Defining Frames to Structure Agile Development in Hybrid Settings - A Multi-Case Interview Study

Nils Prenner
Leibniz Universität Hannover
Software Engineering Group
Germany
nils.prenner@inf.uni-hannover.de

Jil Klünder
Leibniz Universität Hannover
Software Engineering Group
Germany
jil.kluender@inf.uni-hannover.de

Kurt Schneider
Leibniz Universität Hannover
Software Engineering Group
Germany
kurt.schneider@inf.uni-hannover.de

ABSTRACT

Companies often combine agile and plan-based methods to so-called hybrid development approaches to benefit from the advantages of both. Recent research highlights conflicts introduced when combining agile and plan-based approaches in the different phases of the software lifecycle. For example, using both agile and plan-based methods during the requirements engineering of a project requires a decision on how many requirements should be gathered up-front and how many can be gathered during the runtime of a project. These conflicts need to be solved in order to construct a successful development approach. In order to investigate why the conflicts exist, how they are addressed in industry, and how they are related to each other, we performed a multi-case interview study with 15 practitioners. Our results reveal that the conflicts exist because companies use plan-based approaches to structure their agile development and define spaces of freedom and flexibility at the same time. From this insight and our results, we derive a theory that shows how companies structure their development stepwise by defining frames.

CCS CONCEPTS

• **Software and its engineering** → **Agile software development**; **Waterfall model**; *Requirements analysis*; *Software implementation planning*; *Software architectures.*

KEYWORDS

Hybrid development approaches, plan-based development, agile development, multi-case interview study

ACM Reference Format:
Nils Prenner, Jil Klünder, and Kurt Schneider. 2022. Defining Frames to Structure Agile Development in Hybrid Settings - A Multi-Case Interview Study. In *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering (ICSSP'22), May 20–22, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3529320.3529324

ICSSP'22, May 20–22, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9674-5/22/05.
https://doi.org/10.1145/3529320.3529324

1 INTRODUCTION

On the one hand, software companies want to be adaptable to changing requirements and need to deal with uncertain requirements [1, 8]. Further, they want to develop new features fast and have a short time-to-market [17]. Therefore, companies increasingly implement agile methods [1, 8]. On the other hand, companies have to satisfy safety and security requirements, manage a large-scale development, and need planning stability [3, 5]. Because of these reasons, companies also use plan-based methods and combine them with agile methods to hybrid development approaches [21]. Kuhrmann et al. [12] define hybrid approaches as any combination of agile or plan-based methods. This includes also development approaches where only agile methods are combined with each other. These approaches are less problematic since they do not need to combine opposite principles [21]. In this paper, we want to analyze how to deal with these opposite principles, thus focusing on development approaches where agile and plan-based methods are combined, i.e., a subset of hybrid approaches. We call these approaches agile-plan-based hybrid approaches (short: APH approaches) [21].

Problem Statement: The combination of agile and plan-based development approaches leads to conflicts in the development process. The conflicts arise because companies need agile and plan-based methods similarly for the same activity [1] and have to decide how much emphasis they have to put on either agile or plan-based methods [21]. We identified conflicts in the requirements engineering, architecture, planning, coordination, and documentation activity [21]. For example, the conflict in the requirements engineering activity arises between up-front and continuous requirements analysis. Plan-based approaches strive for an up-front requirements engineering because it clarifies the project. Agile development approaches use an agile requirements engineering to react to changes and to deal with uncertain requirements. In APH approaches, both kinds of requirements engineering approaches are used in combination, requiring to find the right balance. If the up-front requirements engineering is used too much, requirements may are gathered that are wrong. Too few up-front requirements engineering hinders the development team from getting a proper understanding of the project. Therefore, a successful development approach requires a compromise between agile and plan-based practices. This applies to each of the above-mentioned activities.

In previous work [21], we identified a research gap regarding how these compromises can be reached and which additional practices are necessary to properly combine agile and plan-based approaches.

[1] An activity describes a collection of tasks and practices that have the same goal, e.g. requirements engineering.

Objective: In this paper, we want to better understand why the conflicts exist and how they are addressed by companies in order to guide researchers and practitioners. We also want to reach an overall theory of how the conflicts are connected.

Contribution: To investigate the conflicts and to build a theory about their existence and their solution, we applied Grounded Theory. Our data collection is based on a multi-case interview study with 15 participants. Our results show that the conflicts exist because the companies want to structure their agile development and define borders and free spaces in which the development can move. We identified five areas that companies structure by using plan-based methods. These areas are: Requirements engineering, architecture, planning, testing, and coordination. Throughout the paper, we call the structures that are defined in each area *frames*. We further analyzed the dependencies between these frames and discovered that they build upon each other during the creation of an APH approach.

Outline: The remainder of the paper is structured as follows: In Section 2, we present related work and background information to our research, followed by Section 3, where we present our research design. In Section 4, we present the results of our interview study and explain each identified frame in detail and the dependencies among them. In Section 5, we discuss our findings and the threats to validity. In the end, we conclude in Section 6.

2 RELATED WORK AND BACKGROUND

The systematic investigation of hybrid approaches started with the *HELENA study* [2] that is based on a systematic literature review by Theocharis et al. [24]. They analyzed which development methods were combined in practice and discovered a wide use of hybrid approaches. The Helena study asked practitioners about their used methods and practices [14]. Noll and Beecham [19] analyzed the data and found out that 66 % of the cases use an APH approach. Klünder et al. [11] discovered that most often Scrum, Iterative Development, Kanban, Waterfall, and DevOps are combined. Tell et al. [23] could confirm the assumption by West et al. [25] that a lot of companies follow the "Water-Scrum-Fall" like process.

General investigations of APH approaches are mostly done on the level of methods and practices as in the papers mentioned before. Apart from that, several case studies exist that, however, only investigate a special example but do not provide general insights.

Bick et al. [2] investigated the coordination challenges in large-scale software development based on the planning misalignment in a hybrid development approach. They recommend to use regular inter-team meetings to improve dependency awareness. Further, they propose an iterative planning process on inter-team level that runs ahead of the development process and permits an iterative adjustment process on team level.

Cao et al. [16] describe the benefits of an up-front architecture phase combined with extreme programming. A stable up-front architecture creates a foundation where different services can be built upon. It also gives the developers a clearer understanding of the whole system and dependencies can be managed more easily.

[2]Hybrid development approaches in software system development: www.helenastudy.wordpress.com

Heeagar and Nielsen [8] describe the challenge due to the inflexibility of documents in APH approaches and the effort to change them. Their presented solution is to write lower-level documentation iteratively and fill the lower-level specification accordingly to the information that comes in.

