is made to implement backlog items, including system architecture modifications and high-level designs. This process, also referred to as backlog grooming or refinement, involves the product owner and the team preparing the work for the sprint to ensure that it is sufficiently detailed to start development. After planning and preliminary designing activities have been completed, a sprint takes place, which focuses on the development of new functionalities. Sprints contribute to the gradual evolution of the system. After the sprint ends, team members meet to demonstrate the implemented features *(sprint review)*. The team also meets to reflect on the previous sprint, identify areas for improvement, assess risks and concerns, and devise strategies to improve the next sprint *(sprint retrospective)*. PrAg can conduct multiple sprints for a project and follows a release schedule of every four weeks, starting from 2020. PrAg's approach is well-aligned with Scrum, given its smaller organizational unit of 50 employees with independent teams. In contrast, the Scalable Agile Framework is designed for larger organizations, where interconnected teams collaborate on enterprise-scale projects (Leffingwell, 2010). Moreover, PrAg's approach differs from Scrumban, where work is managed through a continuous flow system, with tasks pulled based on capacity using boards and cards. The features are released as soon as they have been completed. In contrast, PrAg uses fixed-length sprints, where project teams commit to a defined set of work for the sprint duration, using product and sprint backlogs. The features are released after every two sprints.

2.4. Communication channels & formality

Communication can take on two distinct forms: informal and formal. Informal communication is characterized by interactivity and includes unplanned interactions that occur during daily activities (Törlind and Larsson, 2002; Dorairaj et al., 2011). Kraut et al. (1990) distinguish informal from formal communication based on the degree of preplanning and spontaneity. Informal communication involves minimal planning and lacks scheduled timing, participants, or an agenda in advance. Moreover, it is interactive and changes based on participants' current interests, needs, and understanding. On the other hand, formal communication requires more extensive planning efforts with known participants and a predetermined agenda.

Our case company uses Microsoft *Teams* to facilitate informal communication through oral calls, one-on-one written chats (not part of our dataset), and project-related channels. Collaboration occurs through project-related channels, which focus on relevant discussions in the proper context to ensure transparency and traceability, thereby reducing the need for formal meetings (Hubbard et al., 2021). Moreover, the company uses its issue management system, *Insite*, for formal communication through tickets. Although the specific timing of reassigning a ticket to a colleague is not explicitly defined, communication is bound to occur when the ticket is reassigned due to a structured development process and, thus, lacking spontaneity. In our case company, pair programming is implemented on an as-needed basis, either to support junior employees or to address particularly complex problems, and in these situations, a designated ticket owner is always in place.

2.5. Projects sampling

The releases of both products are developed by carrying out a series of projects, each dedicated to implementing one or more functionalities. We are interested in the written project-related communication that has taken place in the communication channels (for more information, refer to Sections 2.4 and 2.6). Thus, we applied purposive sampling (Patton, 2002) to select projects based on five criteria: (i) the conversation must involve at least two human participants; (ii) each participant must have written at least one message; (iii) the conversation must be related to the project; (iv) the project teams must have worked in a hybrid mode, with members alternating between working in an office and remotely on specific days (de Souza Santos and Ralph, 2022), and (v) the projects must have taken place between 2019 and 2023 (see Fig. 1). We chose to study this period as teams heavily relied on digital communication

Table 3
Project details from our dataset.

Project	Description	Team members	Messages	Words	Labels
PrWF1	This project explores the possibility of integrating with the Office365 calendar and adds functionality to save emails in the file.	32	187	4 646	436
PrWF2	The optimization of planned tasks, which enables clients to schedule tasks such as sending emails on a certain day and time.	18	26	815	55
PrWF3	The translation functionality has been redesigned for improved efficiency, and the existing translations have been manually updated.	18	116	2 853	284
PrWF4	The optimization of the authorization tool, which enables clients to tailor role-based access to specific functionalities.	6	24	373	60
PrWF5	The optimization of theme customization in *Insite* was implemented.	17	146	3 470	400
PrWF6	Two-factor authentication was implemented for Outsite, an extranet for clients.	20	76	2 571	178
PrWF7	This project focused on researching possibilities to implement webhooks to facilitate communication with external applications.	5	12	165	20
PrWF8	The calendar feature was implemented in *Insite* (web-based).	7	42	482	85
PrWF9	This project focused on discontinuing sending SMS for login.	10	51	564	115
Total PrWf		133	680	15 939	1633
PrAG1	The Bazel build system has been implemented in the code repository.	10	460	7 260	871
PrAG2	An activation system has been developed to conceal a specific functionality module until a customer truly intends to deploy the functionality module.	9	145	2 067	270
PrAG3	A feature has been developed that allows an accountant to categorize specific transactions based on factors like department, providing insights into the performance of certain parts of the organization.	9	309	6 202	847
PrAG4	A foundational feature of a workflow system has been developed that allows for the addition of an approval step to a quotation.	8	144	1 879	301
PrAG5	Provision was developed for collecting data to be sent in a message.	11	256	4 548	652
PrAG6	Provision was developed for importing data in a standardized manner.	6	218	3 311	392
PrAG7	Functionality was developed to upload CSV files and import data from these files.	9	113	1 756	235
PrAG8	Provisions were developed to manage binary data.	6	152	2 551	311
PrAG9	Provision was developed to create custom UI views on top of the standard UI views that were included in the product.	16	299	5 291	390
PrAG10	Scan and recognize functionality has been developed to enable the automatic processing of incoming invoices.	16	188	2 853	213
PrAG11	Provision was developed to create custom property screens.	12	406	6 736	956
Total PrAg		112	2690	44 454	5438

channels due to the restrictions imposed by the COVID-19 pandemic. Although some projects started before the pandemic, they continued for a considerable duration, with project teams operating in a hybrid mode; thus, these projects remain pertinent to our study. When we analyzed the composition of the project teams, we observed that one PrWf project had significantly more members compared to the other PrWf projects using the interquartile range method (Vinutha et al., 2018). Consequently, we excluded this project from our analysis, recognizing it as an outlier due to its significant deviation in team composition. Table 3 provides an overview of the retained projects.

2.5.1. *Project team compositions*

Project teams are formed by selecting members from the software development organizational units and other units such as *product management, test & quality, and knowledge & content*. As a result, there are multiple project team compositions. A detailed overview of the project teams' composition can be found in the online appendix (Ly et al., 2024). The development methods of the project teams align with those of their respective software products: PrWf uses an industry-adapted version of the Waterfall model, while PrAg uses an industry-adapted version of the Scrum framework.

2.6. *Data extraction and preparation*

Project-related communication was acquired from communication channels *Teams* and *Insite* with the knowledge of the executive management and team members involved in the selected projects at AFAS Software. Although the data was publicly accessible to all employees, they were given the choice to decline access if they did not want their project-related communication to be studied. Every project within our sample has an assigned *Teams* channel. The data within *Teams* was extracted through Microsoft's secured API, which required a request form to be submitted and approved, taking approximately a week. The cost per gathered message was $0.00075. The result was 20 JSON files, each file dedicated to one project. A SQL query was executed on AFAS Software's cloud database to obtain the data from *Insite*, and the result was exported to `.rpt` files. These files contain communication in the form of comments attached to tickets pertaining to feature requests, modifications, or issues related to existing features. The communication extracted from *Insite* and *Teams* was written in Dutch and organized according to product, project, and communication channel. We obtained one Word document for *Teams* and one for *Insite* per project, totaling 20 projects and resulting in 40 unique Microsoft Word documents. The written communication extracted from *Insite* corresponds to 127 tickets for PrWf and 75 tickets for PrAg. These documents included the

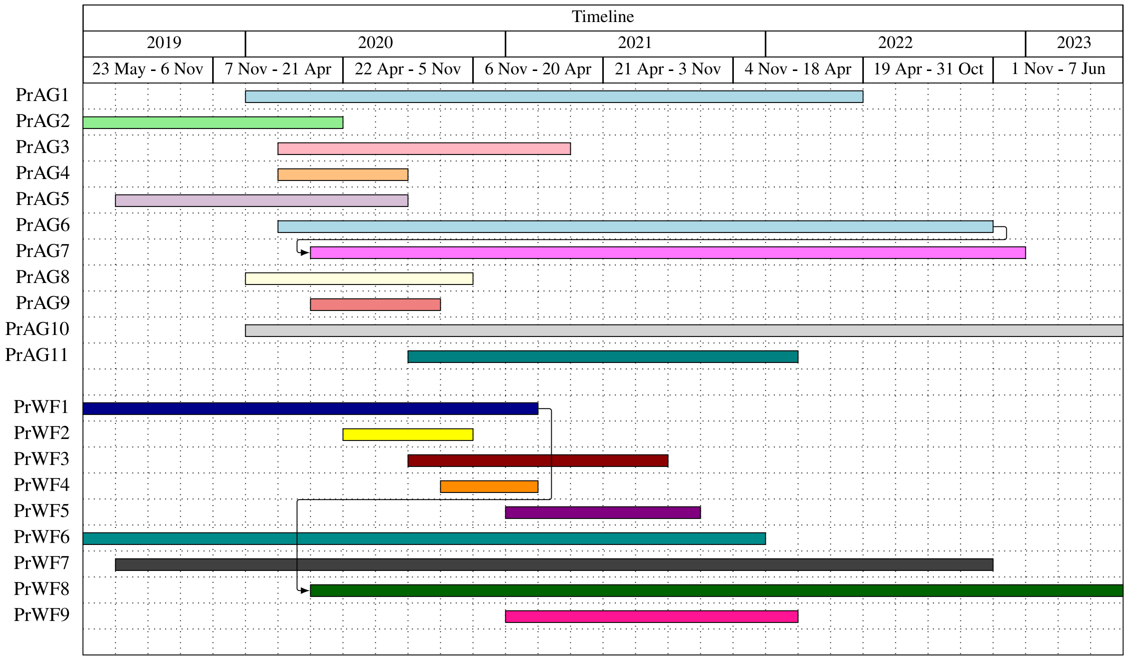

Fig. 1. Timeline of the twenty selected projects, with PrAg following a release schedule of every four weeks, starting from the year 2020, and PrWf adhering to a release frequency of three times a year. PrWf8 is dependent on PrWf1: PrWf1 was a project aimed at exploring the possibility of integrating with the Office 365 calendar. Since integration was not possible, the decision was made to implement the calendar in *Insite* instead. The remainder of PrWf1 was focused on adding functionality to save emails in the file.

Table 4
Communication channels data details, showing average values across the projects.

PrWf	*Teams*	*Insite*
Team members	~4 team members	~11 team members
Messages	7 messages	69 messages
Words	150 words	1621 words
PrAg	*Teams*	*Insite*
Team members	~6 team members	~5 team members
Messages	197 messages	47 messages
Words	3237 words	804 words

The data in Table 4 shows that, on average, PrWf has more participants, messages, and words in *Insite*, while PrAg reveals the opposite pattern, with more communication occurring in *Teams*. However, it is crucial to move beyond these basic quantitative metrics and examine the *content* of the messages. This examination requires analyzing factors such as the intended meaning behind a text and discussed topics that can help assess the relevance of the communication. Our data analysis focuses on an in-depth, qualitative investigation of the communication contents, not only analyzing the quantitative but also delving into the qualitative aspects.

project's code and title, the names and roles of project team members, and the written messages arranged in sequential order. Table 4 provides an overview of the average values extracted from the communication channels across the projects. While we acknowledge that this data set only represents a part of the communication, we were unable to collect oral communication, one-to-one chats, team meeting chats, emails, or WhatsApp conversations due to privacy regulations.

2.7. Coding scheme and data analysis

Based on the literature, we constructed a coding scheme that characterizes the communication content in terms of (i) the Project Management Life Cycle (PMLC) (Westland, 2007), (ii) the Software Development Life Cycle (SDLC) (Bourque and Fairley, 2014), and (iii) speech acts (Searle, 1969, 1979). The details of the coding scheme, including coding units, descriptions, and examples, are presented in Table 5.

We integrated the PMLC and the SDLC into our coding scheme due to the diverse nature of our projects, requiring a shared foundation for communication comparison. This integration assists in identifying communication topics and their alignment with the PMLC/SDLC phases. Furthermore, it facilitates the analysis of communication patterns and gives insights into how communication evolves throughout the project's life cycle. While independent, the cycles are related, *"the project life cycle encompasses all activities of the project, while the systems development life cycle focuses on realizing the product requirements"* (Taylor, 2003). Note that our labeling of the PMLC/SDLC phases is a *mapping* of the communication contents based on its topic (a message related to creating a GANTT chart would be mapped, e.g., to the planning phase of the PMLC), but it does not allow us to make inferences on the sequencing or timing of these phases in the projects.

Inspired by other researchers in software engineering (Wood et al., 2018; Morales-Ramirez et al., 2019), we incorporated speech acts into our coding scheme to gain insights into the *illocutionary force* or intended meaning behind the text and to uncover any implications or hidden intentions. This approach can reveal communication and collaboration patterns and highlight potential areas of conflict or misunderstanding.

