Working paper

Michael Neumann

Cultural Impact on Agile Methodologies:
A causal impact model
1. Introduction

The use of agile methodologies in software development has grown steadily over the past two decades. Agile models such as Scrum or XP are used today both in co-located teams and in distributed software development. On the one hand, this spread is accompanied by the fact that more and more companies, regardless of their size or industry, are using agile approaches. Also, it can be observed that agile approaches have also established themselves in global software development (cf. [BMB15], [Hos08], [JW12]).

For the successful use of agile methodologies, aspects such as values and principles (cf. [BJ19], [MKK09]) as well as different cultures, including Organizational cultures (c. f. [AGA16], [BJ19], [BZT+14], [KSK+19]), are of importance. Several studies have been published in the context of the influences of different national cultures on global software development (cf. [HT07], [LKT17], [SSA+11], [Su15]).

Ayed et al. in [AVH17] have so far only identified the influence of different cultural levels on agile software development. With the help of their qualitative study, the authors were able to demonstrate the influences of national culture in [AVH17]. Javdani and Ziaei, for example, presented further findings on the influences of organizational culture in their study in [JZ16]. There are also both for national culture (cf. [GBS11], [HG11], [Hwa12], [MD07]) and for organizational culture (cf. [CSR+19], [Gal09], [PFG13], [PT18], [QH08]) various publications in the literature.

This paper aims to present a formal model that depicts the cultural influences on agile elements. This model should enable a better understanding of the adaptations of agile elements due to the application in different (cultural) contexts.

In the following section 2, I present the theoretical background for the model development. Here, agile approach models and their elements are first discussed. Furthermore, the fundamentals of culture are presented, and causal and impact are introduced. Subsequently, I give the development of the MoCloAE in section 3. For this purpose, the creation of the list of agile elements is presented in Section 3.1.

Furthermore, the second dimension of the cultural models is described in Section 3.2. Building on these two dimensions, the MoCloAE is presented in Chapter 3.3 in the form of a metamodel and the right specification. Finally, the limitations of this study will be discussed before a summary is presented.
2. Background

2.1. Agile methodologies and their elements

To be able to map the cultural influence on agile methodologies, it is necessary to consider the granular components of these process models as possible. Agile methodologies are usually defined in the form of guidelines. For Scrum, this is, for example, the Scrum Guide [SS17]. In addition to a procedural approach, practices and roles are also described in these frameworks. Since various agile approaches exist, there is a multitude of agile practices. In the guidelines of Scrum [SS17] and XP [Bec00] alone, 12 different practices and artifacts are described. There may be a fundamentally similar understanding of what is meant by an agile practice in literature and practice, but there is no uniform definition. The understanding in the literature differ from one another: According to Diebold and Zehler in [DZ16], agile practices are: “...established instructions, e.g., tasks, activities, technical aspects, or guidelines, with a specific focus or aspect in the development of software which is performed according to single or less agile core values and Agile Principles”. Abrahamsson et al. define in [ASR+02, p. 20]: “Practices are concrete activities and work products that a method defines to be used in the process.” Diebold and Dahlem describe agile practices as methods that are used in agile approaches [DD14]. Sidky et al. [SAB07, p. 206] define agile practices as “... concrete activities and practical techniques used to develop and manage software projects in a manner consistent with the agile principles”. Williams describes agile practices in [Wil10] as follows: “Conversely, practices are the applications of principles to a particular situation. Practices change as one moves from one environment and situation to another.”

Since it can be assumed that a cultural influence does not only have an impact on practices but also roles, it makes sense to consider these two aspects. For this reason, the term Agile Element is introduced below:

**Definition 1 (Agile Element)** Agile elements are practices or roles that are used in agile methodologies. They are based on the values and principles of the agile manifesto [Wil10].

Due to the complexity resulting from a large number of different definitions of agile elements in the literature, an investigation at this level does not make sense. Instead, it is necessary to extract agile elements from the respective process models, taking specific criteria into account and within the framework of a systematic procedure. The extraction and its result, an overview of agile practices, are presented in Chapter 3.1.
2.2. Fundamentals of culture

There is no uniform definition of the term culture in the literature. Various studies indicate that the context of the respective work is decisive for the definition of the cultural concept (c.f. [Ham07], [KK52], [Oli95]). The following definition, according to Oettinger in [Oet93, p. 41] is used for the present paper: „... the term is used to describe the customs, beliefs, social structure, and activities of any group of people who share a common identification and who would label themselves as members of that group.“.

As with the elements of agile methodologies (see chapter 2.1), it is necessary to make a contextual limitation concerning the culture. To describe cultural contexts, Karahanna et al., in [KES06], the model of the connected cultural levels is presented. It makes sense to adopt this concept for this work. The problem here is that there are different definitions in the literature for the term culture level (cf. Hofstede in [Hof93, p. 25] and Karahanna et al. in [KES06]). Due to this, the term for this work is defined as follows:

**Definition 2 (Cultural levels)** Cultural levels categorize cultures based on their context.

*Cultural levels* create the opportunity to view and examine different cultures in a contextually differentiated manner. They help to separate cultures from one another and provide a frame of reference for classifying specific cultures. There may be a logical connection between cultural levels. Several authors describe that the levels can influence each other (cf. Hofstede in [Hof93, p. 25], Karahanna et al. in [KES06], Scholz in [Sch00], Walsh and Kefi in [WK08]).

*Cultural levels* create the opportunity to view and examine different cultures in a contextually differentiated manner. They help to separate cultures from one another and provide a frame of reference for classifying specific cultures. There may be a logical connection between cultural levels. Several authors describe that the levels can influence each other (cf. Hofstede in [Hof93, p. 25], Karahanna et al. in [KES06], Scholz in [Sch00], Walsh and Kefi in [WK08]).

2.3. Causal and impact models

According to Russo in [Rus11], *causal and impact models* are mostly used to depict causal or non-causal relationships between variables with the help of a formal structure. According to Russo, in [Rus11], both a formal and graphic representation are usually used. According to Greenland and Brumback in [GB02] there are various forms of *causal and impact models*, for example, „Graphical models“, „Potential-outcome (counterfactual) models“, and „Multifactorial causation and the sufficient-component cause model. “
Causal and impact models are used in software engineering to describe causal relationships and influences. Diebold and Zehler use a causal and impact model in [DZ15] to depict the effects of agile practices on various characteristics, such as product quality, customer satisfaction, or development costs. In [LM10], Lamersdorf and Münch describe the relationships between different facets in the context of global software development using a causal model. These facets are, for example, the distance or linguistic differences in the software development teams.

Estimating the development costs of software is another area of application for these models in software engineering. Trendowicz uses a causal model for his CoBRA method for cost estimation presented in [Tre13]. Lederer and Prasad present a hypothetical model in [LP98], which shows the potential causes of incorrect estimates of software development costs. Furthermore, causal and impact models are used in the context of software metrics (cf.[FN00], [FKN02]).

Causal models are also used in other specialist areas, such as medicine (cf. [GB02], [IOE98], [Lef00]), marketing ([HCL96]) or also political science ([MC67]).

For the present work, the form of causal and impact models is suitable for modeling the theoretical model. This is justified in particular by the fact that this form of modeling enables the cultural influence on agile elements to be represented. Furthermore, they are already used in software engineering and also in the context of agile software development. When creating the model for this work, the Agile Practices Impact Model (APIM) by Diebold and Zehler in [DZ15] is formally oriented.

### 3. Model of Cultural Influences on Agile Elements

#### 3.1. The dimension of Agile Elements

We create a dimension of agile elements (see chapter 2.1) for the theoretical model. The agile elements are described separately from their affiliation to agile methodologies. They are united under one general term based on their essence. This reduces the complexity resulting from the variance in the description of agile characteristics in the literature. With the help of the dimension of agile elements, an allocation of the influences of cultural levels is made possible.

The following subsection describes the motivation for extracting agile elements. It explains why no existing overview of agile elements is used and explains why it makes
sense to extract your own. The following section presents the systematic procedure for extraction. In this context, the results of the individual steps are presented and explained. Finally, the overview of extracted agile elements is presented. In this context, the categories and the agile characteristics contained therein are individually defined and explained.

3.1.1. **Motivation to extract agile elements**

There are three reasons to create a new overview of generally described agile elements:

1. The abovementioned extensive variety of agile elements in the literature is also accompanied by a variety of definitions of agile practices and roles (see Chapter 2.1). Several overviews of agile practices have been published in the past (see Table 1). These differ in terms of the number and description of agile practices.

To identify the existing overviews of agile practices, a “backward snowballing,” according to Wohlin in [Woh14] durchgeführt. In the course of the backward snowballing, the references of the primary studies of the SLR von Neumann from [Neu20] were examined about existing overviews of agile practices. First, the references that were already in the SLR's result set were excluded. The following inclusion and exclusion criteria from [Neu20] were used for the further selection:

- Inclusion criteria: IC2, IC3
- Exclusion criteria: EC1, EC2, EC3

If the respective referenced source fulfills these criteria, the content was examined for an existing (and thus referenced) or newly created overview of agile practices.