Kautz and Madsen [10] investigated agile development in practice in a case study and discovered that plan-based approaches give a structure to the development process by long-term planning specifications, while agile practices allow flexibility.

These examples point to existing conflicts in APH approaches between agile and plan-based methods. In our previous work [21], we conducted a systematic mapping study and identified the following five conflicts:

Conflict between up-front and continuous requirements engineering: This conflict concerns the difficulty of the decisions about how much up-front requirements engineering is necessary and when the project should switch to a continuous requirements engineering.

Conflict between up-front and continuous architecture and design: This conflict describes the difficulty of the decision about how much up-front design of the software architecture is necessary and when the project switches to a continuous design of the architecture.

Conflict between central decision making and self-organized teams: This conflict concerns the decision about how far teams are self-organized and when central decision-making takes over.

Conflict between long- and short-term planning: This conflict emerges because companies have to decide how much up-front planning can be used and how it is used together with short-term planning.

Conflict between explicit documentation and tacit knowledge: This conflict concerns how much documentation is needed and which information can be kept in the heads of the project members.

Next to the conflicts in APH approaches, we conducted a systematic mapping study to analyze the general patterns for the organization of hybrid approaches [20]. We found three general methods to organize APH approaches. The first method is the Waterfall-Agile-Method (WAM). It uses the five waterfall model with its phases (requirements engineering, architecture, development, testing, and operations) as the main method with an agile part during the development phase. The second method is the Waterfall-Iterations-Method (WIM). It consists of iterations with the phases of the waterfall model inside these iterations. The last method is the Pipeline-Method (PM). There, the phases of the waterfall model are performed in parallel for different increments. For example in an iteration I, the increment N is developed. During the same iteration, the requirements are elicited and analyzed for increment N+1 and quality assurance is performed for increment N-1. In the next iteration I+1, all increments migrate to the next phase. We also found out that the WIM and PM are often combined with the WAM during development.

3 RESEARCH METHOD

In our literature research [20, 21], we discovered conflicts that exist in APH approaches and which methods are used by companies to organize APH approaches. However, we only discovered the existence of these conflicts without any information about how they can be solved and their connection to the organization of APH approaches. We aim at creating a theory about how the conflicts are connected to each other.

Our research is based on Grounded Theory which does not use a concrete research question but allows common concepts to emerge from the data that represent the most important topics of the participants [6, 7]. Further, it aims at discovering the relationships between them and forming an overall theory. Therefore, we decided to adopt Grounded Theory as our research method. In the following, we describe our Grounded Theory procedure in detail.

3.1 Study Design and Data Collection

For the data collection, we decided to conduct an interview study, where we can get insights into multiple cases. An interview study also allows discussing interesting points with the interviewees in more detail, compared to a survey study. We looked for participants that work or have worked in a project with a hybrid setting in a company. Since the conflicts affect different areas of the development process, we looked for different roles, like project managers, product owners, developers, and testers. To acquire participants, we stated an invocation among the industry contacts of our research institute and on LinkedIn. In total, we interviewed 15 practitioners in Germany that work with an APH approach. Table 1 shows an overview of the interviewees with their experience, role, the business context of their project, and project size. The participants come from different business areas, like finance, insurance, automotive, and the development of business processes. The size of the development teams ranges from 3 to 80.

For our study, we used semi-structured interviews. They provide a structure and direction for the interview and allow at the same time to explore interesting topics and do not hinder the interviewees to mention important points [9]. For conducting the interviews, we followed the guidelines by Runeson and Höst [22] and Hove and Benta [9]. The interviews were conducted between April and December 2021 and lasted about 60 minutes on average. The interviews were recorded and transcribed afterward by the first author. One interview was not recorded due to technical difficulties. However, in this case, we noted the relevant information during the interview.

For the interviews, we defined a rough course. At the start, we asked the interviewees about their background, experience, and the project they are involved in order to get a better understanding of the context. Further, we asked about the role they embody in those projects and which tasks and responsibilities they have. The next block of questions covered the process of the project. This includes the question about up-front activities, which methods, practices, and artifacts are used during development, and which activities are performed at the end of the project. During this part, we also inquired about the reasons or advantages behind the use of the described activities, practices, and artifacts. This enables us to get a better general understanding of the projects and take the project context into account when analyzing the results.

In the second part of the interview, we covered, as far as applicable, the handling of the described conflicts [21]. Here we asked how far plan-based and agile approaches are used for the activities and why. At the end of the interview, we asked for challenges and problems in APH approaches that were not already covered by our questions.

3.2 Data Analysis

The coding process of Grounded Theory consists of open, axial, selective, and theoretical coding and was performed by the first author. The results and codes were discussed with the second author afterward. To demonstrate our coding process in more detail, we show how we extracted one of our main categories *the coordination frame* from the transcripts. Figure 1 shows the example.

The first step in our coding process was to divide the transcripts into independent paragraphs that describe one distinguished topic. The topics of the paragraphs were summarized with *keywords*. Afterward, we assigned *codes* to the keywords which describe the keywords with two or three words. As all the interviews were held in German, we translated all presented phrases into English. The following example shows one of these paragraphs:

Raw Data: *"Self-organization usually ends where we have a fundamental question that not only affects our team, but also other teams. Mostly, it is fundamental technical matters, because the technical requirements mostly concern a special team. Fundamental questions usually go up (project lead), we wouldn't like to organize that ourselves."*
Keyword: *Self-organizations ends at fundamental technical topics that concerns more teams.*
Code: *Team border*

In this example, the interviewee describes that fundamental questions where decisions affect the whole project and not only the single development team are cleared with the project or architecture lead of the project. Therefore, we assigned the code *team border*. After the open coding process, we applied axial coding to the codes to form *concepts* by constant comparison of the codes. The concept we derived from similar codes like the one above is *borders of self-organization*.

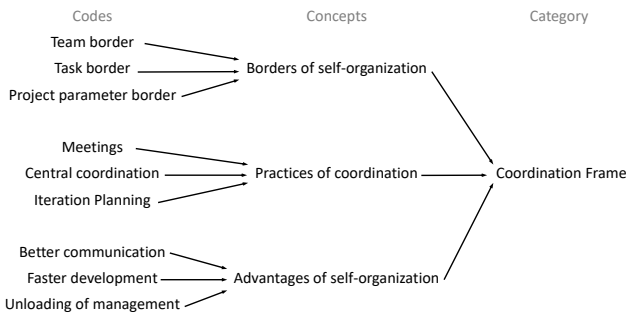

Figure 1: Example of codes, concepts and category

The emerging concepts were again constantly compared to form *categories* to reach the next level of abstraction. During the analysis

of the concepts, we found out that self-organization is defined by borders in order to define free spaces for the teams and not lose control over the teams.