2.7.1. Segments and coding units

Our data represents *message exchanges*, a process where two or more individuals communicate with each other by sending and receiving messages containing information. To analyze such exchanges, we divided the data into segments based on the topic of each message. Whenever the topic changed, we created a new segment. We related these segments to the PMLC and the SDLC phases. Whenever coding within one segment, we analyzed the preceding and succeeding segments and any references to earlier or later segments to gain a better understanding of the communication patterns. We labeled speech acts at the sentence or phrase level.

Our coding approach deviates from the conventional notion of speech acts, which typically suggests that a single sentence should primarily exhibit a sole speech act type; we rely on more recent researchers who acknowledged that a sentence may convey multiple speech acts simultaneously (Qadir and Riloff, 2011). Table 6 illustrates our coding approach with a partial conversation from PrAg1, translated into English.

Table 5
Our coding scheme, descriptions, and examples from our domain of enterprise applications.

PMLC phase (coding unit: per segment)	Description taken from Westland (2007)	Example
Initiation (including requirements, abbreviated as ‘initiation’)	A problem or opportunity prompts the formulation of a business case with solution options, followed by a feasibility study, and a final recommended solution is put forward. Once the recommended solution is approved, the project begins with clearly defined objectives, scope, and structure. A project manager is appointed to assemble the project team and establish an office environment. Approval is sought to continue to the detailed planning phase.	“The desire is also to have the capability to easily access the translated description of the country (of the organization) from any data collection for connectors, provisions, reports, analyses, etc., without needing to resort to various workarounds (such as scripting, etc.) for something that appears to be straightforward”. (PrWf3)
Planning	Key activities include developing a comprehensive project plan outlining tasks, dependencies, and time frames, along with plans for resources, finances, quality, risk management, customer acceptance criteria, stakeholder communication, and procurement for effective project execution. The project will have been planned in detail and is ready to be executed.	“I have secured this with a feature so that we can plan this”. (PrAg10)
Execution-Monitoring-Control (abbreviated as ‘execution’)	This phase involves implementing the plans created during the planning phase. While each plan is executed, management processes are undertaken to monitor and control the project’s output of deliverables. This includes identifying changes, assessing issues and risks, evaluating quality, and measuring deliverables against acceptance criteria. The project proceeds to the closing phase once all the deliverables have been implemented, and the client has accepted the final solution.	“If this needs to be solved, the technical possibilities must be examined. The way it is currently sent is no longer sufficient”. (PrWf1)
Closing (or closure)	Final project deliverables are handed over to the client, documentation is transferred, contracts are terminated, project resources are released, and project closure is communicated. A post-implementation review takes place to assess the project’s success and the lessons learned for future projects.	“As far as I’m concerned, the project should go to documentation”. (PrWf3)
SDLC phase (coding unit: per segment)	Description taken from Bourque and Fairley (2014)	Example
~~*Requirements*~~[a]	Requirements pertaining to the activities encompassing elicitation, analysis, specification, validation, and ongoing management throughout the entire life cycle of the software product. **[SWEBOK, Chapter 1]**	“I would like to include this in the backend project in which we will ensure that we support multiple formats”. (PrAg7)
Design	The process wherein software requirements are analyzed to generate a description of the internal structure of the software that will serve as the basis for its construction. **[SWEBOK, Chapter 2]**	“After consulting with [UI/UX designer’s name], it is deemed more effective to consolidate the settings under the three dots, similar to the Gmail add-on”. (PrWf1)
Construction	Software construction refers to the detailed creation of working software through a combination of coding, verification, unit testing, integration testing, and debugging. **[SWEBOK, Chapter 3]**	“I really want to implement that host pull request. All queries with a skip/take must return lines with InstanceIds, otherwise a client can never paginate”. (PrAg3)
Testing	Software testing consists of the dynamic verification that a program provides expected behaviors on a finite set of test cases, suitably selected from the usually infinite execution domain. **[SWEBOK, Chapter 4]**	“For testing, I recommend focusing on less frequently used functions, such as import, getconnector, updateconnector of users, and user synchronization”. (PrWf7)
Release	Activities that pertain to both the pre-delivery and delivery stages of the software. **[We split Chapter 5 of the SWEBOK into release and maintenance.]**	“The following (plus the extensions in the User Interface) were included when PrWF [version number] was finalized. (PrWf6)
Maintenance	Activities that pertain to support of the software after it has been released or delivered. **[We split Chapter 5 of the SWEBOK into release and maintenance.]**	“I am going to do a pilot at a customer, if this has the desired result, we will come up with a patch”. (PrWf1)
Speech act (coding unit: per sentence or phrase)	Description taken from Searle (1969, 1979) S = Speaker, X = Situation	Example
Representatives (or assertives)	A speech act that tries to represent a situation or condition and also describes the states or events in the world. S believes X.	“I’ve updated the tasks on the board”. (PrAg7)
Directives	A speech act that causes the hearer to take a particular action. S wants X.	“Can you also update the Excel?” (PrWf3)
Commissives	A speech act whose point is to commit the speaker to some future course of action. S intends X.	“I will merge this to our branch”. (PrAg7)

(*continued on next page*)

Table 5 (*continued*).

Speech act (coding unit: per sentence or phrase)	Description taken from Searle (1969, 1979) S = Speaker, X = Situation	Example
Expressives	A speech act that expresses the speaker's emotions and attitudes towards the proposition. S feels X.	"Works like a charm :)" (PrWf5)
~~*Declaratives*~~[b]	A speech act that immediately alters the state of affairs in the world. S causes X.	"You are kicked off the project!" (hypothetical quote)

[a] In our dataset, the requirements phase of the SDLC showed a notably low frequency. Within our case company, the requirements engineering phase primarily occurs prior to and independently of the *Insite* and *Teams* communication channels. Consequently, we merged this phase with the initiation phase of the PMLC. This observation is confirmed by the survey respondents: when it comes to requirements-related communication, for PrAg, 50% primarily use oral communication, 30% use *Teams*, and 20% use other written communication channels; for PrWf, 69.23% mainly use oral communication and 30.77% use *Teams*.

[b] Additionally, declaratives were excluded from the codebook as no occurrences were found in our dataset.

Table 6
Illustration of our coding based on an excerpt of PrAg1: two segments, with sentences having different speech act types: representative, directive, commissive, and expressive; and the PMLC phases and the SDLC phases.

Spk	Text	PMLC	SDLC
Lead	I have started working on the support for analyzers: [anonymized GitHub links]	exec-monitoring-control	construction
	Do you think this is the right way to go, [Arch1]?		
	I'm putting the task back for now to work on sth else first.		
Arch1	♡		
	Looks good! I just didn't expect analyzers on the BUILD files of projects.		
Lead	Where would you expect them? via the context data?		
Arch1	Yes, or as deps, or just through context_data.		
	I think via context would be easier/more elegant to conditionally make it only for release builds. See https://docs.bazel.build/versions/master/configurable-attributes.html		
Lead	👍		
Arch1	Another file name with spaces has been checked in.	exec-monitor-control	construction
	This causes bazel to fail [anonymized error path]		
Arch2	☺		
Arch1	I'll fix it on the reformat branch.		
Lead	You have to call out the [anonymized] team.		
Arch1	It's a trailing space, so I missed it. When we have a PR build, that won't happen anymore.		
Arch2	👍		

2.7.2. *Qualitative data analysis*

We imported the Microsoft Word documents containing communication content into the qualitative data analysis software *Nvivo*. The first author employed *abduction* (Walton, 2014), an approach combining deductive and inductive methods to code the communication content. This approach provides a common foundation for comparing communication content and allows for the expansion of the coding scheme with new content-based codes.

Additionally, to enhance the reliability of our coding and minimize potential biases, we involved an independent coder (not an author of this paper) in the initial phases of our research; we discussed the disagreements as a means to obtain the coding scheme employed in this paper. Due to the time-intensive nature of the coding task, the coder analyzed the communication from five randomly selected projects: PrWf2, PrWf3, PrWf4, PrWf5, and PrAg4. The coder, a Master's student in Business Informatics who was interning at the case company, analyzed both communication channels for each project. The student was knowledgeable in coding techniques and the domain of enterprise applications. The coding process was conducted as follows: (1) the first author explained the coding scheme (an early version of Table 5 without the PMLC aspect) and approach (see Table 6) to the independent coder; (2) both the first author and the coder independently labeled communication of two projects; (3) a face-to-face meeting was held to discuss coding differences, primarily in the inductive codes (e.g., the inductive labels *"requesting"* or *"advising"* fall under the deductive label *"directive"*), and adjustments to the coding scheme were made; (4) another round of independent coding was conducted on the communication of the remaining three projects using the revised coding scheme; (5) a final meeting was held to review and resolve any remaining differences.

The tagging of these projects led to a *substantial agreement* (Landis and Koch, 1977) (0.78) for the speech acts, and a *fair-to-moderate agreement* (0.40) for the SDLC phases. The latter result can be explained by the need to have a profound knowledge of the projects to precisely determine to which SDLC phases the messages refer to. Based on the discussion between the taggers, we decided to introduce the PMLC phases as a separate aspect to decouple project management from software development considerations.

Fig. 2 outlines the data analysis process. We begin by familiarizing ourselves with the communication content of the project teams. The first gray block illustrates the deductive coding (steps 2a & 3a). Deductive codes, such as *initiation (including requirements)*, are employed for statistical analyses (steps 4a, 5a, 6a). The subsequent gray block presents inductive coding (steps 2b-5b). Inductive codes, like *requirements validation*, are used for context and interpretation (step 6b). Both deductive and inductive coding occur concurrently.

2.7.3. *Quantitative data analysis*

The code frequency of speech acts, the PMLC, the SDLC, and their underlying codes were extracted from *Nvivo* using a matrix coding query and were imported into Excel. As we conducted an analysis based on the deductive codes, we added the frequency of inductive codes to that of the deductive codes. A detailed breakdown of the labeled data can be found in the online appendix (located in the *"Excel Files"* folder, named *"JSS Paper.xlsx"*) (Ly et al., 2024). We converted the absolute numbers to percentages within each category considering product, project, and communication channel in Excel. As an illustration, in the PrWf1 project associated with the formal communication channel, the absolute count for the "directive" speech act is 96. To calculate the percentage, we divided this count by the total number of occurrences of all speech act types in the PrWf project, which amounts to 240. Therefore, 96 divided by 240 results in 40%. These percentages facilitate comparisons in distribution. To conduct these calculations, one author developed a Python script, which is available in the online

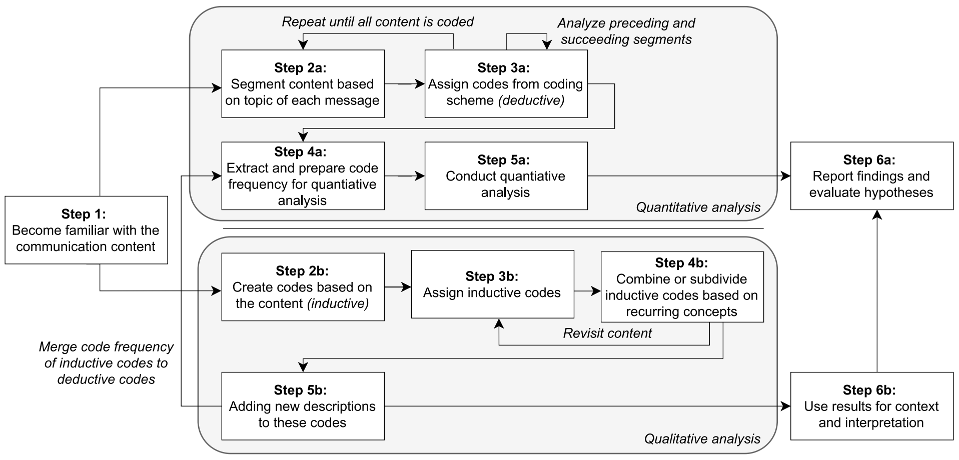
Repeat until all content is coded
Analyze preceding and succeeding segments
Step 2a: Segment content based on topic of each message
Step 3a: Assign codes from coding scheme (deductive)
Step 4a: Extract and prepare code frequency for quantitative analysis
Step 5a: Conduct quantitative analysis
Quantitative analysis
Step 6a: Report findings and evaluate hypotheses
Step 1: Become familiar with the communication content
Step 2b: Create codes based on the content (inductive)
Step 3b: Assign inductive codes
Step 4b: Combine or subdivide inductive codes based on recurring concepts
Revisit content
Merge code frequency of inductive codes to deductive codes
Step 5b: Adding new descriptions to these codes
Step 6b: Use results for context and interpretation
Qualitative analysis

Fig. 2. Overview of quantitative and qualitative analysis.

appendix (Ly et al., 2024), while another author performed the same calculations using Excel to check for inconsistencies.