Six new sources were identified through backward snowballing. These six sources and the assignment to the primary study are shown in the following table:

<table>
<thead>
<tr>
<th>#</th>
<th><strong>Authors</strong></th>
<th><strong>Title</strong></th>
<th><strong>Identified in the primary study</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>DD14</td>
<td>Diebold, Philipp; Dahlem, Marc</td>
<td>Agile practices in practice: A mapping study</td>
<td>[ABG17]</td>
</tr>
<tr>
<td>KMP12</td>
<td>Kurapati, Narendra; Manyam, Venkata Sarath Chandra; Petersen, Kai</td>
<td>Agile Software Development Practice Adoption Survey</td>
<td>[OK18]</td>
</tr>
</tbody>
</table>
The five sources [DD14], [JW10], [JW12], [KMP12], and [Sid07, p. 87ff.] have been identified from the references of the primary studies from [Neu20]. Only the publication by Williams [Wil10] has additionally been identified in the references of the first five sources mentioned.

In addition to the variance of agile practices in these overviews described at the beginning of this subchapter, the research contexts and goals in the individual publications naturally differ from one another. Kurapati et al. [KMP12] and Sidky [Sid07, p. 87ff.] refer to the introduction and use of agile methodologies in practice. In his work [Sid07, p. 6f.], Sidky presents a systematic procedure for introducing agile methods in companies. Kurapati et al. deal in [KMP12], among other things, with the questions of which agile practices are combined in practice or to what extent XP and Scrum are used. Jalali and Wohlin deal with the context of global software engineering and want to answer the question in which framework of global software engineering and under which circumstances which agile practices are successfully used within the framework of a systematic literature search [JW10], [JW12].

Besides, the systematic procedure concerning the creation of some overviews and the database used is at least partially not shown (e.g., Diebold and Dahlem in [DD14], Jalali and Wohlin in [JW10]). In the publications by Kurapati et al. in [KMP12], Sidky in [Sid07, p. 87ff.], and Williams in [Wil10], the database of the respective overview is comprehensible. Also, Kurapati et al., the systematic procedure is described in [KMP12] and Sidky in [Sid07, p. 87ff.].

Another reason is that agile practices are defined differently in the regulations of different process models. An example of this is the specification of how agile practices are to be carried out. The rules on how the daily stand-up meeting in Scrum (see Definition of Daily Scrum in [SS17]) and Kanban (see Daily Standup Meeting in [And11]) is to be carried out differ from one another, for example, concerning the process. However, the objective of this practice appears at least similar. There are also differences concerning the procedural descriptions. The Scrum Guide [SS17] defines the Sprint (iteration) as an artifact. The Test First
approach is followed in XP [Bec00]. Besides, there is a difference in the description of agile practices in connection with the value-based basis. In XP, this is shown, for example, by the practice of collective ownership. With Kanban, on the other hand, most of the practices are described in terms of process optimization. The differences in the description and definition of agile practices in the respective sets of rules can also be related to the basis and origin. While XP and Scrum were developed and defined by initiators who were also involved in determining the agile manifesto, the source of Kanban in software development cannot be associated with them. Also, the significant differences in the XP and Scrum process models can be explained by the fact that the initiators have different professional backgrounds. The objective of developing these procedural models was other.

From the author's point of view, the reasons described representing a sufficient justification for creating a new overview of agile elements for the present work. The following describes the systematic procedure for extracting agile elements. In this context, the interim results of the four steps are presented, and the decisions made in each case are explained. This description of the procedure guarantees the traceability of the extraction and list creation. The list of extracted agile practices is then presented and explained.

3.1.2. Systematic extraction of agile elements

The basis for the extraction is the lists of agile practices from the publications mentioned above as well as the agile practices of the methodologies Scrum [SS17], XP [Bec00] and Kanban [And11] defined in the guidelines.

The extraction of agile elements is carried out in five steps. The result of this procedure is the list of extracted agile elements. The following description of the systematic extraction is also based on this structure.

1. Identification and listing of the agile practices redundantly listed in the various publications
   In this first step of the extraction, the lists of agile practices in the publications mentioned above are compared with one another. The agile practice is considered to be listed more than once if it can be identified in at least two publications by different authors. It must be taken into account here that similar agile practices are listed under other terms. An example of this is the planning meeting. This is referred to by Jalali and Wohlin as well as Diebold and Dahlem as a planning meeting. Sidky runs it as collaborative planning and
Kurapati as a sprint planning meeting. One reason for this is that some authors have chosen new names for the agile practices when creating a separate list to be able to describe and list them separately from the agile approaches. Likewise, in some cases, several agile practices or elements of these have been combined under a more general term. The identified agile practices are also referred to for this work regardless of their description in the rules and regulations of the respective process models (see Table 2).

<table>
<thead>
<tr>
<th>#</th>
<th>Agile Practice</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Continuous Integration and Builds</td>
<td>Continuous Integration: [JW10], [JW12], [KMP12], [Sid07, p. 87ff.], [Wil10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuous / automated builds: [JW12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuous Integration/deployment: [DD14]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nightly Build: [Wil10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ten Minutes Build: [Wil10]</td>
</tr>
<tr>
<td>2</td>
<td>Daily Standup Meetings</td>
<td>Standup Meeting(s): [JW10], [JW12], [Wil10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stand Ups: [KMP12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Daily progress tracking meetings [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Daily discussion [DD14]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time boxing [DD14]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scrum Meeting: [Wil10]</td>
</tr>
<tr>
<td>3</td>
<td>Pair Programming</td>
<td>Pair Programming: [JW10], [JW12], [KMP12], [Wil10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Paired programming: [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td>4</td>
<td>Retrospective</td>
<td>Retrospective(s): [JW10], [JW12], [KMP12], [Wil10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reflect and tune process: [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Learning Loop: [DD14]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time boxing: [DD14]</td>
</tr>
<tr>
<td>5</td>
<td>Test</td>
<td>Test Driven Development: [JW10], [JW12], [KMP12], [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td>8</td>
<td>Planning meeting</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Using and maintaining a list of all features and their status (backlog)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Review Meeting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time boxing: [DD14], [JW12]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Continuous specification: [DD14]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Evolving and hierarchical specification: [DD14]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Backlog: [JW10], [JW12]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sprint Review Meeting: [KMP12]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Iteration Demonstration: [Wil10]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Continuous customer satisfaction feedback: [Sid07, p. 87ff.]</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Automation-Driven Root Cause Analysis of Failures: [Wil10]</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Acceptance Test Driven Development: [Wil10]</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Unit Test Driven Development: [Wil10]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Code and Tests: [Wil10]</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Continuous customer satisfaction feedback: [Sid07, p. 87ff.]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Evolutionary requirements: [Sid07, p. 87ff.]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Agile project estimation: [Sid07, p. 87ff.]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adaptive planning: [Sid07, p. 87ff.]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Collaborative planning: [Sid07, p. 87ff.]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Negotiated scope: [Wil10]</td>
<td></td>
</tr>
</tbody>
</table>

**Code and Tests**

- [Wil10]
| 9 | Iteration based process | Plan features, not tasks: [Sid07, p. 87ff.]
Planning at different levels: [Sid07, p. 87ff.]
Planning meeting: [DD14], [JW10]
Sprint Planning Meeting: [KMP12]
Time boxing: [DD14]
Wideband Delphi Estimation: [Wil10]

| 10 | Customer integration and collaboration | Short iteration: [JW10], [Wil10]
Sprint / Iterations: [JW12], [KMP12], [Wil10]
Client based iterations: [Sid07, p. 87ff.]
Risk driven iterations: [Sid07, p. 87ff.]
Time boxing: [DD14]
Close collaboration: [JW10]
Customer immediately accessible: [Sid07, p. 87ff.]
Customer contract revolving around commitment of collaboration: [Sid07, p. 87ff.]
Customer contract reflective of evolutionary development: [Sid07, p. 87ff.]
Customer commitment to work with developing team: [Sid07, p. 87ff.]
Customer involvement: [DD14]
Onsite / proxy customer: [JW10], [JW12], [KMP12]