After axial coding, we applied selective coding to identify the core category. We identified that the companies use the plan-based elements in their development approach to structure their agile development. With the plan-based methods, they delimit the field in which a project can move. Therefore, we named our core category: *Frames for the agile development*. The last step in our coding process is *theoretical coding* where the relationships between different concepts and categories were analyzed. This step was further supported by the use of *memoing*, where we documented important thoughts during the coding process. We see that self-organization is, for example, influenced by the architecture of the software. The more dependencies the development teams have to each other or to other projects the less self-organized are they. The data collection and analysis were conducted iteratively in order to react to emerging themes in the process. In this way, we identified an additional conflict in the testing area and incorporated it in our interview course.

4 FRAMES IN APH APPROACHES

During our described coding process, we discovered that the projects define the coarse boundaries for the project at the beginning. These boundaries take into account the content of the project (requirements), the planning parameters (timeline and budget), the architecture, the testing process, and the coordination of the teams. These boundaries were also called frames, which is why we decided to choose this term:

> *Interviewee 01: "I'm more a friend of saying: Define it roughly first, then I'll have a frame in which I can move."*

The frames are used to give the project structure and stability and define a free space in which the project can move and be flexible at the same time.

Finding 1: Companies use APH approaches to give structure and stability to their agile development and define boundaries in which the project can move. The identified frames are: The requirements engineering, the architecture, the planning, the testing, and the coordination frame.

In the following, we describe each of the frames in more detail and the relationships between the frames.

4.1 The Planning Frame

Companies and customers need a rough estimation of the required time, budget, and skills for the whole project to set the frame for the project and to plan other projects.

> *Interviewee 10: "[The developers] look at the customer's scope and what the complexity of the application is. And then they estimate how long it will take. With such a risk, this timeline can be upheld."*

Regarding the management of the projects' scope, we discovered differences resulting in four different *modes*. In the last column of Table 1, we show which modes we identified in each case. These different modes influence how the project is planned.

On the one side of the spectrum is the *Framed-Agile-Mode (FAM)*. In these cases, the requirements are not known and changes can always occur. Therefore, the companies only set the initial time, budget, and skill frame for the whole project and let the scope be flexible by adapting the priority of features. The difference between the FAM and a pure agile approach is that in a pure agile approach estimations for a whole project are not performed.

> *Interviewee 01: "That is I have to say that I will somehow deliver parts of the software by a certain date, but then, of course, I cannot say exactly what the content is."*

We named the second mode *Milestone-Agile-Mode (MAM)*. In these cases, we often observe requirements that have a deadline when they have to be implemented and deployed. They are often also referred to as *must haves*. These requirements mostly have a fixed scope with a fixed time. In order to be flexible enough, the companies plan the development of the other requirements in an agile way around the milestones of the project. In these cases, the company also flexibly assesses whether other features can be pre- or postponed.

> *Interviewee 11: "There are start of production dates, where things just have to be there. [...] If needed, you reduce the scope. There is a certain baseline that needs to be implemented. [...] And then, there is always a handful of requirements that are just so low in priority [...] that they are simply not implemented."*

The third mode is the *Plan-based-Agile-Mode (PAM)*. In this mode, the requirements are uncertain, but the customer or company needs planning certainty. The solution is to divide the development in periods for which the scope is fixed and planned. These periods have a duration between three and six months.

> *Interviewee 13: "It makes sure that what the teams will do over the next three months is very, very clear. [...] The other involved projects are not as agile as we are. Therefore, for us, it is a compromise at this interface [to the other projects]."*

The last mode we discovered is the *Execution-Agile-Mode (EAM)*. In these cases, the requirements are often known, but the solution is uncertain. Despite the unknown solution, the scope and time frame for the whole project is often defined beforehand. Nevertheless, the companies use agile methods but apply them rather as a management and control tool. For example, the structure of Scrum enables the project lead to always have an overview of the status of the project.

> *Interviewee 05: "Sprint planning, backlog, sprint review, daily scrum [...] are very important building blocks that you can use to build up a month [...] as a project manager [...]. Further, as a project manager, I know where we stand every day."*

Agile methods also encourage constant communication between different teams and departments. Since the solution is often unknown in these cases, this is beneficial, because it enables a fast reaction when problems arise. However, in these cases, the mere application of agile methods does not lead to agility, since change

Table 1: Demographics of the Interview Participants

ID	Experience (Years)	Role	Project	Project Size (# of people)	Development Approach	Mode
01	25	Project Manager / Software Architect	IT services for banks	5 - 20	WAM + PM	Framed Agile
02	> 30	Project Manager	IT service for banks	30 - 50	WAM + PM + WIM	Milestone Agile
03	> 25	Product Manager	IT services for banks	15-20	PM	Plan-based Agile
04	0.5	Software Architect	IT service for insurance	20 -30	WAM + PM	Milestone Agile
05	> 10	Tester	IT service for banks	20	WAM + PM	Agile Execution
06	10	Developer	Automotive supplier	4	WAM	Agile Execution
07	25	Product Owner / Software Architect	Supplier for satalites	10	PM	Milestone Agile
08	> 4	Developer	Accessmanagement System	3	WAM + PM	Milestone Agile
09	35	Agile Coach	IT services for insurance	30	WAM + PM	Framed Agile
10	> 5	Resource Manager	IT services for business processes	50	WAM	Agile Execution
11	8	Agile Coach	IT services for business processes	5 -10	WAM + PM	Milestone Agile
12	13	Developer / Software Architect	IT services for a railroad company	90	WAM + PM + WIA	Milestone Agile
			IT services for insurance	5	WAM	Agile Execution
13	7	Agile Coach	IT serives for logistic	7 - 11	WAM + PM	Framed Agile / Plan-based Agile
14	> 10	Agile Coach	IT services for insurance	10 - 30	WAM + PM	Framed Agile
15	> 30	Product Owner	Automotive supplier	60 - 80	PM	Milestone Agile

requests have often to be used to change the scope.

Finding 2: We identified four different modes of how the scope is managed in APH approaches: The Framed-Agile, the Milestone-Agile, the Plan-Based-Agile, and the Agile-Execution-Mode. The scope can be either rather fixed or be adapted by reevaluating the priority of requirements.