We imported the altered Excel file into SPSS. Based on the hypotheses (as presented in Section 2.1), we had to choose between parametric and non-parametric tests. We performed a normality test, and in most cases, the significance of Shapiro–Wilk is below 0.05; thus, the data is not normally distributed and fails to meet the assumption for the t-test (Fay and Proschan, 2010). We, therefore, opted for the Mann–Whitney U test, the non-parametric alternative of the t-test. We calculated the effect size using Pearson's correlation coefficient. To interpret effect size, we adhere to the guidelines established by Benesty et al. (2009): $r < 0.3$ is considered a small effect, $0.3 \leq r < 0.5$ signifies a medium effect, and $r \geq 0.5$ indicates a large effect.

2.7.4. Survey design, sampling and analysis

We created a supplementary survey to validate or refute the findings from our qualitative and quantitative analyses. The survey questions are based on the findings of our analyses. However, we had to limit the number of questions to prevent respondent fatigue and only included questions where additional context was needed. The mapping of the findings to the survey questions can be found in Appendix B. Moreover, we grouped related questions, facilitating respondents to consider related questions collectively. We intentionally excluded questions that could lead to the identification of respondents, such as questions pertaining to specific roles. Despite the tendency for respondents to gravitate towards the middle option on a 5-point Likert scale, we chose this scale over a 7-point Likert scale for ease of comprehension (Joshi et al., 2015). The survey was created using Qualtrics due to its ability to address privacy and security concerns.

The respondents consisted of team members involved in the selected projects (see Table 3), with a primary emphasis on team members such as UI/UX designers, software architects, and test & quality employees, as our analysis (Section 3.3) has shown that most communication revolves around software construction, including testing and maintenance. The first and second authors distributed the survey via email to the team members. One week later, the authors sent a follow-up email as a reminder. In total, there were 24 respondents, with 14 for PrWf and 10 for PrAg. Two PrWf respondents did not answer all questions; thus, the statistical analyses were conducted based on the responses received for each question. We used the *results* and *filters* functionality of Qualtrics to analyze the survey results. Furthermore, we removed two survey questions due to inadequate formulation, as they did not align with our intended objectives of assessing whether the project teams were using the industry adaptations of the Waterfall model and the Scrum framework. To elaborate, the question *"How well defined are the project requirements before the planning and execution of projects begin?"* did not address the distinction between the Waterfall's fixed, well-documented requirements before the development phase begins and Scrum's evolving requirements throughout the development process. Similarly, the question *"How flexible is your team in adjusting development activities in response to changes?"* does not specify how often or in what context the adjustments are made (i.e., phases/sprints), making it difficult to differentiate between the two approaches. Lastly, we created diverging stacked bar charts with the Vega editor[1] to present the survey results as they facilitate the comparison of responses.

3. Results and interpretation

We report on the results of the statistical tests concerning our hypotheses, as detailed in Section 2.1. The tables, including the quantitative results, can be found in Appendix A. The subsequent sections are ordered according to the sub-research questions, starting with the PMLC phases, followed by the SDLC phases, and concluding with speech act types. The findings are presented within gray blocks, labeled in ascending numerical order, with corresponding hypotheses enclosed in brackets for organization. The format is as follows: Finding ($H_{\text{hypothesis number}}$). Following each finding, we provide our interpretation. Subsequently, we present the survey results that provide additional insights regarding the findings obtained from the content analysis of the written communication. We either reject or fail to reject the hypotheses based on these results.

3.1. How does the employed development paradigm impact the content of written communication pertaining to the Project Management Life Cycle phases? (RQ1A)

We begin by investigating how the development paradigm influences communication content across various PMLC phases, examining differences in communication content between PrWf and PrAg projects. The results presented here are based on the statistics shown in Table A.7a.

Finding 1 (H_1): In our analyzed projects, the overall written communication content that pertains to the *planning* phase of the PMLC is significantly more frequent in PrWf ($p = 0.004$, $r = 0.48$), whereas the overall written communication content related to the *execution-monitoring-control* phase is significantly more frequent in PrAg ($p = 0.003$, $r = 0.49$).

[1] https://vega.github.io/.

Finding 1 provides empirical evidence confirming the principle of the Agile Manifesto (Fowler and Highsmith, 2001), which prioritizes *"Responding to change over following a plan"*. This approach is exemplified by quotes such as: *"The issue has its origins in the scanbox proposal, if the creditor is removed before you open the scan, the query returns nothing (204) causing the handling code to crash. Now the creditor is reset, and the user has to figure it out for themselves :)"* *(*PrAg*10, formal communication content)* and *"The prototype is not yet functional, our initial ideas were somewhat naive. I have moved the pull request out of draft status, the build is running now, and we'll see what it says"*. *(*PrAg*11, informal communication content).* Agile methods provide the flexibility to adapt the approach during the execution-monitoring-control phase, while Waterfall adheres to a plan-driven method that copes less well with unexpected outputs from intermediate processes (Dima and Maassen, 2018).

Finding 2 ($H_{1.1}$): In formal communication channels, projects from PrWf exhibit a significantly higher communication frequency than PrAg projects regarding the *planning* phase ($p = 0.019$, $r = 0.54$), while PrAg projects show higher frequency for communication contents that pertain to the *execution-monitoring-control* phase ($p = 0.017$, $r = 0.55$). **Finding 3 ($H_{1.2}$):** In informal communication channels, PrAg projects exhibit a greater communication frequency regarding the *execution-monitoring-control* phase ($p = 0.036$, $r = 0.49$) compared to PrWf projects.

Finding 1 seems to derive from formal communication, as Finding 2 reveals statistical significance within formal communication channels for the planning phase in PrWf projects and the execution-monitoring-control phase in PrAg projects. In informal communication channels (Finding 3), the higher emphasis on the execution phase for PrAg projects is confirmed, while the prevalence of planning-related contents in PrWf projects does not hold. The lack of significance in the planning phase can be attributed to planning typically occurring in dedicated planning meetings through formal communication channels. These meetings have not been captured and are not part of our dataset. Once planning has been completed, project team members clearly understand task allocation. However, during the execution phase, communication may still be necessary for collaboration or coordination, which can take place via formal or informal channels (see Sections 2.3.2–2.3.1 for a description of the development methods).

Finding 4 ($H_{1.1}$): The communication contents in formal channels that pertain to the *closing* phase are more frequent in PrWf projects than in PrAg projects ($p = 0.046$, $r = 0.46$).

Finding 4 may be explained by the activities performed in the closing phase, involving the handover of project deliverables, the transfer of documentation, and the communication of project closure (Westland, 2007). These communications are likely to occur through formal channels as these channels facilitate record-keeping throughout the entire project life cycle, ensuring a trail of communication and decision-making. This record can assist the team members responsible for documentation by providing a reference point for addressing any questions, as exemplified by quotes such as *"Ticket approved @documentation, release notes, and help need to be updated (to my knowledge, at least 'Theme' should be 'Content on InSite and OutSite'). Possibly also courses?" (*PrWf*5, formal communication content)* and *"[Documentation employee], see these videos. Could you take care of these? Please coordinate the content with [product manager]"*. (PrWf9, formal communication content). Moreover, as per Table 1, PrWf is in the maturity stage of the Product Life Cycle and has a larger customer base, thus prioritizing client documentation, while PrAg is in the introduction stage, where the principle of the Agile Manifesto is applicable: *"Working software over comprehensive documentation"*.

Upon examining the survey findings, we had to exclude one survey question related to planning from our analysis due to inadequate formulation (see Section 2.7.4). Consequently, we cannot confirm Finding 1; we cannot conclude that waterfall-developed projects prioritize the planning phase of the PMLC. However, as shown in Fig. 3, we find that the perceptions of how much communication pertains to the execution-monitoring-control phase partly align with Finding 1, suggesting that agile-developed projects tend to prioritize communication related to this phase. Therefore, we reject the hypothesis regarding the execution-monitoring-control phase:

- H_0^1**:** The content of written communication within a development team is not affected by the development paradigm for the execution-monitoring-control phase, $p = 0.004$, $r = 0.48$ (medium effect).

3.2. How does the formality of the communication channel impact the content of written communication pertaining to the Project Management Life Cycle phases? (RQ1B)

We analyze how the formality of the communication channel impacts the communication content by comparing the relative frequency of the associated PMLC phases. The results are based on the data in Table A.7b.

Finding 5 (H_2): Regarding the *initiation* phase, communication contents in informal channels are more frequent than those in formal channels ($p = 0.035$, $r = 0.35$). **Finding 6 ($H_{2.2}$):** The prevalence of informal channels for contents regarding the *initiation* phase is mostly due to PrAg ($p = 0.050$, $r = 0.43$).

In the initiation phase, stakeholders convene to define project goals, assess feasibility, and establish a foundation for subsequent project activities (Westland, 2007). The communication content regarding the initiation phase in informal channels ($\bar{x} = 30.92$) is more prevalent than in formal channels ($\bar{x} = 17.00$) (Finding 5), which may be explained by the need for flexibility, collaboration between stakeholders, and exploration inherent in the activities of this phase, which can be facilitated by informal channels.

In relation to Finding 6, wherein informal channels in PrAg show a higher communication frequency regarding the initiation phase than formal channels, this may be attributed to Agile's emphasis on collaboration, continuous refinement, and stakeholder involvement. Agile's iterative and adaptable approach encourages ongoing communication to achieve a shared understanding of evolving project requirements and integrate stakeholder feedback. Thus, these methods may motivate team members to engage more with informal channels, facilitating dynamic interactions that contribute to efficient decision-making and problem resolution.

Finding 7 ($H_{2.2}$): We identify a difference in the communication contents that pertains to the *execution-monitoring-control* phase for PrAg ($p = 0.028$, $r = 0.48$) between formal (73.78%) and informal channels (47.94%).

The majority of PrAg communication pertains to the execution-monitoring-control phase ($\bar{x} = 73.78\%$ in the formal channel and $\bar{x} = 47.94\%$ in the informal channel). This observation may be attributed to the close collaboration among team members involved in the development of PrAg.

When analyzing the survey findings, we can conclude that regarding the initiation phase (question 3 in Table B.10), communication content in informal channels is more prevalent than in formal channels. The respondents of PrAg and PrWf perceive informal channels as predominant regarding this phase. For PrWf, this consists of 30.77% *Teams* and

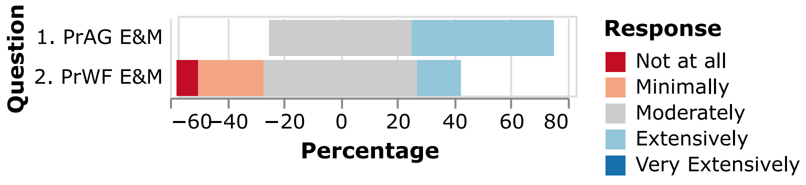

Fig. 3. Respondents' perception of the extent of communication in the 'Execution-Monitoring-Control' phase with PrAg 50% for Moderately and 50% Extensively and PrWf 7.69% Not at All, 23.08% Minimally, 53.85% Moderately, 15.38% Extensively.

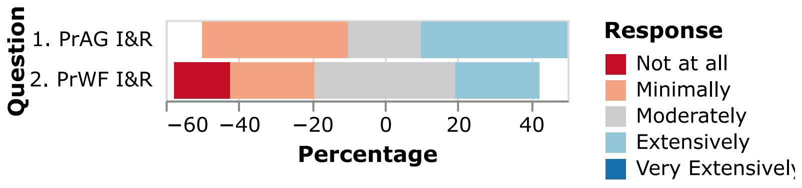

Fig. 4. Respondents' perception of the extent of communication in the 'Initiation' phase with PrAg 40% Minimally, 20% Moderately and 40% Extensively and PrWf 15.38% Not at all, 23.08% Minimally, 38.46% Moderately and 23.08% Extensively.

69.23% oral communication. Notably, a substantial portion of communication takes place orally, which is not part of our dataset. For PrAg, 30% indicate *Teams* as the primary channel, 50% oral communication, and 20% other written communication channels, which aligns with Findings 5 and 6. Fig. 4 suggests that PrAg respondents perceive a higher level of communication regarding the initiation phase than PrWf respondents. The results from the survey confirm the statistical outputs, and we, therefore, reject two hypotheses regarding the initiation phase based on the following p-values and effect sizes (r):

- H_0^2: The content of written communication regarding the initiation phase within a development team is not affected by the formality of the communication channel, $p = 0.035$, $r = 0.35$ (medium effect).
- $H_0^{2.2}$: The content of written communication regarding the initiation phase within an agile development team is not affected by the formality of the communication channel, $p = 0.050$, $r = 0.43$ (medium effect).