| 11 | Coding standards | Code standards: [JW10], [JW12] |
| 11 | Coding standards: | Coding standards: [KMP12], [Sid07, p. 87ff.]
| 12 | Refactoring | Refactoring: [DD14], [JW10], [JW12], [KMP12]
|    | Continuous improvement (Refactoring): | Continuous improvement (Refactoring): [Sid07, p. 87ff.] |
| 13 | Tracking progress | Burndown Charts: [JW10], [JW12]
<p>|    | Virtual Scrum Wall: | Virtual Scrum Wall: [JW10] |
|    | Informative Workspace: | Informative Workspace: [Wil10] |
|    | Tracking iteration progress: | Tracking iteration progress: [Sid07, p. 87ff.] |
|    | Progress monitoring: | Progress monitoring: [DD14] |
|    | Tracking progress (tracking of progress of project): | Tracking progress (tracking of progress of project): [KMP12] |
| 14 | Communication | Communication: [KMP12] |
|    | Close collaboration: | Close collaboration: [JW10] |
|    | Instant messages: | Instant messages: [JW10] |
|    | Frequent face-to-face communication: | Frequent face-to-face communication: [Sid07, p. 87ff.] |
|    | Unattached communicative teams: | Unattached communicative teams: [DD14] |
| 15 | Metaphor / Vision | Metaphor: [KMP12] |
|    | System metaphor: | System metaphor: [JW10] |
|    | Product vision: | Product vision: [DD14] |
| 16 | Office structure | Frequent (collocated) face-to-face interaction between developers &amp; users: [Sid07, p. 87ff.] |
|    | Ideal agile physical setup: | Ideal agile physical setup: [Sid07, p. 87ff.] |
|    | One Team / sit together: | One Team / sit together: [JW12] |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>17</strong></td>
<td><strong>Empowered and motivated team</strong></td>
</tr>
<tr>
<td></td>
<td>Office (office structure that support agile development): [KMP12]</td>
</tr>
<tr>
<td></td>
<td>Sit together: [Wil10]</td>
</tr>
<tr>
<td></td>
<td>Collaborative teams: [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td></td>
<td>Empowered and motivated team: [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td></td>
<td>Self-organizing teams: [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td></td>
<td>Small cross-functional teams: [DD14]</td>
</tr>
<tr>
<td></td>
<td>Task volunteering: [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td></td>
<td>Whole Team: [Wil10]</td>
</tr>
<tr>
<td><strong>18</strong></td>
<td><strong>Small and frequent releases</strong></td>
</tr>
<tr>
<td></td>
<td>Continuous delivery: [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td></td>
<td>Delivering frequent releases: [DD14]</td>
</tr>
<tr>
<td></td>
<td>Short/small releases: [KMP12], [Wil10]</td>
</tr>
<tr>
<td></td>
<td>Smaller and more frequent releases (4-8 weeks): [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td><strong>19</strong></td>
<td><strong>Energized Work</strong></td>
</tr>
<tr>
<td></td>
<td>40 hour week: [KMP12]</td>
</tr>
<tr>
<td></td>
<td>Energized Work: [Wil10]</td>
</tr>
<tr>
<td></td>
<td>Sustainable pace: [Wil10]</td>
</tr>
<tr>
<td><strong>20</strong></td>
<td><strong>Collective Code Ownership</strong></td>
</tr>
<tr>
<td></td>
<td>Collective Code Ownership: [Wil10]</td>
</tr>
<tr>
<td></td>
<td>Collective Ownership: [KMP12]</td>
</tr>
<tr>
<td><strong>21</strong></td>
<td><strong>Documentation</strong></td>
</tr>
<tr>
<td></td>
<td>Agile documentation: [Sid07, p. 87ff.]</td>
</tr>
<tr>
<td></td>
<td>Enough documentation: [JW12]</td>
</tr>
<tr>
<td></td>
<td>Documentation: [KMP12]</td>
</tr>
<tr>
<td><strong>22</strong></td>
<td><strong>Software configuration management</strong></td>
</tr>
<tr>
<td></td>
<td>Configuration and Change Management: [KMP12]</td>
</tr>
<tr>
<td></td>
<td>Software configuration management: [Sid07, p. 87ff.]</td>
</tr>
</tbody>
</table>
In the first step, 21 agile practices were identified, which are listed several times. It should be noted that the same agile practices are referred to differently in the various publications (see Table 2). In some practices, too, several assignments have been made from one publication. An example of this is the agile practice of Time boxing mentioned by Diebold and Dahlem. The assignments for each agile practice are explained separately below:

### Continuous integration and deployment

In [DD14], Diebold and Dahlem combine the agile practice of Continuous integration with the practice of Continuous deployment and use a combinatorial term for this practice. In the present work, Continuous deployments are not used concerning this practice, as it is assumed that this is taken into account in the agile practice of small and frequent releases.

Williams describes two different practices in this context in [Wil10]: Continuous integration and Nightly Build. However, she also notes that agile practices should be used in combination. According to Williams, the combined application of both practices provides the daily availability of the current development status of the software in test environments. According to Williams, Nightly builds also offers the possibility that the automated test cases can be executed as part of the build process, thereby increasing the quality of the software. In [Sid07, p. 102], Sidky also describes continuous integration in combination with regular, automatic build processes: "It is preferred that each integration is verified by an automated build tool in order to detect any integration errors as quickly as possible." Jalali and Wohlin also list in their overview
in [JW12] with *Continuous / automated build* an agile practice in this context. With the practice "*Ten Minutes Build,*" Williams leads another practice in [Wil10]. It describes the short duration of the build process so that it can be carried out as often as possible. This enables faster integration of new source codes.

Jalali and Wohlin in [JW10] and Kurapati et al. in [KMP12], *Continuous Integration* is also listed as an agile practice in their overviews.

**Daily Standup Meeting**

Stand-up meetings (referred to as *Stand Ups* for short; cf. [KMP12]) in the context of agile methodologies are usually understood to mean daily meetings, e.g., to synchronize the development team [SS17]. For this reason, this practice is referred to as the *Daily Standup Meeting* in the present work. The assignment to the *Daily discussion* practice mentioned by Diebold and Dahlem in [DD14], as well as the *Daily progress tracking meetings* mentioned by Sidky in [Sid07, p. 87ff.], takes place on this basis. In addition to the agile, practical Scrum Meeting, Williams also leads the *Stand-Up Meeting* in their overview in [Wil10]. It describes the content of the Daily Scrum procedure. She concretizes this practice as a stand-up meeting by addressing the peculiarity of standing during the meeting. Williams recommends that the meeting be conducted in a room without chairs. Jalali and Wohlin also use the stand-up meeting practice in their overviews in [JW10], [JW12]

In this work, time boxing is taken into account the agile practices that are carried out as a meeting. Time boxing is understood as a restriction concerning the application of agile practices and is therefore not considered as an independent practice. Diebold and Dahlem list this aspect as a separate agile practice [DD14] but do not justify it. This assignment is described once for the *Daily Standup Meeting* practice.

**Pair Programming**

Pair programming is an agile practice of XP [Bec00].

The practice is not listed in Diebold and Dahlem's list [DD14] Sidky calls this paired programming in [Sid07, p. 87ff.]. The authors Jalali and Wohlin in [JW10], [JW12], Kurapati et al. in [KMP12], as well as Williams in [Wil10], have taken over the term from XP and partly refer to XP in the description of the practice (e.g., Williams in [Wil10]).
Retrospective

In agile methods, the Retrospective often has the goal of reflecting on the underlying procedure and building on it to improve it (see Sprint Retrospective [SS17]). For this reason, those of Dahlem and Diebold in [DD14], Kurapati et al. in [KMP12], and Sidky in [Sid07, p. 87ff.] mentioned agile practices in the context of process reflection and optimization. Jalali and Wohlin in [JW10], [JW12] name the practice as Retrospectives in their reviews.

Williams' description in [Wil10] includes other aspects in addition to the stated goals concerning process optimization. She says, for example, that the discussion will discuss what was processed in the previous iteration or how many errors were identified.

Since this practice is a meeting within the framework of Scrum [SS17] that is subject to time boxing, this practice has also been assigned from the list by Diebold and Dahlem [DD14].

Test (Test Driven Development / Test First)

Test Driven Development is known as an agile practice (cf. [Bec15], [GW04], [JS05], [MW03], [WMV03]). A central component here is the Test First approach, which is also described in XP as an agile practice [Bec00]. For this paper, this Test First approach is described as an agile practice under the term Test Driven Development.

Jalali and Wohlin in [JW10], [JW12], Kurapati et al. in [KMP12], and Sidky in [Sid07, p. 87ff.] list Test Driven Development as an agile practice in their overviews. Williams, in [Wil10] also describes the Test First approach under the name Code and Tests. Williams distinguishes between two types of test in [Wil10]:

- **Acceptance testing**

  In [Wil10] regarding [IEE90], Williams defines Acceptance testing as „… a formal process that is conducted to determine whether or not a system satisfies a set of criteria (i.e., “acceptance criteria”) that are predetermined by the customer to enable the customer to determine whether or not to accept a system“. 
According to Williams in [Wil10] from [Bec05], [JAH01], the test-driven character of Acceptance testing is achieved in that the acceptance tests are carried out jointly by the various roles (product manager, customer, software tester, and developer as well as user interface designer) The beginning of the iteration. Each role takes on a related activity. It is up to the product manager to formulate the technical specifications. For the software tester, for example, these form the basis for creating test cases.

According to Williams, the tasks of the software tester include the acceptance tests and other test activities (system tests, integration tests). The author points out that the test cases should mainly be able to be carried out automatically. This also applies to the acceptance tests. Williams points out in [Wil10] that this particular form of Test Driven Development optimizes communication and cooperation in the entire development team. It also points out the early involvement of the role of the software tester, which otherwise typically becomes active later in software development projects.

Both Jalali and Wohlin in [JW10], [JW12], and Kurapati et al. in [KMP12] include acceptance tests as agile practices in their overviews.

- **Unit testing**
  Williams describes Unit Tests in [Wil10] as the technical activity of implementing automated tests of individual components. These serve as the basis for the implementation of the source code. Williams points out that the implementation of unit tests and source codes is carried out cyclically in alternation. About several studies ([BN06], [NMB+08], [SWM07], [WKN09], the author also describes the advantages of this practice. According to Williams, Unit Tests improve quality, for example, by reducing errors.

  Sidky defines unit tests in [Sid07, p. 102f.] As an agile practice. He describes unit tests as the validation of the smallest testable part of a software application. Like Williams, he also points out the automation
of unit tests. In addition to Williams and Sidky, unit tests are also included in the overviews by Jalali and Wohlin in [JW10], [JW12], and Kurapati et al. in [KMP12]. The authors also point out automated tests.

Sidky defines Unit Tests in [Sid07, p. 102f.] as an agile practice. He describes unit tests as the validation of the smallest testable part of a software application. Like Williams, he also points out the automation of unit tests. In addition to Williams and Sidky, unit tests are also included in the overviews by Jalali and Wohlin in [JW10], [JW12], and Kurapati et al. in [KMP12]. The authors also point out automated tests.