If several teams are involved in the project, a roadmap helps to show how the different parts of the projects are connected and where dependencies to other projects exist. It gives the project more structure and stability and enables project managers and developers to keep an overview.

Interviewee 11: "Like pieces of a puzzle. Sometimes something depends on something else. [...] These roadmaps are super helpful to keep track of things, especially in large projects."

However, the roadmap must remain on the level of a vision. A too narrow roadmap demands too much rework if changes occur. A good roadmap needs to remain maintainable. The roadmap has to be constantly adjusted to changes. The most important planning task is the constant review of the status and planning the development order of new features.

Interviewee 11: "A roadmap changes and it changes extremely fast. [...] The more fine-grained you write it down, the more you document yourself to death because it is a living document. It needs constant care and maintenance."

4.2 The Requirements Engineering Frame

The next identified frame is defined during the up-front requirements engineering phase. The phase sets a frame of the content of the project by creating a common understanding of the project and its vision. This is important to give the project a goal and to set initial boundaries in which the developers and stakeholders can move. The creation of a common understanding fosters the transparency of the project and ensures that the project does not go in the wrong way right from the beginning.

Interviewee 06: "Because in the past, and that was perhaps also the problem, [...] we never gave any thought to the requirements beforehand and didn't even know where to go. And [up-front RE] definitely helps to think a little more about the requirements in this phase and get clarity about it."

This way, stakeholders can also be sure that their requirements are considered. Especially, if there are many stakeholders, it is beneficial to collect all their requirements at the beginning. Later, during the project, it is sometimes difficult to reach all stakeholders. Compliance requirements or other important non-functional requirements, like security or performance, have to be considered directly at the beginning of the project and, hence, need to be considered beforehand.

Interviewer: "Is there a benefit to extend the up-front requirements engineering phase?
Interviewee 12: "Yes, if you have a lot to do with compliance and have to fulfill a lot of legal requirements. Or lots of framework conditions that are impossible to avoid."

However, it happens that customers want to discuss details that are often uncertain and unknown. This introduces the risk of getting lost in details, losing overview, and discussing detailed requirements under vagueness. This is especially important because it creates an apparent certainty that is not there in reality.

Interviewee 13: "We're talking about a 30,000€ block right now. If you [customer] had listened, you would have heard it's a million plus or minus 30%. That is we are talking about information in the absolute blur range. Then [up-front requirements engineering] has no value."

The ideal point to end the up-front requirements engineering phase is when the project participants reach a rough and common understanding of the application and the vision for the project.

Interviewee 01: "My goal at the beginning of a project is to understand these requirements at least on a detailed level so that (a) I understand them and (b) that I am able to give at least a rough indication of how complex the implementation would be."

This point is refined by the ability of the project participants to give a rough estimation for the project. This includes statements about the estimated cost and length of the project and which employees are needed for the project. These estimations can be provided in conjunction with a risk assessment about how certain these estimations are. When everyone agrees on these rough estimations and has the gut feeling to have sufficient information, to begin with the development, the development can start. If the requirements are rather uncertain, it is also possible to only concentrate on one successful use case and the functionality that is needed for this use case.

Finding 3: The up-front RE phase strives to give a first understanding of the project such that the team members can provide a coarse estimation of the duration, the cost, and the needed skills.

We analyzed the tasks that are used by companies to reach this point during the up-front requirements engineering phase. The most important task is an analysis of what the customer needs.

Interviewee 08: "We did [requirements analysis] by using prototypes and wrote down beforehand which use cases exist."

During the analysis, different use cases are gathered and analyzed. Based on these insights, the needed features and their functionality can be derived. Epics are reported to be a suitable practice to retain the overview (without getting lost in detailed story cards). Since the scope of the project is often variable, the requirements have to be analyzed regarding their priority. The requirements that are most valuable for the customer have the highest priority. Another suitable practice is the creation of a roadmap, helping to present the vision and the direction for the project. It is important to keep the roadmap on the level of a vision to manage changes and let enough flexibility. A roadmap helps to convey the vision among the project team and fosters transparency of the project.

Finding 4: The requirements engineering frame consists of the coarse use cases of the software that form a vision for the project.

In the column "Development Approach" of Table 1 we present an overview of which of the three described methods to organize APH approaches were used by the investigated cases to organize their whole approach (WAM, WIA, or PM). In 12 out of 15 cases, the Waterfall-Agile-Method (WAM) was used. The data shows that almost all cases use the WAM to set a requirements engineering frame for the project. The three cases that use only the Pipeline-Method (PM) are continuous projects that partly run for years and therefore do not need an initial phase.

The initial requirements engineering phase at the beginning of the project only sets a frame for the requirements. During the development phase, iterative requirements engineering is used. Our results show that all cases use the PM for their requirements engineering during the development phase. The tasks during the iterative requirements engineering are:

1. Intake of new requirements: New requirements have to be gathered and evaluated regarding their business value.

2. Estimation: To create a better understanding of these requirements, the developers can take a look at them and evaluate their feasibility. Based on this, the developers give a rough estimation of the effort to implement these requirements.

3. Backlog Grooming: At the beginning of the project, the requirements are gathered on the level of epics. During the development, these epics have to be filled with more detailed user stories.

4. Refinement: The user stories that are considered ready for development by the product owner are presented to the developers and discussed in regular refinement meetings.

5. Planning of Development: If also the developers give their approval of the user stories and consider them as ready for development, the user stories can be planned.

4.3 The Architecture Frame

An up-front architecture provides a technical frame for the development in which the teams can move and provides an overview of the application. Also, up-front architecture has benefits or even has to be conducted. Important non-functional requirements like performance or security have to be considered at the beginning because it is often difficult to integrate them later.

Interviewee 01: "Capacity, availability, performance is something that I have to take into account very early on in the design, otherwise I'll hit a wall there. Also, if I have very strict security requirements, then I have to incorporate them into the design right from the start."

The most often mentioned point is to define the interfaces at the beginning of the project. This enables the development teams to move freely in their designated area.

Interviewee 09: "If the interfaces are clear, then it doesn't really matter what the implementation looks like behind them."

Also, the technological framework conditions, like coding language or the infrastructure of the application, have to be set. However, there is the risk of defining too much architecture up-front, because it reduces the project's degrees of freedom and often leads to rework.

Interviewee 12: "The more fine-grained you make the architecture, the more frequently you have to revise it afterward [...]. I'm more in favor of letting the architecture evolve over time. [...] Under the condition that certain framework conditions are met, such as security, for example."