Regarding the execution-monitoring-control phase, respondents predominantly declare (survey question 5 in Table B.10) to use informal channels, with PrAg using 90% *Teams* and 10% oral communication, while PrWf declares to use 53.85% *Teams*, 30.77% oral communication, and 7.69% *Insite*. This difference may be attributed to cultural differences, with PrAg favoring *Teams* for communication, while PrWf uses both *Teams* and oral communication, with *Teams* being the predominant communication channel. This result contradicts our previous Finding 7, where formal channels have a higher frequency of communication. This contradiction may be explained by the presence of other data, such as one-on-one *Teams* chats, that we do not have access to due to privacy concerns. Thus, we fail to reject hypothesis $H_{2.2}$.

3.3. How does the employed development paradigm impact the content of written communication pertaining to the Software Development Life Cycle phases? (RQ2A)

In a similar vein to Section 3.1, we conduct a statistical analysis to investigate the impact of the development paradigm on communication content, with a specific emphasis on the phases of the SDLC. The results are derived from the data presented in Table A.8a. In the findings of this section, we examine whether communication pertaining to a specific SDLC phase occurred more frequently in PrAg or PrWf projects. This comparison is specified within brackets, including the respective p values and effect sizes (indicated with r).

Finding 8 (H_1): Our analysis reveals statistically significant differences with medium effect sizes in three of the five SDLC phases, specifically in *construction* (PrAg, $p = 0.050$, $r = 0.33$), *testing* (PrWf, $p = 0.005$, $r = 0.46$), and *maintenance* (PrWf, $p = 0.035$, $r = 0.35$).

PrAg has a remarkably higher communication frequency regarding the construction phase ($\bar{x} = 85.14\%$), while the other phases are infrequent. This prevalence presumably leads to differences in the testing and maintenance phases (Finding 8), as the PrAg communication content is heavily focused on software construction. Moreover, it is plausible that PrAg's increased communication during the construction phase is due to the integration of the construction and testing phases, as automated tests are written alongside the code. This result refines Finding 1 by explaining that PrAg's increased communication frequency regarding the PMLC execution-monitoring-control phase is mainly related to software construction aspects. To illustrate, the following PrAg quote applies to the PMLC execution phase and the SDLC construction phase: *"Can you propose how you intend to submit the append/hide/etc., then I can connect with that"*. (PrAg11 informal communication content).

Finding 9 ($H_{1.1}$): Analyzing only the content in formal communication channels, the same patterns of Finding 8 emerge for the *construction* (PrAg, $p = 0.048$, $r = 0.44$), *testing* (PrWf, $p = 0.008$, $r = 0.59$), and *maintenance* (PrWf, $p = 0.017$, $r = 0.53$) phases.

Most communication in formal channels pertains to the construction phase for both PrAg ($\bar{x} = 83.36\%$) and PrWf ($\bar{x} = 63.55\%$). The difference in construction may be explained by the higher frequency of communication regarding testing in PrWf than in PrAg. This pattern is also present in H_1.

We observe a difference in Findings 8 and 9 regarding the testing phase, with PrWf showing a higher communication frequency than PrAg: $\bar{x} = 22.95\%$ vs. $\bar{x} = 4.17\%$. The difference in testing may be attributed to the absence of automated tests in PrWf and the testing responsibility held by a dedicated testing department, thus requiring communication between the software architects and test & quality project team members in order to report defects, clarify issues, and verify resolutions. In contrast, PrAg relies on software architects writing unit, integration, and end-to-end tests, reducing the need for communication.

Written communication regarding maintenance (Finding 9) did not occur in the formal communication channel of PrAg, likely attributed

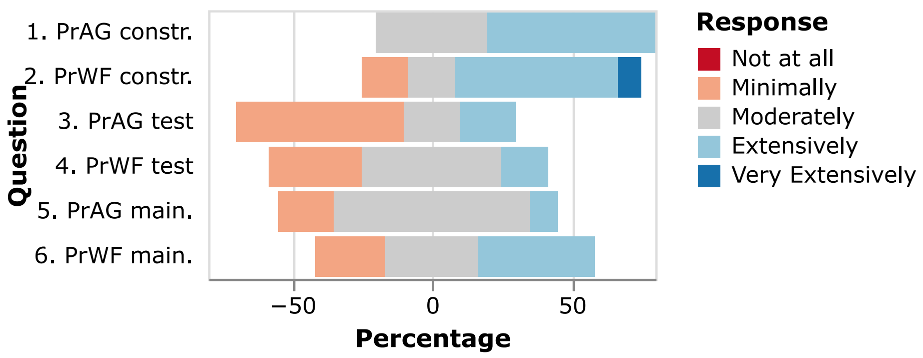

Fig. 5. Respondents' perception on the extent of communication related to 'Software Construction' with PrAg 40% Moderately and 60% Extensively and PrWf 16.67% Minimally, 16.67% Moderately, 58.33% Extensively, and 8.33% Very Extensively, 'Software Testing' with PrAg 60% Minimally, 20% Moderately, 20% Extensively and PrWf 33.33% Minimally, 50% Moderately, 16.67% Extensively, 'Software Maintenance' with PrAg 20% Minimally, 70% Moderately, 10% Extensively and PrWf 25% Minimally, 33.33% Moderately, 41.67% Extensively.

to its recent launch and smaller user base and the use of a continuous integration & delivery pipeline, allowing errors to be discovered before a release (Shahin et al., 2017) and thus not occurring in the formal communication channel. Conversely, as PrWf is in the Product Life Cycle's maturity stage with a substantial user base, most issues or feature requests are anticipated to be communicated through its formal communication channel. Yet, the absolute frequency is low, only $\bar{x}$ = 4.13%.

The survey results, shown in Fig. 5, indicate that both PrAg and PrWf perceive extensive communication related to software construction. This perception aligns with our earlier Finding 8 that PrAg and PrWf exhibit a higher communication frequency regarding the construction phase. However, respondents predominantly perceive informal channels as the primary communication channels (survey question 7 in Table B.10). For PrAg, 40% use *Teams*, and 60% rely on oral communication, while for PrWf, 33.33% use *Teams*, 58.33% opt for oral communication, and 8.33% use *Insite*. This discrepancy does not align with our previous Finding 9, which states that the higher communication frequency may be attributed to formal communication channels. Consequently, we fail to reject hypothesis $H_{1.1}$.

The observed difference in Finding 8, where PrWf exhibits a higher communication frequency than PrAg regarding the testing phase, is substantiated by respondents' perceptions. Respondents from PrWf perceive a higher level of testing-related communication compared to respondents from PrAg (survey question 7 in Table B.10). In the context of testing-related communication, PrWf predominantly uses *Insite* (25%), *Teams* (50%) and oral communication (25%), while PrAg relies on *Insite* (30%), *Teams* (60%), and oral communication (10%) as primary communication channels. This result does not align with Finding 9, which states that the higher communication frequency may be attributed to formal communication channels. Consequently, we fail to reject hypothesis $H_{1.1}$.

The survey results suggest that respondents from PrWf perceive a higher frequency of communication regarding software maintenance compared to those from PrAg, aligning with Finding 8. Notably, respondents from both groups reported a higher perceived level of communication on software maintenance (as indicated in survey question 7 of Table B.10) than our dataset revealed. This discrepancy could be attributed to the perceived primary use of informal communication channels. In PrAg, 70% of maintenance communication occurs through *Teams* and 30% through oral communication, while PrWf allocates 50% to oral communication, with the remaining 25% through *Insite* and 25% through *Teams*. This result contradicts Finding 9, suggesting that higher communication frequency may be attributed to formal channels. Consequently, we fail to reject hypothesis $H_{1.1}$.

Based on the data presented in this section, we reject three hypotheses, considering the following p-values and effect sizes (r):

- H_0^1: The content of written communication within a development team is not affected by the development paradigm for the software construction phase, $p = 0.050$, $r = 0.33$ (medium effect).
- H_0^1: The content of written communication within a development team is not affected by the development paradigm for the software testing phase, $p = 0.005$, $r = 0.46$ (medium effect).
- H_0^1: The content of written communication within a development team is not affected by the development paradigm for the software maintenance phase, $p = 0.035$, $r = 0.35$ (medium effect).

3.4. How does the formality of the communication channel impact the content of written communication pertaining to the Software Development Life Cycle phases? (RQ2B)

We statistically analyze whether the communication formality impacts the communication content that pertains to the SDLC phases. However, the results of the statistical tests presented in Table A.8b do not show any statistically significant differences for H_2, $H_{2.1}$, and $H_{2.2}$. In our analyzed projects, the communication formality does not seem to impact the frequency of the communication content for the SDLC phases.

3.5. How does the employed development paradigm impact the use of speech acts types in written communication? (RQ3A)

We examine the last aspect of communication content in our coding scheme: speech acts. The results presented here are based on Table A.9a. Although H_1 did not reveal any differences, analyzing the results split by formality revealed significant differences.

Finding 10 ($H_{1.1}$): In formal channels, *directives* are more prevalent for PrAg than for PrWf ($p = 0.004$, $r = 0.64$). **Finding 11 ($H_{1.2}$)** Within informal channels, *representatives* are more common for PrAg than for PrWf ($p = 0.017$, $r = 0.54$).

Finding 10 could be attributed to the Agile method adopted by PrAg, wherein team members collaborate closely to implement features. Quotes such as *"You should also introduce a failed event that is thrown if the action fails"*. (PrAg5 formal communication content) are classified as directives as they prompt the recipient to take action.

Similarly, for Finding 11, it is plausible that team members in PrAg may be more inclined to share work updates than the team members in PrWf throughout the day. For instance, statements such as *"Redirect when creating an event is now working, by the way"*. (PrAg10 informal communication content) are considered representatives due to their informative nature.

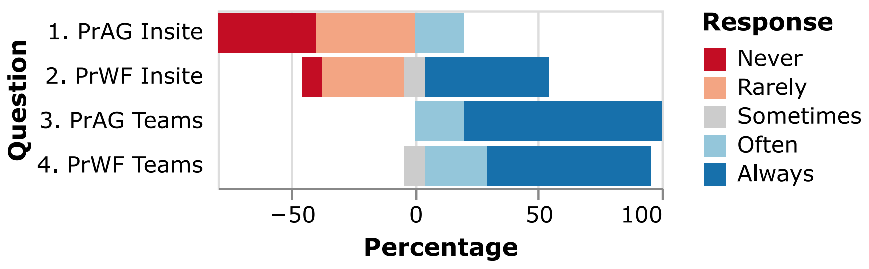

Fig. 6. Respondents' perception on asking questions with the intention of prompting colleagues to take action *(directives)* with the use of *Insite*: PrAg 40% Never, 40% Rarely, 20% Often, and PrWf, 8.33% Never, 33.33% Rarely, 8.33% Sometimes, 50% Always and use of *Teams* with PrAg 20% Often and 80% Always, and PrWf, 8.33% Sometimes, 25% Often, 66.67% Always'.

Finding 12 ($H_{1.2}$): We identify a significant difference for *commissives* in informal communication, with higher frequency in PrAg than in PrWf ($p = 0.033$, $r = 0.48$).

Commissives are speech acts intended to commit the speaker to a future course of action, exemplified by statements such as *"I will pick it up when I have some time in between"* (PrAg3 informal communication content) and *"So I will continue testing this once the error has been resolved"* (PrWf5 formal communication content). Building on the reasoning for Finding 10, the tasks have been allocated via formal channels, which might explain why the percentages are low: 5.37% for PrAg and 2.82% for PrWf.

According to the survey results regarding the use of directives in Fig. 6, PrAg respondents perceive that they rarely (40%) and never (40%) ask questions on *Insite* to prompt colleagues into action. In contrast, PrWf respondents perceive that they rarely (33.33%) and never (8.33%) use directives on *Insite*. This perception does not align with Finding 10, where directives are more prevalent in formal channels for PrAg compared to PrWf. Consequently, we fail to reject the hypothesis $H_{1.1}$.

Fig. 7 presents the perception of respondents using representatives. PrWf respondents perceive rarely (33.33%) and never (8.33%) use representatives on *Insite*. Regarding PrWf, this preference may stem from cultural norms within the case organization, where colleagues may favor oral communication via *Teams*. In contrast, PrAg respondents perceive rarely (40%) and never (50%) share information with colleagues on *Insite*. The perception that PrAg respondents do not share information in *Insite* might be attributed to their use of Agile standups, during which they share progress, discuss upcoming tasks, tackle obstacles, and coordinate efforts. Additionally, there is an integration with *Azure DevOps*, whereupon the finalization of a pull request, an automated message is generated in *Insite*.

Furthermore, PrAg respondents reported using representatives often (20%) and always (80%) in *Teams*, while PrWf respondents report using representatives sometimes (8.33%), often (25%), and always (66.67%) in *Teams*. These survey results support Finding 11, which shows that representatives are more prevalent in informal communication channels for PrAg than for PrWf. Thus, we reject the hypothesis for representatives:

- $H_0^{1.2}$: The content of written communication, conveyed as representatives, within a development team in informal communication channels is not affected by the development paradigm, $p = 0.017$, $r = 0.54$ (large effect).