Williams names another agile practice in [Wil10]: Automation-Driven Root Cause Analysis of Failures. Williams describes how to identify the causes of adverse events. For this purpose, it names, for example, errors in the application and the lack of implementation of requirements. The practice is, therefore, also assigned to the test practice.

In [DD14], Diebold and Dahlem describe the aspect of quality assurance more generally. To do this, you have two practices: Quality check and validation practice.

**Review Meeting**

The review meeting enables the person responsible for the product (e.g., product owner in Scrum or the customer) to examine and evaluate the results of the previous iteration [SS17]. In [DD14], Diebold and Dahlem name this practice as an outcome review. Like the retrospective practice described above, this agile practice is also described in Scrum as a ceremony [SS17]. That is why Diebold and Dahlem assign the quality of Time boxing to the review meeting in this work.

Jalali and Wohlin in [JW10], [JW12], and Kurapati in [KMP12] refer to the agile practice Sprint Review described in Scrum (c.f. [SS17]). In [Sid07, p. 87ff.], Sidky names this agile practice as Continuous customer satisfaction feedback. Williams uses the term Iteration Demonstration for this in [Wil10].

**Using and maintaining a backlog (list of all requirements) and their status**

The use of backlogs is common in agile methods (cf. the guidelines of Scrum [SS17] and Kanban [And11]). In this paper, various forms of backlog, such as
the product or sprint backlog in Scrum, are understood. Therefore, in addition to the allocation based on the general term backlog, listed by Jalali and Wohlin in [JW10], [JW12] Williams' mention of the release and iteration backlog in [Wil10] is also taken into account. The importance of using a backlog is also shown by Sidky in [Sid07, p. 101]. In [Sid07, p. 101], he describes the agile practice as maintaining a list of all features and their status (i.e., backlog). The practice includes creating and maintaining the backlog. He specifies administration in [Sid07, p. 101] as "adding, removing, specifying, updating, and prioritizing the backlog items". According to Sidky, it is less critical who (or which role) gets the authority to carry out these activities; much more is to decide that a backlog exists.

According to various authors, a backlog is a list of technical and business requirements (c.f. [Sid07, p. 101], [Wil10]). Backlogs are living artifacts; they are subject to changes, for example, in terms of content or order. Sidky lists these requirements as Evolutionary requirements in [Sid07, p. 97]. In [Sid07, p. 97], he classifies evolutionary requirements as "the most important agile practice that helps the delivery of software incrementally in shorter iterations". The evolutionary characteristic is described by Sidky's assumption that requirements can be changed. According to Sidky, the continuous development of requirements based on customer feedback means that the highest value of the software to be implemented is achieved for the client (or customer).

Diebold and Dahlem also name the continuous analysis (and change) of the requirements in [DD14] as ongoing specification analysis. The authors also describe in [DD14] the agile practice Evolving and hierarchical specification. The aspect of the hierarchy, which Diebold and Dahlem name, can be related to the urgency of a requirement as well as to the status. This depends, for example, on the particular application for use and the form of the backlog. Regardless of the exact consideration, e.g., which form of the backlog can influence the hierarchy, the assignment makes sense at this point.

Planning Meeting

The Planning Meeting is listed as an agile practice by various authors (cf. [DD14], [JW10]) Kurapati et al. describe in [KMP12] the practice regarding the Scrum ceremony Sprint Planning Meeting [SS17]. Sidky describes the practice as Adaptive Planning in [Sid07, p. 107].
The organization of the next iteration is the focus of the planning meeting. This requires the estimation of individual requirements (e.g., product backlog items). For this purpose, Sidky describes the practice of *Agile Project Estimation* in [Sid07, p. 110]. Estimates in agile software development often have a playful character (see, for example, the agile practice Planning Game in XP [Bec00]).

Sidky also describes in [Sid07, p. 100f.] the practice of considering planning features, not tasks. According to the author, this makes the feature easier to understand for the customer, since tasks are often formulated in the technical language of software development. This, in turn, leads to a customer-oriented development process, in which aspects such as the focus on (business) values are made possible.

Furthermore, in [Sid07, p. 98], he explains the need to consider different levels in project planning. He describes this agile practice as planning at different levels. According to Sidky in [Sid07, p. 98], agile software development usually has two levels: "Release planning (dealing with the overall product)" and "iteration planning (dealing with current iteration)."

Sidky also points out in [Sid07, p. 92f.] that planning should be carried out by all stakeholders\(^1\) (customer, developer, manager) of the project. He calls this collaborative planning. According to the author, using this practice leads to employees feeling involved. The practice also leads to improved communication and collaboration between stakeholders.

The time boxing practice listed by Diebold and Dahlem in [DD14] is also taken into account here, as is the case with the retrospective and review meeting practices. This is justified in particular by the fact that Kurapati et al. conduct the practice as a Scrum ceremony in their overview. Williams also points to this aspect in [Wil10]. She calls this practice in the context of planning iterations Negotiated Scope.

---

\(^1\) Sidky mentions as a stakeholder in [Sid07, p. 92f.] for example: "...client, the developers, and the managers".
this aspect in [Wil10]. She calls this practice in the context of planning iterations *Negotiated Scope*.

**Iteration based process**

Agile software development is often carried out iteratively. Examples of agile methodologies that provide an iterative process are Scrum (definition of Sprints in [SS17]) and XP [Bec00]. Concerning Scrum, several authors refer to this practice under the name *Sprint* in their overviews ([JW12], [KMP12], [Wil10]). Jalali and Wohlin in [JW10], as well as Williams in [Wil10], also use the term *Short iteration* for this practice. Williams calls iterations referring to [Lar04] as "... mini-project ...". At the end of each iteration, potentially deliverable “working code” is provided by the team.

In [Sid07, p. 105], Sidky points out for this practice that the customer must be able to influence the requirements to be implemented for each iteration. He justifies this with the prioritization of the backlog, which he controls. From his point of view, the iterative approach in agile software development is essential to be able to react quickly to changing circumstances. Sidky defines this practice in [Sid07, p. 105] as *Client Driven Iterations*. He also describes in [Sid07, p. 100] the practice of performing iterations in a risk-controlled manner. He calls this Risk Driven Iterations. According to Sidky in [Sid07, p. 100], this has the advantage that risks can be identified at an early stage, making the development process more effective.

According to Williams, in [Wil10], iterations in agile teams are always limited in time. This property also describes sets of rules of agile methodologies (e.g., Scrum; see [SS17]). Time boxing is, therefore, also assigned to this practice by Diebold and Dahlem from [DD14].

According to Williams, in [Wil10], iterations in agile teams are always limited in time. This property also describes sets of rules of agile methodologies (e.g., Scrum; see [SS17]). Time boxing is, therefore, also assigned to this practice by Diebold and Dahlem from [DD14].

**Customer integration and collaboration**

The integration and cooperation with the customer is a success factor of agile software development (c.f. [BSE+16], [BZT+14], [CC08], [GKS18], [HNM11], [MKK09], [OK18], [SSK20], [STE17], [TMO+20]) and is described in agile methods (e.g. on-site customer in XP; see [Bec00]). Several authors ([JW10],...
[JW12], [KMP12]) take this practice into account in their overviews and focus on accessibility and integration. Diebold and Dahlem refer to this in [DD14] as *Customer involvement*. In their overview in [JW10], Jalali and Wohlin lead the practice as a close collaboration. Sidky explains into the practice under the designation *Customer immediately accessible*.

The integration and cooperation with the customer is a success factor of agile software development (see chapter 4.4.2) and is described in agile methodologies (e.g., on-site customer in XP; see chapter 2.2.2). Several authors ([JW10], [JW12], [KMP12]) take this practice into account in their overviews and focus on accessibility and integration. Diebold and Dahlem refer to this in [DD14] as customer involvement. In their overview in [JW10], Jalali and Wohlin lead the practice as a close collaboration.

According to Sidky in [Sid07, p. 96], the basis for successfully integrating the customer into the agile procedure is the customer's commitment to work with the development team. He describes this practice as customer commitment to work with a developing team and describes it as a prerequisite for two other agile practices that are important in this context and are therefore assigned here. On the one hand, there is the practice of customer contract reflective of evolutionary development. Sidky describes in [Sid07, p. 99] the importance of the customer's understanding of the type of iterative procedure. This goes hand in hand with the fact that the requirements should not be fully recorded and documented at the beginning of the project, but the customer should also be open to change.

According to Sidky in [Sid07, p. 96], the basis for successfully integrating the customer into the agile procedure is the *Customers commitment to work with the development team*. He describes this practice as a prerequisite for two other agile practices that are important in this context and are therefore assigned here. On the one hand, there is the practice of *Customer contract reflective of evolutionary development*. Sidky describes in [Sid07, p. 99] the importance of the customer's understanding of the type of iterative procedure. This goes hand in hand with the fact that the requirements should not be fully recorded and documented at the beginning of the project, but the customer should also be open to change.
As a second associated agile practice, Sidky explains in [Sid07, p. 108f.] that the teamwork with the customer is more important than the focus on requirements. Sidky justifies this with the fact that successful cooperation creates the possibility of accepting changes. He describes this agile practice as Customer contract revolving around commitment of collaboration.

Coding standards

In [Wil10], Williams refers to the agile practice of Collective Code Ownership (also described for XP in [Bec00]). In this context, it indicates the use of Coding Style Guidelines. This practice is assigned here because, according to Williams, uniform specifications regarding the source code are made in the guidelines. According to Williams, these guidelines allow software developers to familiarize themselves more easily with and edit the source code of team members.