A too detailed architecture leads to a loss of oversight. Sometimes, architecture has to be globally created for the whole project because all parts need to exactly fit together. However, an architecture that is too narrow for the whole project slows down the development teams since they often wait for decisions. Also, the teams need some degree of freedom in which they can move.

Interviewee 11: "The deeper and more specific you make the architecture at this point, the more the conversations go in a direction that is not productive. You actually want to avoid this at this point, because you tend to pay attention to this overall crosscut and then let the individual teams run their own course."

Interviewee 09: "If people are pinned on [...] an architecture, then they have to wait for others at certain points. The architecture imposes a constraint on developers, making them slower."

Finding 5: The up-front architecture focusses on the definition of interfaces considering important non-functional requirements.

In all cases where the WAM is used, it is also used for architecture. Only three cases integrated the architecture activity directly into the PM. In most cases, the architecture was developed during the implementation. However, often, there is a role or a team that maintains the architecture frame.

Finding 6: The architecture phase in the PM is often omitted, but the architecture is governed by a team or person.

Architecture is closely bound to the documentation of a project. Documentation is important and the trend we observed in our interview study is to document rather too much than too little. However, documentation also has to be written and maintained. This makes documentation an exhausting task. Our results show that companies only document things that are necessary for the understanding of the software and concern the architecture. Here it is important to document the structure of the software that represents an overall overview.

Interviewee 01: "Know how my system is set up. How is it structured. What interfaces do I have. What components do I have. What functionalities do I have. What do these components represent."

If documentation is merely another depiction of something that already exists, it is not used. Further, models that can be easily derived from the code source are often not documented. However, code documentation is still considered important. To evaluate documentation, senior developers are asked to determine if the documentation suffices enough to understand the software. Knowledge that can be obtained from books or other sources is not documented.

Interviewee 12: "The documentation of professional knowledge is unnecessary. Further, if I have a model that is derived from a source, I have to ask myself: Do I need this model?"

Finding 7: Information that is necessary to understand the software is documented, while textbook knowledge and information that are easily to obtain are not documented.

4.4 The Testing Frame

We identified an additional conflict in the testing area. The conflict exists because plan-based approaches propagate to separate testing from the developers to ensure a structured testing process, while agile approaches propagate testing by the developers to ensure fast feedback. To assure that all tests are conducted, a testing frame with which the teams work has to be defined. This defines which tests are conducted by the developers and if an additional test team is necessary. Our results show that only one case uses the WAM for testing. That means that all companies strive to deploy regularly and test iteratively during development. According to our results, test automation is essential for fast development and supports agility because it generates fast feedback.

Interviewee 11: "What is always very good is to automate the system tests."

We observe that the companies strive to shorten the period between implementation and tests. Therefore, all unit tests are conducted by the developers. When it comes to integration and system tests, there are two variants of how the tests are conducted. The first option is that these tests are conducted by an extra team. The second option is to let these tests also in the hands of the developers. The advantage of the first option is to have a clear test responsibility and a team that steers the whole testing process. Further, this way the developers can focus on the development work.

Interviewee 12: "One advantage is that you don't know when you roll it out, whether it will still work with the others. If you haven't tested properly or communicated with the others, then of course a test team that channels everything a bit is worth its weight in gold."

In eight cases, the companies used the PM with an extra test team to conduct the tests. In these cases, a synchronized release rhythm to the test team among all development teams has to be defined to coordinate the test process.

There are also benefits of having the tests performed by the developers. An extra test team slows down the development because teams cannot deploy directly. Also, found errors have to be first reported back to the developers before they can fix them which also delays the development. The test responsibility in the hand of the developers facilitates having an executable version of the software at the end of each sprint and increases the quality awareness of the team. However, this also takes time from the sprint.

Interviewee 12: "The closer testing and deployment are located to the development team, the faster [the developers] get feedback and can fix things."

Finding 8: To set the testing frame, the companies decide which tests can be conducted by the developers and which by an extra test team (if even necessary) and set a release rhythm for the development teams.

4.5 The Coordination Frame

The last frame we identified considers the coordination activity. The participants see self-organized as beneficial because it leads to more communication between the teams, a faster development, and the project leader is unburdened. However, self-organization also has its boundaries because not all decisions can be made by the team itself. To have a structured development, plan-based approaches propagate more central decision-making. To give the teams orientation and security regarding their degree of freedom and to provide structure, a frame for coordination has to be set. In large-scale settings, the teams have to comply with the general framework and rhythm of the project and are limited in their self-organization.

Interviewee 12: "The self-organization ends at the sprint and iteration grid. Of course, we are bound by these dates."

If questions arise that do not only concern the single team but also the whole company or project, the project lead or manager is brought into the decision. This is, self-organization is limited to questions that only affect the single development team.

Interviewee 01: "I can only do self-organization within my team as long as it only affects the nucleus of this team. As a team, I cannot make a decision for my company, for example."

Further boundaries are set if the project parameters, like the time or budget frame, are affected by the decision in the team or the team has to fundamentally deviate from the task.

Interviewee 01: "Of course, if the teams say, we need more money now, they cannot just decide that themselves."

Finding 9: Self-organization ends where decisions have to be made that affects the whole project, the parameter of the project or the team deviates fundamentally from their designated tasks. In these cases, central decision-making is used.

4.6 Dependencies between the Frames

Grounded Theory not only consists of the categories that emerge from the data but also of the dependencies between these categories (theoretical coding and memoing). In Figure 2, we show the five frames and their influence on each other. In the following, we describe each of the dependencies in detail.

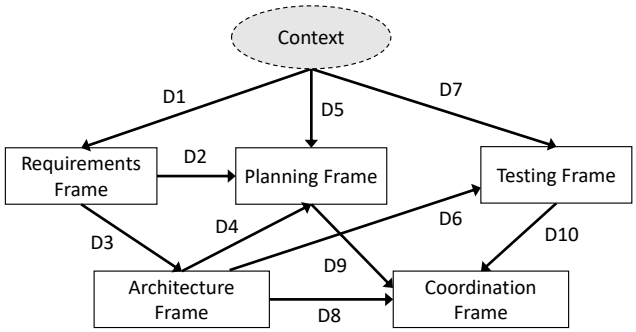

Figure 2: Dependencies among the categories

The context, in which a project is located, influences how the frames are defined. In this study, the context describes the kind of application that is developed, the environment of the project (technical infrastructure, dependencies to other projects etc.), and the customers and stakeholders.