3.6. How does the formality of the communication channel impact the use of speech acts types in written communication? (RQ3B)

Our final statistical analysis examines the effect of communication formality on the percentage of speech act types that occur in the communication content. The results are based on the data in Table A.9b.

Finding 13 (H_2): *Representatives* are more prevalent in formal channels than informal channels ($p < 0.001$, $r = 0.73$). **Finding 14 ($H_{2.1}$ & $H_{2.2}$):** *Representatives* are more frequent in formal communication for both PrWf ($p = 0.001$, $r = 0.77$) and PrAg ($p = 0.001$, $r = 0.69$).

Representatives are used to inform team members, as exemplified by quotes like *"This project only resolves the frequency and run tasks when another task is done"*. (PrWf2 informal communication content). As presented in Finding 13, representatives are more common in formal channels ($\bar{x} = 55.94\%$) than informal channels ($\bar{x} = 31.26\%$), which may be explained by the composition of the project teams of PrAg and PrWf, where members work in different departments. The use of representatives in formal channels ensures transparent communication by ensuring a trail of communication and decisions and can help in alignment between project team members.

Finding 15 (H_2): *Expressives* are more common in informal channels than formal channels ($p < 0.001$, $r = 0.69$). **Finding 16 ($H_{2.1}$ & $H_{2.2}$):** In particular, they (*expressives*) are more common in informal communication for both PrAg ($p < 0.001$, $r = 0.84$) and PrWf ($p = 0.016$, $r = 0.57$).

Findings 15 and 16 reveal that expressives are more common in informal channels than formal channels. Expressives occur as emoticons, GIFs, or utterances that express the speaker's psychological state, for instance, *"Looks good"* (PrWf9 informal communication content). Expressives are more prevalent in informal communication, as they are often used to prevent miscommunication, mainly due to the limitations of text-based communication. For instance, sarcasm or other forms of irony may not be easily conveyed through text. On the other hand, in formal communication, emoticons and GIFs are often considered unprofessional and may be avoided.

Finding 17 ($H_{2.1}$): In the context of PrWf, *directives* are more prevalent in informal channels than formal channels ($p < 0.019$, $r = 0.55$).

Finding 17 may be explained by the underlying action of directives, which is to encourage action. For instance, if a team member requires timely feedback on a proposed solution, it would be more efficient to approach this team member through informal channels (Giuffrida and Dittrich, 2013). For example, in our case company, if the team member being contacted is not logged into *Insite*, there is a risk that the ticket may get lost in their ticket bucket or not receive high priority. In contrast, sending a message via *Teams* can help ensure the ticket receives the necessary attention and action. The observed finding could also be attributed to the culture of PrWf, which leans towards the use of informal communication channels.

The survey results presented in Fig. 7 do not support Findings 13 and 14, which suggest that representatives are more frequent in formal

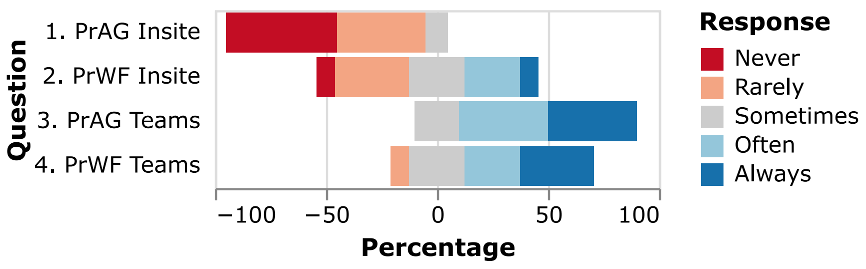

Fig. 7. Respondents' perception on sharing project-related information with colleagues *(representatives)* with the use of *Insite*: PrAg 50% Never, 40% Rarely, 10% Sometimes, and PrWf, 8.33% Never, 33.33% Rarely, 25% Sometimes, 25% Often, 8.33% Always and use of *Teams* with PrAg 20% Sometimes, 40% Often and 40% Always, and PrWf, 8.33% Rarely, 25% Sometimes, 25% Often, 33.33% Always'.

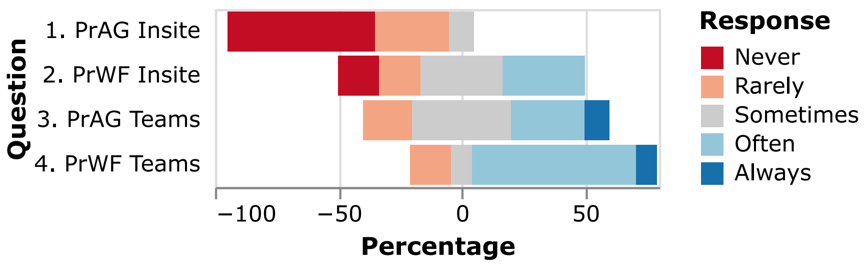

Fig. 8. Respondents' perception on using pictures, GIFs, or emoticons in project-related communication *(expressives)* with the use of *Insite*: PrAg 60% Never, 30% Rarely, 10% Sometimes, and PrWf, 16.67% Never, 16.67% Rarely, 33.33% Sometimes, 33.33% Often and use of *Teams* with PrAg 20% Rarely, 40% Sometimes, 30% Often, 10% Always, and PrWf, 16.67% Rarely, 8.33% Sometimes, 66.67% Often, 8.33% Always'.

channels than informal channels. Both PrWf and PrAg reported a higher perceived frequency of using representatives in informal channels. Thus, we fail to reject hypotheses H_2, $H_{2.1}$, and $H_{2.2}$.

Findings 15 and 16 indicate that expressives are more common in informal channels than formal channels, which is confirmed by the survey results as shown in Fig. 8. Respondents from PrWf and PrAg perceive using expressives more frequently in *Teams* than in *Insite*.

The survey results support Finding 17, indicating that directives are more prevalent in informal communication channels for PrWf. As shown in Fig. 6, respondents from PrWf indicate using directives more frequently on *Teams*: 8.33% sometimes, 25% often, and 66.67% always. In contrast, the reported frequencies on *Insite* are 8.33% never, 33.33% rarely, 8.33% sometimes, and 66.67% always.

Based on these findings, we reject hypotheses, considering the following p-values and effect sizes (r):

- H_0^2: The content of written communication within a development team is not affected by the formality of the communication channel for expressives $p < 0.001$, $r = 0.69$ (large effect).
- $H_0^{2.1}$: The content of written communication within a non-agile development team is not affected by the formality of the communication channel for expressives $p = 0.016$, $r = 0.57$ (large effect).
- $H_0^{2.1}$: The content of written communication within a non-agile development team is not affected by the formality of the communication channel for directives $p = 0.019$, $r = 0.55$ (large effect).
- $H_0^{2.2}$: The content of written communication within an agile development team is not affected by the formality of the communication channel for expressives $p < 0.001$, $r = 0.84$ (large effect).

4. Discussion

We address our sub-research questions in Sections 4.1, 4.1, and 4.3. Subsequently, we elaborate on the insights derived from our analysis in Sections 4.4 and 4.5. In Section 4.6, we compare the results of our study with previous studies. Additionally, we examine potential threats to the validity of our study in Section 4.7. Lastly, we discuss the implications and recommendations in Section 4.8.

4.1. How do the employed development paradigm and the formality of the communication channel impact the content of written communication pertaining to the Project Management Life Cycle phases? (RQ1A & RQ1B)

In our analyzed projects, the employed development paradigm influences written communication content pertaining to the execution-monitoring-control phase of the PMLC. Specifically, written communication content related to this phase is more prevalent in projects adopting an Agile approach (PrAg) compared to those following a Waterfall (PrWf) method [Finding 1] (see Table A.7a, H_1). The formality of the communication channel impacts the content related to the initiation phase of PMLC, with informal channels more frequently referring to this phase than formal channels [Finding 5] (see Table A.7b, H_2). The prevalence of informal channels regarding this phase is primarily attributed to PrAg projects [Finding 6] (see Table A.7b, $H_{2.2}$).

4.2. How do the employed development paradigm and the formality of the communication channel impact the content of written communication pertaining to the Software Development Life Cycle phases? (RQ2A & RQ2B)

The employed development paradigm influences the written communication content that pertains to the SDLC's construction, testing, and maintenance phases [Finding 8] (see Table A.8a, H_1). Most communication relates to the construction phase, presumably leading to differences in the testing and maintenance phases. This result provides further insight into the PMLC's findings, explaining that PrAg's communication content related to the execution-monitoring-control phase primarily pertains to software construction aspects. On the other hand, PrWf shows a greater frequency of communication (24.75%) regarding the testing phase than PrAg (2.72%). We attribute this difference in frequency to the reliance of PrWf on a dedicated testing department, which requires communication between software architects and test

& quality project team members to report defects, clarify issues, and verify resolutions. In contrast, PrAg relies on software architects to write unit, integration, and end-to-end tests, thus reducing the need for communication. In our case company, software architects take on responsibilities that go beyond the traditional scope (architectural design, internal and external communication) (Kruchten, 2008); depending on the workload, they can be tasked to write code and automated tests. We also found that communication regarding software maintenance is more prevalent in PrWf than in PrAg. In the projects we analyzed, the communication formality does not seem to impact the frequency of communication content for the SDLC phases.

4.3. *How do the employed development paradigm and the formality of the communication channel impact the use of speech acts types in written communication? (RQ3A & RQ3B)*

In relation to speech acts, the employed development paradigm influences the frequency of the written communication content conveyed as representatives. Representatives are more prevalent in informal channels for PrAg than PrWf [Finding 11] (see Table A.9a, $H_{1.2}$). The formality of communication impacts communication content, particularly concerning expressives and directives. Expressives are more common in informal channels than formal channels for both PrAg and PrWf [Finding 15 and 16] (see Table A.9b, H_2, $H_{2.1}$ & $H_{2.2}$). In the context of PrWf, directives are more prevalent in informal communication channels [Finding 17] (see Table A.9b, $H_{2.1}$).

4.4. *Misconceptions regarding informal communication in Agile and formal communication in Waterfall*

Our analysis of the communication content regarding the PMLC, the SDLC, and speech acts revealed a higher frequency of communication in formal channels, contradicting respondents' perceptions of informal channels being more frequently used [Findings 7, 9, 10, 13, 14]. This discrepancy may be attributed to the fact that Agile development enables teams to be flexible and adaptable to changing requirements, which is often perceived to be facilitated by informal communication according to Boehm and Turner (2004). However, a solid framework, including formal communication, e.g., sprint planning and daily standups, is required to facilitate this flexibility. This interpretation aligns with the study of Thummadi and Lyytinen (2020), which discovered that Agile projects often depended on informal requirements in the early stages. However, as projects progressed, there was a shift towards explicit documentation, occasionally resulting in more documentation than required. In contrast, Waterfall is a plan-driven approach characterized by extensive documentation and limited flexibility in modifying requirements after the planning phase has been completed, which is often perceived to be facilitated by formal communication (Royce, 1970). However, while the main processes, such as design, are clearly outlined, the specific execution within these processes is left undefined. The absence of process guidance may necessitate informal communication to clarify uncertainties, resolve issues swiftly, and ensure effective collaboration.

4.5. *Influence of Agile practices on communication patterns*

We found that for PrAg projects, over 85% of communication pertained to software construction [Finding 8] (see Table A.8a). This communication is influenced by Agile practices and sub-practices such as testing and continuous integration (i.e., DevOps). Agile practices often involve scheduled recurring interactions, which can result in increased communication among team members. In contrast, the sub-practices may lead to reduced communication when they automate some manual steps. In Agile testing, software developers write unit tests for their code, resulting in decreased communication needs compared to manual testing performed by a dedicated tester. In addition, continuous integration automates the process of integrating code into a central repository, where builds and tests are executed, thereby eliminating the need for developers to coordinate and communicate when they can merge their code into the final product.

4.6. *Comparison to related work*

Lin et al. (2016) surveyed 53 software developers to understand Slack adoption. They found that Slack serves personal, team-wide, and community-wide purposes, with team-wide purposes prevalent. Parra et al. (2022) employed Lin et al. (2016)'s coding scheme to classify the purpose of communication in 10,000 Gitter messages exchanged among developers. They found that 83% of the communication was intended to support activities directly related to the development of the system (i.e., team-wide purposes). Our study builds upon these findings; specifically, we focus on analyzing written communication within teams.

Mezouar et al. (2022) studied developers' use of Slack and Gitter chat rooms by analyzing 162 surveys and 21 follow-up interviews. They found similar purposes as Lin et al. (2016) while noting differences in reputation, quality, and timeliness of help, and group awareness. In line with Parra et al. (2022) and Mezouar et al. (2022) found that 66% of survey respondents using Slack and 63% using Gitter identified issue resolution pertaining to software development as predominant. Our study yielded similar results, as we observed a higher communication frequency regarding software construction for PrAg (85.14%) and PrWf (61.33%)

Silva et al. (2022) applied thematic analysis to manually identify software engineering themes from 87 Gitter chatrooms and examined if these themes were also present in 184 public Slack chat rooms. They discovered that the discussions mainly revolved around software development technologies and practices rather than the development process. Their findings are in contrast to those of previously mentioned studies and ours; we observed a higher communication frequency regarding the software construction phase, and our channels are project-related. This inconsistency may be attributed to their choice of chat rooms.