Similarly, Sidky explains the practice in [Sid07, p. 94f.] under the name Coding Standards. He explains that coding standards lead to a uniform language between the software developers in the team. Referring to other authors ([Hun06], [NM01], [Wak01], he also describes the importance of this agile practice for the cooperation between the individual team members. While Kurapati et al. in [KMP12] use the same name as Sidky for this practice, Jalali and Wohlin use Code standards in [JW10], [JW12].

Refactoring

Refactoring is understood to mean the continuous optimization of the source code (see [Bec00]). Sidky describes Refactoring in [Sid07, p. 102] as “cleaning up the code.” As an example, he gives the reason that a redundant source code is removed. In [Sid07, p. 102], Sidky uses the term Continuous Improvement for agile practice and uses Refactoring as a synonym.

Williams describes Refactoring in [Wil10] as essential for the practice of Incremental Design. She also describes refactoring as a practice, which is why it is assigned here.

Jalali and Wohlin in [JW10], [JW12]. Kurapati et al. in [KMP12] as well as Diebold and Dahlem in [DD14] Refactoring, also has an agile practice in their lists.

Monitoring of progress
For the monitoring of progress, various practices are described in the guidelines of agile methods (e.g., Kanban Board in [And11]). Various authors have listed this practice in their overviews: Sidky names it in [Sid07, p. 98] as Tracking iteration progress. Diebold and Dahlem use the term Progress monitoring in [DD14], and Kurapati et al. refer to the practice in [KMP12] as Tracking progress (tracking of progress of project). The authors mentioned the use of general terms for this practice.

Jalali and Wohlin, on the other hand, use the term Burndown Charts in their overviews in [JW10], [JW12]. Burndown Charts are defined in Scrum [SS17]. Williams also describes Burndown Charts in [Wil10] as examples for monitoring the progress towards target achievement. She calls this agile practice the Informative Workspace. From their point of view, this practice helps team members to get an overview of the project status as quickly as possible. She also mentions the Iteration status board as a common practice. For this purpose, Williams in [Wil10] also describes the possibility that the information can also be provided digitally. Jalali and Wohlin also take up this possibility in [JW10]. The authors also conduct the Virtual Scrum Wall practice there.

Communication

Communication is a success factor in agile software development (c.f. [BMP+19], [CC08], [LHC+19], [MKK09], [RRO+19], [SSK20]). Communicative aspects are described as agile practices by various authors (vgl. [DD14], [JW10], [KMP12]).

Sidky explains in [Sid07, p. 101] the connection between the efficiency and effectiveness of the development process and the efficiency and effectiveness of communication between team members. Regarding [Amb20] and [Coc01], he states that there is a connection between direct communication between team members and the effectiveness and efficiency of communication. In [Sid07, p. 101], Sidky describes the agile practice as frequent face-to-face communication.

Metaphor / Product vision

The agile methodologies Scrum and XP each describe an agile practice that sets out the goal of the project and describes the content as simply as possible. In Scrum, this practice is called the Product Vision [SS17]. In XP, the practice is named Metaphor [Bec00].
Diebold and Dahlem list in their overview in [DD14] the agile practice Product vision. The authors Jalali and Wohlin in [JW10] (System metaphor) and Kurapati et al. in [KMP12] (Metaphor) use the XP formulation.

Diebold and Dahlem list in their overview in [DD14] the agile practice Product vision. The authors Jalali and Wohlin in [JW10] (System metaphor) and Kurapati et al. in [KMP12] (Metaphor) use the XP phrasing.

Office structure

The workplace environment influences the success of agile approaches (c.f. [BZT+14], [RKK16]). Sidky lists in [Sid07, p. 110f.] the agile practice Ideal physical setup. In this context, he explains that the entire team should sit and work together in one room. Sidky also explains in [Sid07, p. 110f.] that the office space should be designed openly, i.e., no partition walls or cubicle-like subdivisions are used. Sidky justifies this with the fact that the team members can communicate more easily with one another. Williams also refers to this aspect in [Wil10] regarding [All95] and [OO01] when describing the agile practice Sit Together. Jalali and Wohlin also mention this agile practice in [JW12].

About [Coc01] and [Hig02], Sidky explains that the furnishing of the room (e.g., technical equipment or the arrangement of the tables) must be taken into account. Kurapati et al. also mention the agile practice Office in their overview in [KMP12]. The authors explain that the physical structure of the office supports agile development.

In his list in [Sid07, p. 112], Sidky also takes into account the agile practice Frequent (collocated) face-to-face interaction between developers & users. From his point of view, it is "ideal" when developers and customers or users sit together in one room [Sid07, p. 112]. In the opinion of Sidky in [Sid07, p. 112], this leads to immediate feedback and "incredible communication."

Empowered and self-organizing team

For the organizational characteristics of teams in agile software development, the authors describe various agile practices (e.g. [KMP12]). Sidky explains in [Sid07, p. 93] the practices of Collaborative teams und Empowered and motivated teams. In this regard, he refers to the definition of Collaborative team by Tabaka in [Tab05]. Sidky points out that empowering teams is particularly important for making their own decisions.
Sidky concretizes these practices in his work in [Sid07, p. 101] and names the aspect of the self-organization of agile teams. According to Sidky in [Sid07, p. 101], self-organized teams make their own decisions without having to consult with management. Sidky also explains in [Sid07, p. 95] that the team members can choose the tasks themselves. These are, therefore, not assigned by stakeholders such as project managers. Sidky calls this practice *Task volunteering*.

Sidky also describes in [Sid07, p. 101] that these (self-organizing) teams are often *cross-functional* and the roles of individual team members are not clearly defined. Williams also takes up the *cross-functional* property of agile teams in her explanation of the agile practice she calls the *Whole Team* in [Wil10]. From her point of view, this practice ensures that the members of the agile team cover all necessary skills to be able to develop the product successfully. Diebold and Dahlem have also listed the agile practice of *Small cross-functional teams* in their overview in [DD14].

**Small and frequent releases**

Sidky divides this practice into two practices. For the agile practice of *Continuous delivery*, he describes in [Sid07, p. 98] that the focus is on the regularity of the completion of the product increments. As a prerequisite for this, he states that an increment is provided for each iteration.

In the second agile practice, *Smaller and more frequent releases (4-8 weeks)*, Sidky explains in [Sid07, p. 106f.] that the product increments must also be made available to the customer regularly. This is the only way to achieve the desired effects (such as quick feedback from the customer). Williams also leads this practice in [Wil10] under the name *Short releases*. While Sidky gives 4-8 weeks as the time component, Williams describes this. Three months. In addition to the two authors, Diebold and Dahlem in [DD14] and Kurapati et al. in [KMP12], this agile practice in their overviews.

**Energized work**

The agile practice *40-Hour Week* described in XP [Bec00] is given the same name by Kurapati et al. taken into account in [KMP12].
Williams also mentions the practice in [Wil10] but describes it as *Energized work*. Concerning Beck in [Bec05], it refers to the description of the agile practice in the XP rules. Furthermore, she refers to Tom DeMarco, who points out in [DeM02] that "Extended overtime is a productivity-reducing technique." In [Wil10], Williams mentions the practice of *Sustainable pace* as a synonym for *Energized work*.

**Collective code ownership**

Another agile practice described in XP [Bec00] is *Collective code ownership*. Williams refers in [Wil10] her description of the practice to Nordberg in [Nor03]. She relates to several other agile practices: *Pair programming*, *Coding standards*, *Continuous integration*. From Williams’ point of view, these practices can help address the challenges of *Collective code ownership*. According to Williams in [Wil10], the practices offer a “check and balance” for *Collective code ownership*.

In addition to Williams, Kurapati et al., this practice was included in its overview in [KMP12].

**Documentation**

Sidky describes in [Sid07] the practice as *Agile documentation*. He points out that documentation provides no value to the customer. Sidky concludes from this that the scope of documentation in agile software development should be "minimal." Jalali and Wohlin name in [JW12] their listing in the Praktik *Enough documentation*. The authors do not offer a detailed description. However, the name suggests that a sufficient amount of documentation must be taken into account. In addition to the authors mentioned, Kurapati et al., an agile practice regarding documentation, was added to their overview in [KMP12].

**Software configuration management**

Sidky lists *Software configuration management* in his list of agile practices in [Sid07, p. 98]. He justifies this with the fact that in an iteratively organized software development, the control of the different versions of the source code with the help of tools is crucial. In addition to Sidky, Kurapati et al. listed in agile practice in their overview in [KMP12]. The authors’ name this practice *Configuration and Change Management*.

**Knowledge-sharing**
Sidky describes knowledge sharing tools in [Sid07, p. 95]. He points out that creating a shared understanding leads to better collaboration. In [Sid07, p. 95], Sidky names the following examples for knowledge-sharing tools: "...blogs, wikis and forums...". Besides, in [Sid07, p. 95], he describes the possibility of analog tools such as "Whiteboards." Kurapati et al. name in [KMP12] another agile practice for imparting knowledge: Informative workshops.

Common knowledge, Diebold and Dahlem also list a practice in this context in their overview of agile practices in [DD14].

Planning Game

Several authors call the practice a Planning Game (cf. [JW10], [JW12], [KMP12]). Williams describes in [Wil10] planning poker as an agile practice. In Planning Poker, the entire team estimates based on a given scale. The practice aims to increase the accuracy of the estimate. This is made possible, for example, by the fact that all team members are taken into account in the estimation and, thus, the various experiences and competencies. If the estimated values of the individual participants differ, several rounds will be played. This practice is a Wideband Delphi Estimation [Wil10].