Dependency 1: *In cases with a lot of stakeholders, a high complexity or high criticality of the application more effort to create the understanding of the project is needed. The certainty of the requirements influences when the requirements frame becomes too narrow.*

The requirements engineering frame aims to set an understanding and vision for the project. The more complex a project is, the more effort is needed to get an understanding. If a lot of stakeholders are involved, it is even more difficult. Also, the criticality determines how long it takes because all relevant requirements have to be gathered.

Interviewee 11: "And then there are projects that intervene intensively in the vertical structure of an organization [...]. There are a lot of departments that use the application. You have to collect the requirements first properly."

On the other hand, the certainty of the requirements, especially in the details, determines when the framing for the project becomes too narrow and requirements are discussed under uncertainty.

Dependency 2: *The degree of understanding and the certainty of the requirements influence how precisely the project parameters can be defined.*

The requirements frame defines how well the project team can estimate the project. If the requirements and/or the solution are uncertain, there can always be changes or delays in the development. Therefore, the parameters can only be roughly estimated and have to be checked regularly. The more both aspects are certain, the better is the estimation.

Dependency 3: *Non-functional requirements, dependencies, and the requirements certainty influence how narrow the architecture frame for the software has to and can be.*

In our results, we see that especially critical non-function requirements like security performance have to be considered right from the beginning. They decide how detailed the architecture needs to be. If the software does not stand alone but has many dependencies to other projects, an analysis of the interfaces and how the architecture fits into the existing applications has to be performed. Further, in large-scale development, it is important to have a closer look at the architecture and define the interfaces and the structure in order to enable the teams to work independently and give them boundaries in which they can work.

Interviewee 11: "I nail down the requirements and the communication of the systems based on the interface. If I describe it first and put some effort into it, then I really don't care what the systems do on the left and right. That's what they need and then they can get along with each other."

However, the architecture must not be too narrow, because changes can occur especially if the requirements are uncertain. An analysis of the degrees of freedom has to be performed in order to be as flexible as possible within the defined frame.

Dependency 4: *The analysis of dependencies and interfaces influences if a planning roadmap for the project is needed.*

The more the single components of an application or the application itself are independent, the fewer long-term planning and milestones are needed. If the project teams do not need to consider other projects, a roadmap is rather obstructive, since it needs rework in cases of changes. If different development teams depend on each other, a roadmap is useful to have an overview of how all parts of the project are connected. However, the roadmap should be on the level of a vision in order to keep it maintainable in cases of changes.

Interviewer: "Do you use a roadmap?"
Interviewee 01: "No, this is only used if you have dependencies to fixed releases."

Dependency 5: *The customer or company decides how the scope has to be handled, which influences how the project is planned.*

Our results show that scope cannot always be handled as flexibly as it should. In a dialog with the customer, the conditions regarding the scope have to be determined. Since APH approaches also operate in a complex area, where the requirements and the solution are uncertain, the scope should be as flexible as possible in order to be

able to react to changes.

Interviewee 02: "I do something, I show what I've got there and then I'll look at it. If it comes out that I want to have it differently now, then I can react and act much more flexibly."

Dependency 6: *The resilience of the software against single failures influences whether the testing is done by the developers or by an extra test team.*

In case of an architecture that does not lead to a system crash if single features or services fail, the tests can be rather in the hands of the developers. In these cases, the developers get faster feedback for their work. However, if all services depend on each other and need careful integration, the system is less resilient against single failures. In these cases, the integration and system tests should be laid outside of the single development teams.

Dependency 7: *The testing frame is influenced by how new software can be deployed and if customers demand a high level of test assurance.*

Software companies are often bound to the deployment cycles of their customers or the software needs to be integrated into a larger system. In these cases, software cannot be deployed regularly.

Interviewee 03: "You give the release into production once a year. The entire time in between you have frozen time for production and you are only allowed to access the systems if very urgent bug fixes are necessary."

However, in other cases weekly or even daily deployment is possible. Based on these factors, the rhythm of the test cycles has to be set. If further a high test assurance, traceability, and documentation are needed, an extra test team for integration and system tests is beneficial.

Dependency 8: *The architecture frame influences the degree of self-organization of the teams.*

If there are clear interfaces between the development teams, they can be rather self-organized in their team bubble. If there exist a lot of dependencies, more planning and oversight between the team is necessary. To support this, the architecture can be governed more globally for the whole project. This can be performed by one person or a team. An integration team can support the collaboration of different development teams. However, the architecture frame must not be too narrow since it takes freedom from the development teams and slows them down.

Dependency 9: *The planning frame influences how centralized the planning has to be and how much room for inter-team planning is needed.*

If there are a lot of dependencies between teams or to other projects, the whole planning has to be governed more centralized via a roadmap. In order to avoid losing the overview, a person or team can support the planning and coordination of different teams. Development teams with several dependencies need more room for coordination, for example, by additional regular meetings, like Scrum of Scrum. Coordination can also be supported by integrating the Waterfall-Iteration-Method (WIA) into the PM. At the beginning of an iteration, the teams can use a phase to clear dependencies.

Interviewee 03: "And we also talked about queries, agreements, additions, changes, and so on. And this time is what I call set-up time. And you should actually allow yourself this set-up time before the developer phase. But we do not do that. Instead, this set-up time of coordination is shifted to the development phase."

Dependency 10: *The testing frame influences the degree of self-organization of the development teams.*

Besides the architecture and planning, the testing frame influences how self-organized the development teams are. If the single development team can conduct all the integration testing on their own, they are more self-organized and have a shorter feedback span. If the development teams have to follow the directives from the test team, they are less self-organized, and the whole process lasts longer.

Interviewee 13: "If you deploy code in three months and an error occurs, you won't know anymore what you did there."

5 DISCUSSION

Companies similarly need planning certainty and flexibility. *Therefore, based on our results, we state that one of the main reasons to create APH approaches is to stabilize and structure agile development to a certain point to reach a degree of security for the customer, company, and developers.* We discovered that companies do this by framing their agile development. These frames give structure but also leave room for flexibility. The described conflicts [21] exists because for the definition of these frames the opposing principles of agile and plan-based development have to be weighed against each other. To not define the frames too narrow, it has to be analyzed which principles of agile development should be kept and which can be neglected. For example, our results show that the flexibility of the scope has a major influence on the agility of a project. The different *modes* we discovered form a spectrum and the flexibility of the scope is one key element of the difference between merely doing agile and being agile.