Shi et al. (2021) conducted an empirical study on developers' live chat in Gitter. They discovered that developers frequently discuss topics such as API usage and errors and that the direct/discussed answer pattern is the most frequent in communities. This finding is consistent with ours, as most communication is related to software construction, and representatives and directives are the most prevalent speech acts.

In contrast to the self-reported communication channel usage by developers in the studies by Lin et al. (2016) and Mezouar et al. (2022), other researchers Parra et al. (2022), Silva et al. (2022), and Shi et al. (2021) extracted and examined the written communication content from developer messages. Our approach differs from these studies in that we extract communication data from the internal channels of an organization. Moreover, we examine the communication of teams across 20 projects conducted within the same time frame, which goes beyond the scope of studies that solely focus on software developers. We also consider the development paradigm and formality of communication and complement our analysis with a survey.

The study by Stray et al. (2019) extracted written communication from the internal Slack channels of Geosoft, a large software development organization specializing in engineering. The study focused on four virtual teams, compromising a total of 30 team members, distributed equally at two sites, all involved in a project spanning from 2015 to 2017. Their manual open coding analysis of 500 messages revealed that approximately half of the messages were dedicated to problem-solving, with an average of 21% questions and 25% answers, primarily addressing technical questions and discussions surrounding potential solutions.

While we also perform coding on proprietary communication content, our study differs in several aspects. First, we analyze communication extracted from *Teams* and *Insite*. Secondly, we examine the written communication of 20 hybrid project teams with varying member sizes within a large organization specializing in enterprise application development. Third, we manually coded 3370 messages, exceeding the number analyzed by Stray et al. (2019)'s study. Fourth, our coding scheme is more fine-grained as we incorporate the PMLC, the SDLC, and speech acts to determine the purpose of the communication. Our results also indicate that directives and representatives are the predominant speech acts in our analyzed projects.

4.7. Threats to validity

We address potential threats to the validity of our study by considering construct validity, internal validity, external validity, and reliability, as outlined by Yin (2009) and Runeson and Höst (2009). These measures serve as guidelines for evaluating case study research.

Construct validity evaluates the extent to which the operational measures employed accurately capture and correspond with the conceptual variables and align with the research questions. To ensure that the project teams' development paradigms aligned with their actual practices, we provided detailed descriptions of how the teams operate in relation to their products and ensured that these descriptions align with existing literature. Moreover, two authors have firsthand experience with the development paradigms used in their respective products: the first author with projects for PrWf and the second author with projects for PrAg. The company processes and team members' self-identifications further support this distinction.

Furthermore, team members involved in the selected projects were asked to determine the extent to which team communication is related to various phases of the PMLC and the SDLC using a scale ranging from *"not at all, minimally, moderately, extensively to very extensively"* (see survey questions 2 and 6). While *"moderately"* might be interpreted as slightly positive, it was intentionally positioned as the midpoint of the scale to represent a moderate level of involvement rather than representing strict neutrality or an absence of opinion.

Internal validity refers to the risk of unidentified third factors influencing casual relationships when investigating the impact of one factor on another. Many factors can influence communication within teams, including, but not limited to, product maturity, type, team characteristics, and the COVID-19 pandemic.

Despite differences in product maturity and type between PrAg and PrWf, they are comparable in communication content analysis. First, although PrAg was launched in 2021, it has been under development for 11 years, compared to PrWf's 25 development years. Therefore, both organizational units are mature in terms of within-team communication. Second, despite differences in product types, both products are classified as enterprise software. We argue that it is unrealistic to expect researchers to find two almost identical products within the same organization that adopt different development paradigms.

Moreover, research has shown that team characteristics, such as age (Schloegel et al., 2016) and culture (Shachaf, 2008), can impact communication patterns. Considering the potential for diverse preferred communication styles among project teams, the content and frequency of communication could be influenced. To mitigate this threat, we sampled 20 projects, including 11 for PrAg and 9 for PrWf, with varying team compositions to prevent any team from being over-represented in communication patterns (Ly et al., 2024).

Team-based communication channels help maintain coherence in sharing work-related information, improving awareness, and supporting socialization. However, inconsistent usage or lack of adoption within teams can have the opposite effect (Forsgren and Byström, 2018). We found that *Teams* is the preferred channel for project-related communication for both PrAg and PrWf. Additionally, PrWf has a higher adoption rate of *Insite* than PrAg. This preference may be attributed to the fact that *Teams* facilitates more communication than written communication within project-related channels, including one-on-one written chats and oral communication. Instead of solely analyzing absolute numbers, we applied a mixed-methods approach and conducted a supplementary survey to gain insights and contextual understanding of our case study. We accept this threat of validity as inherent to the nature of case studies.

The analyzed projects occurred during the COVID-19 pandemic, with both organization units working in hybrid arrangements; these arrangements were already in place before the pandemic. The units made adjustments to their internal communication methods. PrAg intensified daily stand-ups and *Teams* communication to replace informal interactions, while PrWf used *Teams* voice calls for informal communication. This change possibly led to increased communication frequency in the channels, which could affect team communication dynamics and how information is exchanged and interpreted. The use of *Insite* remained unchanged. Nevertheless, our findings remain relevant, given the ongoing hybrid working mode of the units.

External validity assesses the degree to which findings are applicable beyond the specific context in which they were obtained. Although we analyzed the conversations of 20 projects from two communication channels of two products in terms of development paradigm, we do not argue for the generalizability of our findings to other products or communication channels. Additionally, the variations in how project teams in industry adapt the development paradigms limit the extent to which our findings can be generalized. Our single case study does not seek generalization of findings; instead, case studies facilitate analytical generalizations, whereby the findings are applied to other cases sharing similar characteristics, thus contributing to theory development (Runeson and Höst, 2009).

Reliability pertains to the replicability of data and analysis, regardless of the researchers involved. The first author coded the conversations. To reduce bias and enhance the reliability of our coding, we involved an independent coder in the early phases of our research; we discussed the disagreements as a means to obtain the coding scheme employed in this paper. While it would be preferable for the independent coder to label all the data, this was not feasible due to the time-intensive nature of applying our coding method. Consequently, the independent coder analyzed five projects. In terms of data disclosure, we cannot share the Nvivo file due to the presence of sensitive information. However, we provide access to the coding scheme, survey questions, SPSS files, Python script, and Excel files for transparency and to allow the reproduction of the statistical analyses (Ly et al., 2024).

4.8. Implications and recommendations for practitioners and researchers

4.8.1. For practitioners: implications and recommendations

Our method could guide the design of techniques, tools, and strategies to support effective and efficient communication practices within software organizations. Managers responsible for overseeing software product implementation can use our method to analyze the written communication content for operational management purposes. To start with, managers can evaluate whether the communication content aligns with their adopted development paradigm. Subsequently, they can use insights derived from the analysis to understand how the formality of communication channels impacts the content, enabling them to make informed decisions on choosing the proper communication channels for their teams. Consequently, managers can steer communication to better align with their objectives — determining where specific communication occurs and where it should ideally take place. Lastly, an in-depth understanding of the communication patterns of teams can assist in identifying potential bottlenecks or areas for improvement in the software development process. We acknowledge that analyzing team communication using our coding method is time-intensive. Therefore, we invite researchers to explore and develop more efficient methods for labeling large datasets, as exemplified by the work of Ahmed et al. (2024), to enable practitioners to put these methods into practice. Based on our findings, we provide two recommendations for practice:

1. *Providing operational guidance in the Waterfall development process (derived from Section 4.4):* We recommend implementing or improving structured process guidance in Waterfall projects by establishing policies, standards, best practices, and procedures. This guidance will reduce the reliance on informal communication to clarify uncertainties, facilitating more efficient collaboration and swifter issue resolution.
2. *Minimizing manual coordination in Agile projects (derived from Section 4.5):* We recommend leveraging automated sub-practices such as continuous integration and automated testing. These practices reduce the need for manual coordination and communication, enabling project team members to focus on higher-value interactions. This recommendation aligns with the common principles of Agile software development (such as Fowler's 11 continuous integration principles). However, we encourage teams not only to adopt these practices but also to understand and apply the underlying principles that make them effective.

4.8.2. *For researchers: implications and recommendations*

We provide empirical evidence on how within-team communication in a software development organization is affected by the development paradigm and communication channel formality. Based on our findings, we suggest two research directions worth exploring, guided by the following questions:

1. *How do communication patterns differ across organizations when comparing open source and proprietary software development projects?* Applying our method to analyze the communication content in other organizations is needed for theory building and generalizability of our findings. Furthermore, the accumulated theory can be used to design communication patterns, method fragments, or a combination of both, which software organizations can use to improve communication practices within their teams.
2. *To what extent does integrating additional sources of communication, such as automatic meeting transcripts and emails, improve the reliability and comprehensiveness of research findings on team communication dynamics?* Our study focused on written within-team communication across two communication channels. However, team members also communicate through other channels. Quotes such as *"In consultation with [...]",* (PrAg7) indicate that conversations took place, and decisions were made outside the studied channels. Therefore, they were not present in our data set. To gain a comprehensive understanding of team communication, it is important to consider other sources of communication. Due to privacy concerns, we were unable to collect oral communication, one-to-one chats, team meeting chats, emails, and WhatsApp conversations. We suggest developing a tool that enables colleagues to voluntarily upload communication content for research purposes.

5. Summary

In this paper, we performed statistical and content analyses on the written communication of 20 hybrid project teams within AFAS Software to investigate how the development paradigm and communication channel formality influence communication content and how this content pertains to the PMLC, the SDLC, and speech act types. Our results revealed that the employed development paradigm and communication formality influence written communication for various phases and speech act types. Contrary to common perceptions, our examined Agile projects seem to require a framework and formal communication to facilitate flexibility, while Waterfall projects rely on informal communication to address uncertainties, resolve issues, and ensure effective collaboration due to the lack of process guidance. However, it is important to note that the generalizability of our findings warrants further investigation. We found that internal communication channels serve as a valuable data source, which can provide managers with insights they can leverage for operational management.

CRediT authorship contribution statement

Delina Ly: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Michiel Overeem:** Writing – review & editing, Writing – original draft, Visualization, Validation, Conceptualization. **Sjaak Brinkkemper:** Writing – review & editing, Supervision. **Fabiano Dalpiaz:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Delina Ly reports a relationship with AFAS Software that includes: employment. Michiel Overeem reports a relationship with AFAS Software that includes: employment. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We express our gratitude to AFAS Software for their valuable data contributions, with special thanks to Dennis van Velzen for facilitating this research and Arjan van Hooff for retrieving *Insite* data. We also appreciate the contributions of the project team members working on PrAg and PrWf for sharing their time and insights. Furthermore, we extend our appreciation to Axel Vugts for his assistance in coding the communication of five projects. Finally, we thank Janine Lanting, Andrea Wohlgemuth, Ze Shi Li, the editors, and the anonymous reviewers for their feedback, which has contributed to the quality of the paper.

Appendix A. Quantitative results in table format

We used distribution measures to account for the variations in the data. In our tables, we use the following notation: $\bar{x}$ to represent the arithmetic mean, σ for the standard deviation, N for the number of samples, and U and p values for Mann–Whitney's U test, and r for the effect size calculated using Pearson's r. Green-colored cells indicate statistical significance at the $p \leq 0.05$ level. We did not report the median due to the limited sample size, as this would not be representative of the population.

A.1. *Development paradigm/Communication formality ⇒ PMLC phases*

PrWf4 and PrWf6 do not have any written communication related to the PMLC phases in informal channels, leading to the exclusion and a reduction in the sample size. Moreover, PrAg6 has no written communication about the PMLC phases in formal channels, resulting in a reduced sample size.

A.2. *Development paradigm/Communication formality ⇒ SDLC phases*

There are four projects where communication content did not pertain to the SDLC phases in informal channels: PrWf1, PrWf4, PrWf7, and PrWf8, leading to the exclusion and a reduction in the sample size.

A.3. *Development paradigm/Communication formality ⇒ Speech acts*

See Table A.9.

Table A.7
Statistical testing of the hypotheses on the communication content, focusing on the PMLC phases.

(a) H_1, $H_{1.1}$, and $H_{1.2}$: effect of a different development paradigm (IV) on the communication content, focusing on the PMLC phases.