Several authors call the practice a Planning Game (cf. [JW10], [JW12], [KMP12]). Williams describes in [Wil10] Planning Poker as an agile practice. In Planning Poker, the entire team estimates based on a given scale. The practice aims to increase the accuracy of the estimate. This is made possible, for example, by the fact that all team members are taken into account in the estimation and, thus, the various experiences and competencies. If the estimated values of the individual participants differ, several rounds will be played. This practice is a Wideband Delphi Estimation [Wil10].

User Stories

Requirements are often documented in agile methodologies in form of User Stories (see [LDv+15], [LDv+16], [STE17], [Wil10], [WZW+14]). It is common for this form of requirements documentation to be used in backlogs (cf. [HW15], [SRP19], [Wil10]). Therefore, the practice of User Stories listed by Sidky in [Sid07, p. 87ff.] and by Jalali und Wohlin in [JW10], [JW12] (abbreviated to Stories by Williams in [Wil10] and Kurapati et al. in [KMP12]) taken into account in this assignment.
2. Comparison of the list of extracted practices with the agile practices of Scrum, XP, and Kanban

Based on the list created in step 1, a comparison is now carried out with the agile practices of the methodologies Scrum, XP, and Kanban described in the guidelines. This ensures that the agile practices of the three approaches described are taken into account in the listing for the present work. For the sake of clarity, only the added agile practices are described in Table 3 under consecutive numbering:

<table>
<thead>
<tr>
<th>#</th>
<th>Agile Practice</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>Definition of Done</td>
<td>Scrum, [SS17]</td>
</tr>
<tr>
<td>27</td>
<td>Simple design</td>
<td>XP, [Bec00]</td>
</tr>
</tbody>
</table>

Table 3: Considered practices of the agile methodologies Scrum, XP, and Kanban

In the second step of the extraction, two more agile practices were identified and added to the list. The practice of the **Definition of Done** (DoD) is described for Scrum (see [SS17]). The DoD defines which criteria must be met so that a requirement can be classified as completed. The Scrum Team determines the DoD.

**Simple design** is an agile practice from XP (see [Bec00]). According to Beck, in [Bec00], the software should be developed as simply as possible. This practice aims to reduce complexity. Beck describes in [Bec00] the correct design for software as follows:

1. “*Runs all the tests.*“
2. “*Has no duplicated logic. Be wary of hidden duplication like parallel class hierarchies.*“
3. “*States every intention important to the programmers.*“
4. “*Has the fewest possible classes and methods.*“

3. Comparison of the extracted practices with the lists of publications

In the first step of this process, not all agile practices were considered, as they had to be included in at least two lists. This criterion has been useful in order to be able to consolidate the practices. To finalize the list of agile practices for this work, in this third step, the practices mentioned only in an overview are checked. For this purpose, all practices that were not considered in the previous process were first identified. These agile practices are shown in the following table regarding the respective publication:
<table>
<thead>
<tr>
<th>Agile Practice</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>30% of Cockburn’s level 2 and Cockburn’s level 3 people</td>
<td>[Sid07, p. 87ff.]</td>
</tr>
<tr>
<td>Architecture focus</td>
<td>[JW12]</td>
</tr>
<tr>
<td>Code Ownership</td>
<td>[Wil10]</td>
</tr>
<tr>
<td>Feature Driven Development</td>
<td>[JW10]</td>
</tr>
<tr>
<td>Inspections</td>
<td>[Wil10]</td>
</tr>
<tr>
<td>Low process ceremony</td>
<td>[Sid07, p. 87ff.]</td>
</tr>
<tr>
<td>No big design up front (BDUF)</td>
<td>[Sid07, p. 87ff.]</td>
</tr>
<tr>
<td>No/Minimal number of Cockburn’s Level .-1 or 1b people on team</td>
<td>[Sid07, p. 87ff.]</td>
</tr>
<tr>
<td>Scrum Master</td>
<td>[JW12]</td>
</tr>
<tr>
<td>Scrum of Scrums</td>
<td>[JW10, JW12]</td>
</tr>
</tbody>
</table>

Table 4: Considered practices of the references

Ten agile practices have not yet been included in the lists of publications (see Table 4). A special feature is the listing of previously not considered practices Scrum of Scrums. Although the practice is contained in two overviews ([JW10], [JW12]), these are overviews by the same authors and publications that build on one another. So if agile practices are listed once in each of these two lists, they are still counted once. Therefore, this practice was not considered in step 1 of the extraction process.

In the following, it is checked separately for each agile practice, whether it is integrated into the newly created list of agile practices:

**30% of Cockburn’s level 2 and Cockburn’s level 3 people**

The practice from Sidky’s overview in [Sid07, p. 103f.], deals with the skills of software developers in agile teams. In [Sid07, p. 103f.], He refers to “Cockburn’s level of understanding” (described by Alistair Cockburn in [Coc01]). Sidky argues in [Sid07, p. 104] that software developers work in agile teams that “can deal with new and unexpected problems.”

The focus of this agile practice in Sidky’s list is on the selection of team members and the composition of agile software development teams. From the author’s point of view, the inclusion of this agile practice for the overview of the present work makes sense.

This is reasoned in particular by the fact that the experience and skills in the team have been identified as success factors for agile software development
(c.f. [CC08], [FIK+15], [GKS18], [MKK09], [SS12], [SSK20], [TMO+20]). It is, therefore, obvious to consider an agile practice that deals with the optimization of the composition based on experience and competencies in agile software development teams. It should also be mentioned that Sidky cites another agile practice in this context in his overview (see also No / minimal number of Cockburn’s Level -1 or 1b people on team).

**Architecture focus**

Jalali and Wohlin have listed this agile practice in their overview. The authors do not describe this practice in [JW12]. Since the designation of the practice is kept general, and due to the lack of an explanation, a thematic limitation is not possible, this practice is not taken into account.

**Code Ownership**

Williams describes *Code Ownership* in [Wil10] concerning [Nor03]. Williams defines this as the opposite of the agile practice *Collective Code Ownership* already considered. When applying for *Code Ownership*, a software developer (or a department) is responsible for (and managing) its source code.

According to Williams, in [Wil10], agile teams choose their “*Code ownership strategy*” themselves. From the author’s point of view, *Code Ownership* is not an agile practice. Instead, other authors describe the transition to *Collective Code Ownership* in the context of agile software development (cf. [MM04], [SRP16]). The practice is, therefore, not included in the overview of the present work as an agile practice.

**Feature Driven Development**

Feature Driven Development (FDD) is part of the overview by Jalali and Wohlin in [JW10]. However, FDD is much more of an agile methodology than a practice. The work by Palmer and Felsing in [PF02] applies as the set of rules of FDD. FDD was one of the models that were considered when creating the agile manifesto (cf. [BBv+20]). In addition to Boehm and Williams in [BW07], other authors consider FDD as an agile methodology (cf. [AGJ14], [CH11], [IQF15]). In the present work, FDD is regarded as a methodology and is, therefore, not included in the list of agile practices.
Inspection

Williams explains *Inspections* as an agile practice in [Wil10] regarding Fagan from [Fag76]. According to Williams in [Wil10], *Inspections* aim to improve the quality of “development artefacts.” In the opinion of Williams in [Wil10], these artifacts can be requirements documents or test plans in addition to the source code.

The entire team carries out *inspections*. Maruping et al. name in [MZV09] die the same objectives for the practices of *Inspections* and *Collective code ownership*: „Both collective ownership and code inspection bear the main objectives of finding and fixing coding errors by leveraging the expertise of the entire team.“ According to the authors, the team uses direct communication (*face-to-face communication*) for *Inspections*. For these reasons, inspection is integrated into the newly created overview as an agile practice.

Low process ceremony

Sidky lists the practice *Low process ceremony* in [Sid07, p. 109f.]. In [Sid07, p. 109f.], he describes the *Process ceremony* as “… the level of paperwork involved in a process.“. In his opinion, companies using agile software development require a low level of complexity in the organizational processes. He justifies this with the fact that otherwise, challenges arise in terms of reacting to changes.

From the author's point of view, organizational aspects must be taken into account in agile software development. The results of the SLR show that various organizational aspects influence the success of agile approaches. Examples of this are a favorable environment (workplace environment, learning environment) or support from management. For these reasons, the practice is included in the list.

No big design up front (BDUF)

*No big design up front* is an agile practice in the overview by Sidky in [Sid07, p. 99]. The author explains that an extensive design at the beginning of the project is not necessary for agile software development. Preferably, for example, through the self-organization of the team, the design activities are carried continuously out in iterations.
The practice is relevant for the present work, as it can be directly assigned to various agile characteristics of agile software development. In addition to the iterative approach, this also includes, for example, the ability to react to changes. An extensive design draft would oppose this insofar as it is not value-based. At least not if the customer doesn’t ask for it.

No/minimal number of Cockburn’s Level -1 or 1b people on team

As with the agile practice, \textit{30\% of Cockburn’s level 2 and Cockburns’ level 3 people} described above, this agile practice, described by Sidky in [Sid07, p. 112], focuses on the composition of the team based on the Software developer skills.

This practice is also included in the overview of the present work. The same reason is given for this as for the \textit{30\% of Cockburn’s level 2 and Cockburns’ level 3 people} practice described above.