Our results show that during the framing process the single frames build upon each other and define stepwise the development approach. This supports the perception of Boehm and Turner [4] who describe that APH approaches are built up rather than tailored down. Only after the framing process is done, the development approach can be actually filled with methods and practices. In the literature, APH approaches are mostly discussed on the level of methods and practices [11, 23]. Existing approaches to create and tailor hybrid development approaches depend solely on which methods and practices from agile and plan-based approaches should be chosen [4, 15, 23]. However, the results of our previous work [21] and this study show that the mere discussion of methods and practices is by far not sufficient to build APH approaches. This argumentation is supported by the finding of Kuhrmann et al. [13]. They discovered that the used methods and practices have little influence on the perceived degree of agility. *Based on that, we state that the framing process is an important missing link in the creation of APH approaches.* After the companies define the frames, they can build their development process. We see that the WAM together with the PM is a good framework for APH approaches. In the following, we list different recommendations to guide practitioners while defining

the process of an APH approach based on our insights from this study and previous work:

Recommendation 1: We recommend to only use the requirements and architecture phase of the WAM to frame the project and to omit the testing and operations phase of the WAM. The project team should strive to create a runnable tested version at least after each iteration. This enables the team to get feedback and always be able to deploy.

Recommendation 2: We recommend that the scope of the project is flexible since also APH approaches operate in complex areas (Framed-Agile-Mode). Even, if there are concrete milestones that have to be fulfilled, the scope has to be partly flexible to have buffer. We also recommend using rather short planning cycles to generate fast feedback. However, if the scope is not flexible, we recommend using the *Plan-Based-Agile-Mode* instead of the *Agile-Execution-Mode* because it uses more of the advantages of agile development. In these cases, the period for which scope has to be fix has to be determined. It is important to choose shorter periods to prevent uncertainties. The creation of solution concepts for new requirements can facilitate the estimation and planning of these requirements. If the *Agile-Execution-Mode* has to be used after all, the project has to be carefully tracked in order to see issues early.

Recommendation 3: We recommend to create a concrete framework for the iterative requirements engineering to ensure a continuous flow. We further recommend understanding the *information flow* of requirements as a waterfall. To ensure a high quality of requirements and structure of the process the steps of the waterfall have to be carefully defined. We recommend using a hierarchy among the requirements documentation to support the hybrid nature of the project. There should be a common understanding of how requirements should look like to properly implement them.

Recommendation 4: We recommend keeping the testing as near to the developers as possible in order to create fast feedback cycles and facilitate the fixing of errors. If software can not be regularly deployed during the development cycles, we, nevertheless, recommend synchronizing as much of the testing with the planning cycle, especially integration testing to create regularly a runnable version of the software and get feedback.

For the future, the research community has to consider that the sole combination of agile and plan-based methods and practices might not solve the problem. It might be necessary to define new methods and practices that combine the principles of agile and plan-based development or even define combined principles for APH approaches. Here, the research community can support the industry. Companies are often narrowed on their day-to-day business and chose rather fast and easy solutions that serve as temporary solutions. They often do not have the time to consider the big picture and the theory behind methods and practices.

5.1 Threats to Validity

Our work is subject to some threats to validity. We discuss according to Maxwell [18], descriptive, theoretical, and interpretive validity and generalizability.

Descriptive Validity. Descriptive validity considers the correctness of the made observations. To mitigate this threat we recorded the interviews and transcribed them afterward for the analysis.

Theoretical Validity. Theoretical validity considers the extent to which we were able to collect the relevant aspects. We choose a multi-case interview study to get a general insight into APH approaches that is not limited to one case. We used semi-structured interviews to be able to explore interesting points that come up during the interview. The interview questions were directly derived from our previous work and discussed with other researchers in the field of agile and hybrid approaches. This may created a bias because we were focused on these aspects. In order to mitigate this threat, we added open questions at the end of the interviews to collect information that the interviewees view as important, but we did not cover with our questions. Further, we conducted the data collection and analysis iteratively to react to new themes. We followed the guidelines by Runeson and Höst [22] and Hove and Benta [9] to get genuine answers. Most important, we assured the anonymity of our participants.

Interpretive Validity. The interpretative validity considers the conclusions drawn from the data. To minimize this threat we followed the coding procedure of Grounded Theory. The whole coding process was conducted by the first author which increases the mono researcher bias. To prevent this effect, the process was reviewed by the second author and the results were discussed. We observe a large agreement among the participants and different points were mentioned multiple times by different interviewees. We identified no contradictory statements. Therefore, we consider our results and findings as valid.

Generalizability. Our interview study consists of only 15 participants and all came from Germany. This impairs the generalizability of our results. However, we strove to include different business contexts, roles, and project sizes in our study to see the application of APH approaches under different contexts and perspectives.

6 CONCLUSION

We conducted a multi-case interview study to investigate how companies handle the conflicts in APH approaches. We discovered that the conflicts exist because companies want to set frames for their agile development. We identified five different frames that are defined by companies: the requirements engineering, the architecture, the planning, the testing, and the coordination frame. Especially the requirements, architecture, and planning frame serve to give a structure in which the development teams can move.

The frames provide an abstract view on APH approaches that is necessary to understand them better. Due to this abstract view, we do not cover all necessary considerations for the creation of APH approaches. Instead, this work shall increase the mindfulness of practitioners and researchers for the existence of these frames and their important role in the creation of APH approaches. Further, it shall guide the way towards a systematic creation of APH approaches. For that, the concrete relationship between the frames and the actual development process have to be further investigated.

ACKNOWLEDGMENTS

We thank all interviewees for participating in our interview study.

REFERENCES

[1] D. Batra, W. Xia, D. Meer, and K. Dutta. 2010. Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry. *Communications of the Association for Information Systems* 27 (01 2010), 379–394.

[2] S. Bick, K. Spohrer, R. Hoda, A. Scheerer, and A. Heinzl. 2018. Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. *IEEE Transactions on Software Engineering* 44, 10 (2018), 932–950.

[3] Jean Binder, Leon IV Aillaud, and Lionel Schilli. 2014. The Project Management Cocktail Model: An Approach for Balancing Agile and ISO 21500. *Procedia - Social and Behavioral Sciences* 119 (2014), 182 – 191. Selected papers from the 27th IPMA (International Project Management Association), World Congress, Dubrovnik, Croatia, 2013.

[4] Barry W Boehm, Barry Boehm, and Richard Turner. 2004. *Balancing agility and discipline: A guide for the perplexed.* Addison-Wesley Professional.