Hypothesis 1 - PrWf vs. PrAg	PrWf			PrAg					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Initiation	24.14	18.82	16	23.50	22.72	21	156.0	0.711	–
Planning	35.93	30.80	16	8.53	13.12	21	77.5	0.004	0.48
Execution	26.46	26.73	16	60.24	30.04	21	72.0	0.003	0.49

Hypothesis 1.1 - PrWf vs. PrAg	Formal			Formal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Initiation	20.87	19.92	9	13.53	17.87	10	31.0	0.245	–
Planning	36.09	26.60	9	10.67	16.64	10	17.0	0.019	0.54
Execution	31.67	24.39	9	73.78	34.43	10	16.0	0.017	0.55
Closing	11.37	14.80	9	2.03	4.75	10	23.0	0.046	0.46

Hypothesis 1.2 - PrWf vs. PrAg	Informal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Initiation	28.33	17.87	7	32.57	23.56	11	35.5	0.785	–
Planning	35.71	37.80	7	6.58	9.29	11	24.0	0.160	–
Execution	19.76	30.00	7	47.94	19.69	11	15.5	0.036	0.49
Closing	16.19	30.76	7	12.91	23.94	11	38.0	0.957	–

(b) H_2, $H_{2.1}$, and $H_{2.2}$: effect of communication formality (MV) on the communication content, focusing on the PMLC phases.

Hypothesis 2 - PrWf & PrAg	Formal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Initiation	17.00	18.72	19	30.92	21.06	18	102.0	0.035	0.35
Planning	22.72	24.96	19	17.91	27.72	18	142.0	0.358	–
Execution	53.83	36.38	19	36.98	27.31	18	120.5	0.123	–

Hypothesis 2.1 - PrWf	Formal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Initiation	20.87	19.92	9	28.33	17.87	7	24.0	0.425	–
Planning	36.09	26.60	9	35.71	37.80	7	28.5	0.748	–
Execution	31.67	24.39	9	19.76	30.00	7	21.0	0.254	–
Closing	11.37	14.80	9	16.19	30.76	7	24.5	0.429	–

Hypothesis 2.2 - PrAg	Formal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Initiation	13.53	17.87	10	32.57	23.56	11	27.5	0.050	0.43
Planning	10.67	16.64	10	6.58	9.29	11	52.0	0.815	–
Execution	73.78	34.43	10	47.94	19.69	11	24.0	0.028	0.48
Closing	2.03	4.75	10	12.91	23.94	11	43.0	0.289	–

Table A.8
Statistical testing of the hypotheses on the communication content, focusing on the SDLC phases.

(a) H_1, $H_{1.1}$, and $H_{1.2}$: effect of a different development paradigm (IV) on the communication content, focusing on the SDLC phases.

Hypothesis 1 - PrWf vs. PrAg	PrWf			PrAg					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Design	4.21	9.10	14	7.86	14.92	22	115.0	0.181	–
Construction	61.33	30.52	14	85.14	18.95	22	94.0	0.050	0.33
Testing	24.75	26.73	14	2.72	6.48	22	75.5	0.005	0.46
Release	7.05	10.76	14	4.26	8.44	22	119.5	0.219	–
Maintenance	2.65	5.06	14	0.03	0.15	22	115.0	0.035	0.35

Hypothesis 1.1 - PrWf vs. PrAg	Formal			Formal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Design	2.85	4.19	9	4.84	9.05	11	47.5	0.864	–
Construction	63.55	21.14	9	83.36	19.79	11	24.0	0.048	0.44
Testing	22.95	13.82	9	4.17	8.84	11	17.0	0.008	0.59
Release	6.52	5.22	9	7.63	11.02	11	42.5	0.577	–
Maintenance	4.13	5.89	9	0.00	0.00	11	27.5	0.017	0.53

Hypothesis 1.2 - PrWf vs. PrAg	Informal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Design	6.67	14.91	5	10.87	19.11	11	12.0	0.075	–
Construction	57.33	45.85	5	86.92	18.86	11	23.0	0.610	–
Testing	28.00	43.82	5	1.27	2.31	11	22.5	0.515	–
Release	8.00	17.89	5	0.88	1.74	11	25.0	0.731	–
Maintenance	0.00	0.00	5	0.06	0.21	11	25.0	0.500	–

(b) H_2, $H_{2.1}$, and $H_{2.2}$: effect of communication formality (MV) on the communication content, focusing on the SDLC phases.

Hypothesis 2 - PrWf & PrAg	Formal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Design	3.94	7.18	20	9.56	17.52	16	115.0	0.130	–
Construction	74.44	22.28	20	77.67	31.60	16	151.5	0.785	–
Testing	12.62	14.61	20	9.62	26.06	16	131.0	0.310	–
Release	7.13	8.70	20	3.11	9.95	16	107.5	0.066	–
Maintenance	1.86	4.36	20	0.04	0.17	16	136.0	0.204	–

Hypothesis 2.1 - PrWf	Formal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Design	2.85	4.19	9	6.67	14.91	5	19.0	0.587	–
Construction	63.55	21.14	9	57.33	45.85	5	22.0	0.946	–
Testing	22.95	13.82	9	28.00	43.82	5	21.0	0.838	–
Release	6.52	5.22	9	8.00	17.89	5	13.0	0.187	–
Maintenance	4.13	5.89	9	0.00	0.00	5	12.5	0.095	–

Hypothesis 2.2 - PrAg	Formal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Design	4.84	9.05	11	10.87	19.11	11	32.5	0.060	–
Construction	83.36	19.79	11	86.92	18.86	11	57.5	0.843	–
Testing	4.17	8.84	11	1.27	2.31	11	60.0	0.968	–
Release	7.63	11.02	11	0.88	1.74	11	52.5	0.542	–
Maintenance	0.00	0.00	11	0.06	0.21	11	55.0	0.317	–

Table A.9
Statistical testing of the hypotheses on the communication content, focusing on speech acts.

(a) H_1, $H_{1.1}$, and $H_{1.2}$: effect of a different development paradigm (IV) on the communication content, focusing on the speech act types.

Hypothesis 1 - PrWf vs. PrAg	PrWf			PrAg					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Representative	44.29	20.99	18	43.03	17.15	22	191.5	0.860	–
Directive	36.72	14.47	18	42.91	13.54	22	128.0	0.057	–
Commissive	3.44	4.77	18	4.66	4.01	22	147.0	0.156	–
Expressive	15.55	17.30	18	9.39	11.72	22	154.5	0.231	–

Hypothesis 1.1 - PrWf vs. PrAg	Formal			Formal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Representative	61.47	10.10	9	51.41	20.88	11	25.0	0.062	–
Directive	28.50	5.81	9	43.42	17.76	11	12.0	0.004	0.64
Commissive	4.06	3.95	9	3.96	5.37	11	42.5	0.587	–
Expressive	5.96	6.87	9	1.21	2.57	11	27.5	0.067	–

Hypothesis 1.2 - PrWf vs. PrAg	Informal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Representative	27.11	13.05	9	34.65	5.20	11	18.0	0.017	0.54
Directive	44.93	16.11	9	42.41	8.29	11	43.0	0.621	–
Commissive	2.82	5.66	9	5.37	1.94	11	22.0	0.033	0.48
Expressive	25.14	19.54	9	17.57	11.61	11	37.0	0.342	–

(b) H_2, $H_{2.1}$, and $H_{2.2}$: effect of communication formality (MV) on the communication content, focusing on the speech act types.

Hypothesis 2 - PrWf & PrAg	Formal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Representative	55.94	17.28	20	31.26	10.04	20	28.5	<.001	0.73
Directive	36.71	15.44	20	43.54	12.13	20	141.0	0.110	–
Commissive	4.00	4.66	20	4.22	4.14	20	187.5	0.730	–
Expressive	3.35	5.40	20	20.97	15.71	20	40.5	<.001	0.69

Hypothesis 2.1 - PrWf	Formal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Representative	61.47	10.10	9	27.11	13.05	9	4.0	0.001	0.77
Directive	28.50	5.81	9	44.93	16.11	9	14.0	0.019	0.55
Commissive	4.06	3.95	9	2.82	5.66	9	24.0	0.120	–
Expressive	5.96	6.87	9	25.14	19.54	9	13.5	0.016	0.57

Hypothesis 2.2 - PrAg	Formal			Informal					
	$\bar{x}$	σ	N	$\bar{x}$	σ	N	U	p	r
Representative	51.41	20.88	11	34.65	5.20	11	11.0	0.001	0.69
Directive	43.42	17.76	11	42.41	8.29	11	51.0	0.532	–
Commissive	3.96	5.37	11	5.37	1.94	11	38.0	0.137	–
Expressive	1.21	2.57	11	17.57	11.61	11	2.0	<.001	0.84

Table B.10
Mapping findings to survey questions.

No.	Category	Finding no.	Survey Question(s)
1	Demographic	N/A	In which software development project team do you work?
2	PMLC	1	Please determine how much team communication relates to project execution and monitoring. Excluded question regarding planning.
–	PMLC	2 & 3	*No survey question.*
–	PMLC	4	*No survey question.*
3 & 4	PMLC	5 & 6	What is the primary communication channel used for team communication in relation to project initiation and requirements? Please determine how much team communication relates to project initiation and requirements.
5	PMLC	7	What is the primary communication channel used for team communication in relation to project execution and monitoring?
6	SDLC	8	Please determine how much of the team communication relates to software construction, software testing, and software maintenance.
7	SDLC	9	What is the primary communication channel used for team communication in relation to software construction, software testing, and software maintenance?
8 & 9	Speech acts	10 & 11	How frequently do you ask questions with the intention of prompting your colleagues to take action in *Insite* (in the comments) and *Teams*? How frequently do you share project-related information with your colleagues in *Insite* (in the comments) and *Teams*?
–	Speech acts	12	*No survey question.*
10 & 11	Speech acts	13 & 14	How frequently do you share project-related information with your colleagues in *Insite* (in the comments) and *Teams*? How often do you use pictures, GIFs, or emoticons in your project-related communication in *Insite* (in the comments) and *Teams*?
12	Speech acts	15	How frequently do you share project-related information with your colleagues in *Insite* (in the comments) and *Teams*?
13	Speech acts	16	How often do you use pictures, GIFs, or emoticons in your project-related communication in *Insite* (in the comments) and *Teams*?
14	Speech acts	17	How frequently do you ask questions with the intention of prompting your colleagues to take action in *Insite* (in the comments) and *Teams*?
15	General	N/A	Is there additional information you would like to provide in relation to your responses to questions on this page? *(repeating)*

Appendix B. Quantitative and qualitative findings mapping to survey questions

See Table B.10.

Data availability

The data that has been used is confidential.

References

Ahmed, T., Devanbu, P., Treude, C., Pradel, M., 2024. Can LLMs Replace Manual Annotation of Software Engineering Artifacts? http://dx.doi.org/10.48550/arXiv.2408.05534, arXiv preprint arXiv:2408.05534.

Benesty, J., Chen, J., Huang, Y., Cohen, I., 2009. Pearson Correlation Coefficient. In: Noise Reduction in Speech Processing. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–4. http://dx.doi.org/10.1007/978-3-642-00296-0_5.

Boehm, B.W., Turner, R., 2004. Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Professional.

Bourque, P., Fairley, R.E., 2014. Guide to the Software Engineering Body of Knowledge - SWEBOK V3.0. IEEE Computer Society, URL: www.swebok.org.

Brinkkemper, S., 1996. Method engineering: engineering of information systems development methods and tools. Inf. Softw. Technol. 38 (4), 275–280. http://dx.doi.org/10.1016/0950-5849(95)01059-9.

Cao, H., Folan, P., 2012. Product life cycle: the evolution of a paradigm and literature review from 1950–2009. Prod. Plan. Control 23 (8), 641–662. http://dx.doi.org/10.1080/09537287.2011.577460.

Dima, A.M., Maassen, M.A., 2018. From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management. J. Int. Stud. (2071-8330) 11 (2), 315–326. http://dx.doi.org/10.14254/2071-8330.2018/11-2/21.

Dorairaj, S., Noble, J., Malik, P., 2011. Effective Communication in Distributed Agile Software Development Teams. In: Sillitti, A., Hazzan, O., Bache, E., Albaladejo, X. (Eds.), Agile Processes in Software Engineering and Extreme Programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 102–116. http://dx.doi.org/10.1007/978-3-642-20677-1_8.

Fay, M.P., Proschan, M.A., 2010. Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. Stat. Surv. 4, 1–39. http://dx.doi.org/10.1214/09-SS051.

Forsgren, E., Byström, K., 2018. Multiple social media in the workplace: Contradictions and congruencies. Inf. Syst. J. 28 (3), 442–464. http://dx.doi.org/10.1111/isj.12156.

Fowler, M., Highsmith, J., 2001. The Agile Manifesto. Softw. Dev. 9 (8), 28–35, URL: https://agilemanifesto.org/.

Galster, M., Angelov, S., Meesters, M., Diebold, P., 2016. A Multiple Case Study on the Architect's Role in Scrum. In: Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., Felderer, M., Amasaki, S., Mikkonen, T. (Eds.), Product-Focused Software Process Improvement. Springer International Publishing, Cham, pp. 432–447.