Scrum Master

Jalali and Wohlin have considered the \textit{Scrum Master} as an agile practice in [JW12] berücksichtigt. The authors do not describe the practice in more detail in [JW12]. \textit{Scrum Master} is described as a role in the agile methodology Scrum (see [SS17]). From the author’s point of view, this is, therefore, not an agile practice. However, the roles of agile methodologies are taken into account in this overview, as they are defined above in chapter 2.1 as an Agile Element. Therefore, the role of the Scrum Master is included in the list of agile elements.

Scrum of Scrums

\textit{Scrum of Scrums} is included in the lists of agile practices by Jalali and Wohlin in [JW10], [JW12]. Sutherland, in [Sut05], first introduced the practice. Scrum of Scrums is described as a meeting. This is where representatives of several Scrum teams come together to exchange ideas with one another. Here, the status of the team is discussed, for example, about current work. The course of the meeting is based heavily on the Daily Scrum (c.f. [SS17]).

The practice is relevant to the present work, as it is used in global software development (cf. [PDL08a], [PDL08b], [PLH12], [SSR+08]). Furthermore, it is used for the exchange of knowledge between several Scrum teams in extensive or complex projects (cf. [QQ14], [Sut05]). For these reasons, the practice is included in the list.
For the sake of clarity, the added agile practices are shown under consecutive numbers in the following table:

<table>
<thead>
<tr>
<th>#</th>
<th>Agile Practice</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>Composition of the team based on Cockburn's level</td>
<td>[Sid07, p. 87ff.]</td>
</tr>
<tr>
<td>29</td>
<td>Inspections</td>
<td>[Wi10]</td>
</tr>
<tr>
<td>30</td>
<td>Low process ceremony</td>
<td>[Sid07, p. 87ff.]</td>
</tr>
<tr>
<td>31</td>
<td>No big design up front</td>
<td>[Sid07, p. 87ff.]</td>
</tr>
<tr>
<td>32</td>
<td>Scrum Master</td>
<td>[JW12]</td>
</tr>
<tr>
<td>33</td>
<td>Scrum of Scrums</td>
<td>[JW10], [JW12]</td>
</tr>
</tbody>
</table>

Table 5: Agile Practices have taken into account that is mentioned once in the references

The practices 30% of Cockburn's level 2 and Cockburns' level 3 people and No / minimal number of Cockburn's Level -1 or 1b people on team are summarized under an agile practice. This is known as the Composition of the team based on Cockburn's level.

In this third process step, five more agile practices were added to the new list.

4. Examination of the list of extracted agile practices in the context of software development practice

The overviews of agile practices referenced so far are academic. However, lists of agile practices are also published in the practical environment of software development. The Agile Alliance presents an overview in [Agi15] in the form of a glossary of agile elements. This glossary is used to compare the content of the list of agile elements to be created here.

The glossary contains 75 entries. Three of these entries are the agile methodologies Scrum, XP, and Kanban. Forty-three additional entries are included in the list of agile practices in this paper. These duplicates were not taken into account. Furthermore, two entries from the glossary do not correspond to the understanding of the agile practice. These are the entries Customer Development and the Version Control [Agi15].

The remaining 27 agile practices are taken into account for the list to be created here and presented in the following table:
Table 6: Integrated agile practices from the Agile Alliance glossary [Agi15]

A detailed description of the agile practices contained in the Agile Alliance glossary is not given here for the sake of clarity. The practices are described in detail in [Agi15].

5. Comparison of the list of extracted elements with the roles of Scrum, XP, and Kanban

As set out in Definition 1 in Chapter 2.1, agile elements include roles as well as practices. The agile elements extracted so far primarily include practices (58). Only two elements are roles.

Roles are usually described in the guidelines of agile methodologies. The roles of Scrum [SS17] and XP [Bec00] are, therefore, included in the overview of agile elements (see Table 7). No roles are defined in Kanban.
Nine more roles have been added to the list of agile elements. The Scrum Master role is already included in the list of agile elements. The Scrum Team role is also defined in Scrum. This summarizes the roles of Scrum Master, Product Owner, and Development Team. Therefore, the Scrum Team is not included in the list of agile elements.

The list of agile elements created contains 69 elements (see Table 8). Eleven of these elements are roles, the remaining 58 agile practices.

<table>
<thead>
<tr>
<th>#</th>
<th>Agile Element</th>
<th>Practice</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Continuous Integration and builds</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Daily Standup Meetings</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Pair programming</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Retrospective / Learning Loop</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Test Driven Development</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Review Meeting</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Using and maintaining a list of all features and their status (back-log)</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Planning Meeting</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Iteration based process</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Customer integration and collaboration (Interaction between customer and developer for effort estimation, scope, and timing of releases)</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Coding Standards</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Refactoring</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Tracking progress</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Communication</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Metaphor / Vision</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Office structure</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Empowered and self-organized team</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Small and frequent releases</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Energized Work</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Roles of the agile models’ Scrum and Kanban
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Collective code ownership</td>
</tr>
<tr>
<td>21</td>
<td>Documentation</td>
</tr>
<tr>
<td>22</td>
<td>Software configuration management</td>
</tr>
<tr>
<td>23</td>
<td>Knowledge sharing</td>
</tr>
<tr>
<td>24</td>
<td>Planning Game</td>
</tr>
<tr>
<td>25</td>
<td>User Stories</td>
</tr>
<tr>
<td>26</td>
<td>Definition of Done</td>
</tr>
<tr>
<td>27</td>
<td>Simple design</td>
</tr>
<tr>
<td>28</td>
<td>30% of Cockburn´s level 2 and Cockburns´ level 3 people</td>
</tr>
<tr>
<td>29</td>
<td>Inspections</td>
</tr>
<tr>
<td>30</td>
<td>Low process ceremony</td>
</tr>
<tr>
<td>31</td>
<td>No big design up front</td>
</tr>
<tr>
<td>32</td>
<td>Scrum Master</td>
</tr>
<tr>
<td>33</td>
<td>Scrum of Scrums</td>
</tr>
<tr>
<td>34</td>
<td>Antipattern</td>
</tr>
<tr>
<td>35</td>
<td>Business Agility</td>
</tr>
<tr>
<td>36</td>
<td>CRC Cards</td>
</tr>
<tr>
<td>37</td>
<td>Definition of Ready</td>
</tr>
<tr>
<td>38</td>
<td>Epic</td>
</tr>
<tr>
<td>39</td>
<td>Exploratory Testing</td>
</tr>
<tr>
<td>40</td>
<td>Facilitator</td>
</tr>
<tr>
<td>41</td>
<td>Given When Then</td>
</tr>
<tr>
<td>42</td>
<td>INVEST</td>
</tr>
<tr>
<td>43</td>
<td>Minimum Marketable Feature (MMF)</td>
</tr>
<tr>
<td>44</td>
<td>Minimum Viable Product (MVP)</td>
</tr>
<tr>
<td>45</td>
<td>Mob Programming</td>
</tr>
<tr>
<td>46</td>
<td>Mock Objects</td>
</tr>
<tr>
<td>47</td>
<td>Niko-niko-Calendar</td>
</tr>
<tr>
<td>48</td>
<td>Open Space</td>
</tr>
<tr>
<td>49</td>
<td>Personas</td>
</tr>
<tr>
<td>50</td>
<td>Project Chartering</td>
</tr>
<tr>
<td>51</td>
<td>Quick Design Sessions</td>
</tr>
<tr>
<td>52</td>
<td>Relative Estimation</td>
</tr>
<tr>
<td>53</td>
<td>Story Mapping</td>
</tr>
<tr>
<td>54</td>
<td>Story splitting</td>
</tr>
<tr>
<td>55</td>
<td>Three C´s</td>
</tr>
<tr>
<td>56</td>
<td>Three Amigos</td>
</tr>
<tr>
<td>57</td>
<td>Three questions</td>
</tr>
<tr>
<td>58</td>
<td>Ubiquitous Language</td>
</tr>
<tr>
<td>59</td>
<td>Usability Testing</td>
</tr>
<tr>
<td>60</td>
<td>Velocity</td>
</tr>
<tr>
<td>61</td>
<td>Product Owner</td>
</tr>
<tr>
<td>62</td>
<td>Development Team</td>
</tr>
<tr>
<td>63</td>
<td>Programmer</td>
</tr>
<tr>
<td>64</td>
<td>Customer</td>
</tr>
</tbody>
</table>
### 3.1.3. Summary of the creation of the list of agile elements

The basis for creating the list of agile elements are existing overviews of agile practices by other authors. For the first step of the creation, six existing lists from academic publications were used to compare the practices mentioned several times. Of the 166 agile practices initially mentioned from these publications, 25 have been included in the new list (shown in table 8 with a green background). Based on the 30 practices of the process models Scrum, XP, and Kanban described in the guidelines, a second comparison was then carried out. As part of this second step, two different agile practices (highlighted in blue in Table 8) were added to the list. In step 1, 14 practices were not considered. These were checked in the third step, and six of these practices were added to the new list (highlighted in orange in Table 8). The Agile Alliance glossary was used for the final content comparison of the practices. The glossary of agile practices has 70 entries. Of these, 27 were added to the list (shown with a yellow background in Table 8). Finally, the new list of agile elements was expanded to include the missing roles of the agile methodologies Scrum and XP.

The list of agile elements created consists of 69 elements. It was created and categorized into five steps. The list of agile elements serves as the first dimension of the theoretical model of cultural influences on agile elements for the present work. The list can also be used as an overview of agile practices. It brings together the knowledge of agile practices from various academic and practice-oriented publications.