[5] Torgeir Dingsøyr, Nils Brede Moe, Tor Erlend Fægri, and Eva Amdahl Seim. 2018. Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation. *Empirical Softw. Engg.* 23, 1 (Feb. 2018), 490–520.

[6] B.G. Glaser. 1992. *Emergence Vs Forcing: Basics of Grounded Theory Analysis.* Sociology Press.

[7] Barney G. Glaser and Anselm L. Strauss. 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Aldine de Gruyter, New York, NY.

[8] Lise Tordrup Heeager and Peter Axel Nielsen. 2020. Meshing agile and plan-driven development in safety-critical software: a case study. *Empirical Software Engineering* 25, 2 (2020), 1035–1062.

[9] S. E. Hove and B. Anda. 2005. Experiences from conducting semi-structured interviews in empirical software engineering research. In *11th IEEE International Software Metrics Symposium (METRICS'05).* 10 pp.–23.

[10] Karlheinz Kautz and Sabine Madsen. 2010. Understanding Agile Software Development in Practice. *International Conference on Information Resources Management* (2010).

[11] Jil Klünder, Regina Hebig, Paolo Tell, Marco Kuhrmann, Joyce Nakatumba-Nabende, Rogardt Heldal, Stephan Krusche, Masud Fazal-Baqaie, Michael Felderer, Marcela Fabiana Genero Bocco, Steffen Küpper, Sherlock A. Licorish, Gustavo Lopez, Fergal McCaffery, Özden Özcan Top, Christian R. Prause, Rafael Prikladnicki, Eray Tüzün, Dietmar Pfahl, Kurt Schneider, and Stephen G. MacDonell. 2019. Catching Up with Method and Process Practice: An Industry-informed Baseline for Researchers. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice* (Montreal, Quebec, Canada) *(ICSE-SEIP '19).* IEEE Press, Piscataway, NJ, USA, 255–264.

[12] Marco Kuhrmann, Philipp Diebold, Jürgen Münch, Paolo Tell, Vahid Garousi, Michael Felderer, Kitija Trektere, Fergal McCaffery, Oliver Linssen, Eckhart Hanser, and Christian R. Prause. 2017. Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond. In *Proceedings of the 2017 International Conference on Software and System Process* (Paris, France) *(ICSSP 2017).* Association for Computing Machinery, New York, NY, USA, 30–39.

[13] Marco Kuhrmann, Paolo Tell, Regina Hebig, Jil Ann-Christin Klunder, Jurgen Munch, Oliver Linssen, Dietmar Pfahl, Michael Felderer, Christian Prause, Steve Macdonell, Joyce Nakatumba-Nabende, David Raffo, Sarah Beecham, Eray Tuzun, Gustavo Lopez, Nicolas Paez, Diego Fontdevila, Sherlock Licorish, Steffen Kupper, Guenther Ruhe, Eric Knauss, Ozden Ozcan-Top, Paul Clarke, Fergal Hugh Mc Caffery, Marcela Genero, Aurora Vizcaino, Mario Piattini, Marcos Kalinowski, Tayana Conte, Rafael Prikladnicki, Stephan Krusche, Ahmet Coskuncay, Ezequiel Scott, Fabio Calefato, Svetlana Pimonova, Rolf-Helge Pfeiffer, Ulrik Pagh Schultz, Rogardt Heldal, Masud Fazal-Baqaie, Craig Anslow, Maleknaz Nayebi, Kurt Schneider, Stefan Sauer, Dietmar Winkler, Stefan Biffl, Cecilia Bastarrica, and Ita Richardson. 2021. What Makes Agile Software Development Agile. *IEEE Transactions on Software Engineering* (2021), 1–1. https://doi.org/10.1109/TSE.2021.3099532

[14] Marco Kuhrmann, Paolo Tell, Jil Klünder, Regina Hebig, Sherlock Licorish, and Stephen MacDonell. 2018. HELENA Stage 2 Results. https://doi.org/10.13140/RG.2.2.14807.52649

[15] Steffen Küpper, Andreas Rausch, and Urs Andelfinger. 2018. Towards the systematic development of hybrid software development processes. In *Proceedings of the 2018 International Conference on Software and System Process.* ACM, 157–161.

[16] Lan Cao, K. Mohan, Peng Xu, and B. Ramesh. 2004. How extreme does extreme programming have to be? Adapting XP practices to large-scale projects. *37th Annual Hawaii International Conference on System Sciences* (2004), 10 pp.–.

[17] Isabel Laux and Johann Kranz. 2019. Coexisting Plan-driven and Agile Methods: How Tensions Emerge and Are Resolved. *International Conference on Information Systems (ICIS)* (12 2019).

[18] Joseph Maxwell. 1992. Understanding and Validity in Qualitative Research. *Harvard Educational Review* 62 (01 1992), 279–300.

[19] John Noll and Sarah Beecham. 2019. How Agile Is Hybrid Agile? An Analysis of the HELENA Data, Xavier Franch, Tomi Männistö, and Silverio Martínez-Fernández (Eds.). *Product-Focused Software Process Improvement*, 341–349.

[20] Nils Prenner, Carolin Unger-Windeler, and Kurt Schneider. 2020. How Are Hybrid Development Approaches Organized? A Systematic Literature Review. In *Proceedings of the International Conference on Software and System Processes* (Seoul, Republic of Korea) *(ICSSP '20).* Association for Computing Machinery, New York, NY, USA, 145–154.

[21] Nils Prenner, Carolin Unger-Windeler, and Kurt Schneider. 2021. Goals and challenges in hybrid software development approaches. *Journal of Software: Evolution and Process* 33, 11 (2021), e2382.

[22] P. Runeson and Martin Höst. 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14 (2008), 131–164.

[23] Paolo Tell, Jil Klünder, Steffen Küpper, David Raffo, Stephen MacDonell, Jürgen Münch, Dietmar Pfahl, Oliver Linssen, and Marco Kuhrmann. 2021. Towards the statistical construction of hybrid development methods. *Journal of Software: Evolution and Process* 33, 1 (2021), e2315.

[24] Georgios Theocharis, Marco Kuhrmann, Jürgen Münch, and Philipp Diebold. 2015. Is water-scrum-fall reality? on the use of agile and traditional development practices. In *International Conference on Product-Focused Software Process Improvement (PROFES'15).* Springer, 149–166.

[25] Dave West, Mike Gilpin, Tom Grant, and Alissa Anderson. 2011. Water-scrum-fall is the reality of agile for most organizations today. *Forrester Research* 26 (2011), 1–17.