Giuffrida, R., Dittrich, Y., 2013. Empirical studies on the use of social software in global software development – A systematic mapping study. Inf. Softw. Technol. 55 (7), 1143–1164. http://dx.doi.org/10.1016/j.infsof.2013.01.004.

Houck, B., Yelin, H., Butler, J., Forsgren, N., McMartin, A., 2023. The Best of Both Worlds: Unlocking the Potential of Hybrid Work for Software Engineers. URL: https://www.microsoft.com/en-us/research/uploads/prod/2023/05/BestOfBothWorlds.pdf.

Hubbard, M., Bailey, M.J., Hess, D., Hellebro, M., 2021. Communicating in Teams. In: Mastering Microsoft Teams: End User Guide to Practical Usage, Collaboration, and Governance. Apress, Berkeley, CA, pp. 49–72. http://dx.doi.org/10.1007/978-1-4842-6898-8_3.

Joshi, A., Kale, S., Chandel, S., Pal, D.K., 2015. Likert scale: Explored and explained. Curr. J. Appl. Sci. Technol. 7 (4), 396–403. http://dx.doi.org/10.9734/BJAST/2015/14975.

Kittlaus, H.-B., Fricker, S.A., 2017. Software Product Management: The ISPMA-Compliant Study Guide and Handbook, vol. 298, Springer Berlin, Heidelberg, http://dx.doi.org/10.1007/978-3-642-55140-6.

Kraut, R.E., Fish, R.S., Root, R.W., Chalfonte, B.L., Oskamp, I.S., Spacapan, S., 1990. Informal Communication in Organizations: Form, Function, and Technology. In: Human Reactions to Technology: Claremont Symposium on Applied Social Psychology. p. 199.

Kruchten, P., 2008. What do software architects really do? J. Syst. Softw. 81 (12), 2413–2416. http://dx.doi.org/10.1016/j.jss.2008.08.025, Best papers from the 2007 Australian Software Engineering Conference (ASWEC 2007), Melbourne, Australia, April 10-13, 2007.

Landis, J.R., Koch, G.G., 1977. The Measurement of Observer Agreement for Categorical Data. Biometrics 33, 159–174.

Leffingwell, D., 2010. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional.

Lin, B., Zagalsky, A., Storey, M.-A., Serebrenik, A., 2016. Why Developers Are Slacking Off: Understanding How Software Teams Use Slack. In: Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion. In: CSCW '16 Companion, Association for Computing Machinery, New York, NY, USA, pp. 333–336. http://dx.doi.org/10.1145/2818052.2869117.

Ly, D., Overeem, M., Brinkkemper, S., Dalpiaz, F., 2024. Online Appendix of the Paper "The Power of Words in Agile vs. Waterfall Development: Written Communication in Hybrid Software Teams". http://dx.doi.org/10.5281/zenodo.13894086.

Mezouar, M.E., da Costa, D.A., German, D.M., Zou, Y., 2022. Exploring the Use of Chatrooms by Developers: An Empirical Study on Slack and Gitter. IEEE Trans. Softw. Eng. 48 (10), 3988–4001. http://dx.doi.org/10.1109/TSE.2021.3109617.

Morales-Ramirez, I., Kifetew, F.M., Perini, A., 2019. Speech-acts based analysis for requirements discovery from online discussions. Inf. Syst. 86, 94–112. http://dx.doi.org/10.1016/j.is.2018.08.003.

Mortada, M., Ayas, H.M., Hebig, R., 2020. Why do Software Teams Deviate from Scrum? Reasons and Implications. In: Proceedings of the International Conference on Software and System Processes. pp. 71–80.

Nolan, A., White, R., Soomro, M., Dopamu, B.C., Yilmaz, M., Solan, D., Clarke, P., 2021. To Work from Home (WFH) or Not to Work from Home? Lessons Learned by Software Engineers During the COVID-19 Pandemic. In: Yilmaz, M., Clarke, P., Messnarz, R., Reiner, M. (Eds.), Systems, Software and Services Process Improvement. Springer International Publishing, Cham, pp. 14–33. http://dx.doi.org/10.1007/978-3-030-85521-5_2.

Parra, E., Alahmadi, M., Ellis, A., Haiduc, S., 2022. A comparative study and analysis of developer communications on Slack and Gitter. Empir. Softw. Eng. 27 (40), 1–33. http://dx.doi.org/10.1007/s10664-021-10095-1.

Patton, M.Q., 2002. Qualitative Research & Evaluation Methods. Sage Publications.

Qadir, A., Riloff, E., 2011. Classifying Sentences as Speech Acts in Message Board Posts. In: Barzilay, R., Johnson, M. (Eds.), Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Edinburgh, Scotland, UK., pp. 748–758.

Royce, W.W., 1970. Managing the development of large software systems. In: Proceedings of the 9th International Conference on Software Engineering. IEEE Computer Society Press, pp. 328–338, URL: https://dl.acm.org/doi/pdf/10.5555/41765.41801.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131–164. http://dx.doi.org/10.1007/s10664-008-9102-8.

Schloegel, U., Stegmann, S., Maedche, A., Dick, R., 2016. Reducing age stereotypes in software development: The effects of awareness- and cooperation-based diversity interventions. J. Syst. Softw. 121, 1–15. http://dx.doi.org/10.1016/j.jss.2016.07.041.

van der Schuur, H., van de Ven, E., de Jong, R., Schunselaar, D., Reijers, H.A., Overeem, M., de Graaf, M., Jansen, S., Brinkkemper, S., 2017. NEXT: Generating Tailored ERP Applications from Ontological Enterprise Models. In: Poels, G., Gailly, F., Serral Asensio, E., Snoeck, M. (Eds.), Proceedings of the Practice of Enterprise Modeling. Springer International Publishing, Cham, pp. 283–298. http://dx.doi.org/10.1007/978-3-319-70241-4_19.

Schwaber, K., 1997. SCRUM Development Process. In: Sutherland, J., Casanave, C., Miller, J., Patel, P., Hollowell, G. (Eds.), Business Object Design and Implementation. Springer London, London, pp. 117–134. http://dx.doi.org/10.1007/978-1-4471-0947-1_11.

Schwaber, K., Sutherland, J., 2020. The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game. URL: https://www.scrum.org/resources/scrum-guide.

Searle, J.R., 1969. Speech Acts: An Essay in the Philosophy of Language, vol. 626, Cambridge University Press.

Searle, J.R., 1979. Expression and Meaning: Studies in the Theory of Speech Acts. Cambridge University Press.

Shachaf, P., 2008. Cultural diversity and information and communication technology impacts on global virtual teams: An exploratory study. Inf. Manag. 45 (2), 131–142. http://dx.doi.org/10.1016/j.im.2007.12.003.

Shahin, M., Ali Babar, M., Zhu, L., 2017. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. IEEE Access 5, 3909–3943. http://dx.doi.org/10.1109/ACCESS.2017.2685629.

Shi, L., Chen, X., Yang, Y., Jiang, H., Jiang, Z., Niu, N., Wang, Q., 2021. A first look at developers' live chat on Gitter. In: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. In: ESEC/FSE 2021, Association for Computing Machinery, New York, NY, USA, pp. 391—-403. http://dx.doi.org/10.1145/3468264.3468562.

Silva, C.C., Galster, M., Gilson, F., 2022. A qualitative analysis of themes in instant messaging communication of software developers. J. Syst. Softw. 192, 111397. http://dx.doi.org/10.1016/j.jss.2022.111397.

de Souza Santos, R., Adisaputri, G., Ralph, P., 2023. Post-pandemic Resilience of Hybrid Software Teams. In: 2023 IEEE/ACM 16th International Conference on Cooperative and Human Aspects of Software Engineering. CHASE, IEEE, pp. 1–12. http://dx.doi.org/10.1109/CHASE58964.2023.00009.

de Souza Santos, R.E., Ralph, P., 2022. A grounded theory of coordination in remote-first and hybrid software teams. In: Proceedings of the 44th International Conference on Software Engineering. ICSE '22, Association for Computing Machinery, New York, NY, USA, pp. 25–35. http://dx.doi.org/10.1145/3510003.3510105.

Storey, M.-A., Singer, L., Cleary, B., Figueira Filho, F., Zagalsky, A., 2014. The (R) Evolution of social media in software engineering. In: Future of Software Engineering Proceedings. In: FOSE 2014, Association for Computing Machinery, New York, NY, USA, pp. 100–116. http://dx.doi.org/10.1145/2593882.2593887.

Stray, V., Moe, N.B., Noroozi, M., 2019. Slack Me If You Can! Using Enterprise Social Networking Tools in Virtual Agile Teams. In: 2019 ACM/IEEE 14th International Conference on Global Software Engineering. ICGSE, pp. 111–121. http://dx.doi.org/10.1109/ICGSE.2019.00031.

Sutherland, J., Sutherland, J., 2014. Scrum: The Art of Doing Twice the Work in Half the Time. Currency.

Taylor, J., 2003. Managing Information Technology Projects: Applying Project Management Strategies to Software, Hardware, and Integration Initiatives. American Management Association.

Thummadi, B.V., Lyytinen, K., 2020. How Much Method-in-Use Matters? A Case Study of Agile and Waterfall Software Projects and their Design Routine Variation. J. Assoc. Inf. Syst. 21 (4), 7.

Törlind, P., Larsson, A., 2002. Supporting Informal Communication in Distributed Engineering Design Teams. In: International CIRP Design Seminar.

Vinutha, H., Poornima, B., Sagar, B., 2018. Detection of Outliers Using Interquartile Range Technique from Intrusion Dataset. In: Information and Decision Sciences: Proceedings of the 6th International Conference on FICTA. Springer, pp. 511–518.

Walton, D., 2014. Abductive Reasoning. University of Alabama Press.

van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L., 2006. Towards a Reference Framework for Software Product Management. In: 14th IEEE International Requirements Engineering Conference. RE'06, IEEE, pp. 319–322. http://dx.doi.org/10.1109/RE.2006.66.

Weilkiens, T., Lamm, J.G., Roth, S., Walker, M., 2022. Model-Based System Architecture. John Wiley & Sons, http://dx.doi.org/10.1002/9781119051930.

West, D., 2016. Product Owner vs. Product Manager. URL: https://www.scrum.org/resources/product-owner-vs-product-manager.

Westland, J., 2007. The Project Management Life Cycle: A Complete Step-By-Step Methodology for Initiating Planning Executing and Closing the Project. Kogan Page Publishers.

Wohlin, C., 2021. Case Study Research in Software Engineering—It is a Case, and it is a Study, but is it a Case Study? Inf. Softw. Technol. 133, 106514. http://dx.doi.org/10.1016/j.infsof.2021.106514.

Wohlin, C., Rainer, A., 2022. Is it a case study?—A critical analysis and guidance. J. Syst. Softw. 192, 111395. http://dx.doi.org/10.1016/j.jss.2022.111395.

Wood, A., Rodeghero, P., Armaly, A., McMillan, C., 2018. Detecting speech act types in developer question/answer conversations during bug repair. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. In: ESEC/FSE 2018, Association for Computing Machinery, New York, NY, USA, pp. 491–502. http://dx.doi.org/10.1145/3236024.3236031.

Xu, L., Brinkkemper, S., 2007. Concepts of product software. Eur. J. Inf. Syst. 16 (5), 531–541. http://dx.doi.org/10.1057/palgrave.ejis.3000703.

Yin, R.K., 2009. Case Study Research: Design and Methods, vol. 5, Sage Publications.

Delina Ly is a software engineer at VX Company. As an external Ph.D. candidate at Utrecht University, she conducts research on deriving business insights from software engineering artifacts.

Michiel Overeem is an engineering manager at AFAS Software. He received his Ph.D. from Utrecht University, where he conducted research on upgrading model-driven, cloud-based software.

Sjaak Brinkkemper is a full professor of organization and information at the Department of Information and Computing Sciences of Utrecht University, the Netherlands. He leads a group of about twenty researchers specialized in product software development and entrepreneurship. In essence, he studies the implications of market conditions on software development methods and processes. He received the best paper awards for several papers, and he has given keynote talks at various conferences. He has been awarded the IFIP Silver Core Medal. He is a member of the steering committee of CAiSE, ECIS, ICSOB, and IWSPM. He is a co-founder of the International Software Product Management Board.

Fabiano Dalpiaz is a full professor of software production at Utrecht University in the Netherlands. He is the principal investigator in the department's Requirements Engineering lab, where the team blends artificial intelligence with information visualization to increase the quality of the requirements engineering process and artifacts. His research is often validated in vivo through collaborations with the software industry. He acted as program co-chair of RE 2023, REFSQ 2021, and RCIS 2020. He was the organization chair of the REFSQ 2018 conference, and he is an associate editor for the Requirements Engineering Journal and the Business & Information Systems Engineering Journal.