### 3.2. The dimension of cultural levels

To enable the allocation of agile elements and cultural aspects and to be able to describe any cultural influence, a second dimension is generated for this model. Cultural levels structure this second dimension (see chapter 2.2). Based on these levels, cultural models are integrated into this second dimension. The cultural models make it possible to map certain cultural aspects and thus establish a concrete reference to agile elements.
3.2.1. Discussion of cultural levels

For this paper, the consideration of the respective cultural levels in agile software development is of central importance.

An exploratory literature search was carried out for this purpose. With the help of this literature research, we want to check which cultural levels are taken into account in the context of (agile) software development. We found that the levels of national culture and organizational culture, in particular, are relevant in the literature (see Table 9).

<table>
<thead>
<tr>
<th>Cultural level</th>
<th>Description</th>
<th>Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>National culture</td>
<td>Cultural influences on agile practices</td>
<td>[AVH17]</td>
</tr>
<tr>
<td></td>
<td>Collaboration in virtual teams</td>
<td>[GB17]</td>
</tr>
<tr>
<td></td>
<td>Influence on the areas of software engineering (e.g., Requirements Engineering)</td>
<td>[GBS11]</td>
</tr>
<tr>
<td></td>
<td>Influence on distributed software development</td>
<td>[HT07]</td>
</tr>
<tr>
<td></td>
<td>End-user involvement in enterprise systems</td>
<td>[Hwa12]</td>
</tr>
<tr>
<td></td>
<td>Scrum in the context of global software development</td>
<td>[LKT17]</td>
</tr>
<tr>
<td></td>
<td>Decision making</td>
<td>[MD07]</td>
</tr>
<tr>
<td></td>
<td>Influence on communication in distributed teams</td>
<td>[SSA+11]</td>
</tr>
<tr>
<td></td>
<td>Influence on teamwork/project success in global software development</td>
<td>[Su15]</td>
</tr>
<tr>
<td></td>
<td>Influence on the motivation of software engineering teams</td>
<td>[VBC+14]</td>
</tr>
<tr>
<td>Organizational culture</td>
<td>Usability in agile software development</td>
<td>[CSR+19]</td>
</tr>
<tr>
<td></td>
<td>Influence on the agile transition</td>
<td>[JZ16]</td>
</tr>
<tr>
<td></td>
<td>The success of lean/agile methodologies in large scale environments</td>
<td>[PFG13]</td>
</tr>
<tr>
<td></td>
<td>Influences on the process optimization when using agile and hybrid methodologies</td>
<td>[PT18]</td>
</tr>
<tr>
<td></td>
<td>Taking cultural influences into account when introducing, applying, and improving agile methodologies</td>
<td>[QH08]</td>
</tr>
<tr>
<td></td>
<td>Influences on global software development</td>
<td>[Gal09]</td>
</tr>
<tr>
<td></td>
<td>Test Governance Framework (Outsourcing IS development)</td>
<td>[DB15]</td>
</tr>
</tbody>
</table>

Table 9: Assignment of the primary studies to cultural influences per cultural level

The level of culture used in most of the primary studies is that of national culture. Furthermore, the cultural levels of the organizational culture are taken into account in the results of the SLR. Based on this knowledge, the culture models are discussed and selected in the following sub-chapters.

3.2.2. Discussion and selection of cultural models

This subchapter aims to select the cultural models for the present work. The selection is made based on a discussion of various models of the cultural levels of national and organizational culture.
3.2.2.1. Model of the cultural level national culture

The results of the literature research presented above show in the narrower context of agile software development and national culture, the focus on the model of cultural dimensions according to Hofstede\(^2\) (cf. [AVH17], [GBS11], [HG11], [Hwa12], [MD07]). Other models relevant in software engineering, such as international communication according to Hall and Hall\(^3\) (cf. [FB17], [MHK05b], [SNH+12]), the GLOBE study\(^4\) according to House et al. (cf. [FB17]), as well as the model of Corporate and work culture according to Trompenaars and Hampden-Turner\(^5\) (cf. [MHK05b], [SNH+12]), are not used in the primary studies identified in the literature research. For this reason, Hofstede’s model of cultural dimensions is discussed below. This discussion is the basis for weighing up whether the model is used for the present work.

The model of cultural dimensions according to Hofstede, also shows a high citation frequency, detached from certain specialist areas in the literature [Bas03]. It is also the most commonly used comparative cultural model (cf. [Jon07, p. 1] in [Ric14, p. 125]) and is discussed in the literature (cf. [Bas03], [IAS07], [McS02], [Ric14, p. 125ff.], [Wil02]).

Hofstede’s model is criticized in the literature based on various facets:

- **Reliability of data collection and data validity**
  
  Several authors controversially discuss the validity of the data. Richter notes in [Ric14, p. 127] concerning Huo and Randall in [HR91, p. 159] and Ng et al. in [IAS07, p. 176] that the limitation to one company (IBM) is to be classified as problematic. Richter summarizes this criticism in [Ric14, p. 126] as follows: *HOFSTEDE only examined employees of the IBM group and then stereotyped entire national populations based on his research results.* Soares et al. describe in [SFS07] the focus on a company as perhaps the most frequently cited point of criticism. About this criticism, Hofstede mentions in [Hof80] the advantages of only carrying out the study in one company. According to the author in [Hof80], distortions in behavior due to different management approaches or other company-specific differences are avoided.

  In [Ric14, p. 127], Richter also refers to McSweeney in [McS02] about data validity, who notes that the test subjects were selected selectively. For example, only middle and senior employees were surveyed. Hofstede and Hofstede themselves also point

---

\(^2\) The Cultural Dimensions of Hofstede are described in [Hof01].

\(^3\) Hall and Hall introduce their model of international communication in [HH90].

\(^4\) House et al. present their GLOBE Study in [HJH+02].

\(^5\) Trompenaars and Hampden-Turner introduce their cultural dimensions in [TH12].
out in [HH05, p. 27] that when the data was collected in South Africa, only white people were interviewed. Furthermore, McSweeney explains in [McS02, p. 96] that the samples per country are too small in certain cases. As an example, Richter cites the sample size for Singapore in [Ric14, p. 128], where Hofstede took 58 elements into account in the sample. Hofstede and Hofstede refer in [HH05] to the selection of the samples and the transferability of the findings from a company to a nation. The authors justify their approach in [HH05, p. 19] with the "general spirit of a nation" of Montesquieu. Hofstede and Hofstede specify in [HH05, p. 19] that the selection of test persons is not decisive as long as they are members of a national community. In [Ric14, p. 127], Richter critically examines this reaction by Hofstede and Hofstede, in which he states that IBM is a multinational corporation in which foreign employees can work in the respective country. Furthermore, Richter notes concerning Williamson in [Wil02, p. 1377] that it has not been examined whether the corporate culture of IBM has affected the workforce at the individual locations. Richter describes this fact in [Ric14, p. 128] as follows: "It is still unclear to what extent this influence of corporate culture affects the behavior and expectations of employees."

In [Ric14, p. 127], Richter also refers to McSweeney in [McS02], concerning data validity, who notes that the test subjects were selected selectively. For example, only middle and senior employees were surveyed. Hofstede and Hofstede themselves also point out in [HH05, p. 27] that when the data was collected in South Africa, only white people were interviewed. Furthermore, McSweeney explains in [McS02, p. 96], that the samples per country are too small in some instances. As an example, Richter cites the sample size for Singapore in [Ric14, p. 128], where Hofstede took 58 elements into account in the sample. Hofstede and Hofstede refer in [HH05] to the selection of the samples and the transferability of the findings from a company to a nation. The authors justify their approach in [HH05, p. 19] with the "general spirit of a nation" of Montesquieu. Hofstede and Hofstede specify in [HH05, p. 19] that the selection of test persons is not decisive as long as they are members of a national community. In [Ric14, p. 127], Richter critically examines this reaction by Hofstede and Hofstede, in which he states that IBM is a multinational corporation in which foreign employees can work in the respective country. Furthermore, Richter notes about Williamson in [Wil02, p. 1377] that it has not been examined whether the corporate culture of IBM has affected the workforce at the individual locations. Richter describes this fact in [Ric14, p. 128] as follows: It is still unclear to what extent this influence of corporate culture affects the behavior and expectations of employees.
Concerning the criticism regarding the data validity, Hofstede and other authors also responded. For example, Anderson et al. in [AAH11] that Bjerke and Al-Meer in [BA93] confirmed the data from Hofstede in the context of their data surveys using Hofstede's questionnaire for individual countries.

Regarding the criticism regarding the data validity, Hofstede and other authors also responded. For example, Anderson et al. in [AAH11] that Bjerke and Al-Meer in [BA93] confirmed the data from Hofstede in the context of their data surveys using Hofstede's questionnaire for individual countries.

- **Outdated data**
  McCoy et al. note in [MGK05] that the data was collected decades ago. In [MGK05], the authors also question whether the data can still be represented after the developments in recent years. Soares et al. describe in [SFS07] in this context that the data do not reflect the current situation concerning internationalization and globalization.

  Hofstede argues in [Hof02, p. 1357] that his key figures relate to fundamental values and that these are based on roots that are thousands of years old. In his opinion, the fundamental values can, therefore, only change very slowly, if at all. It should also be noted that although the initial data collection was a long time ago, more and more data was collected, and this confirms his data (cf. [HHM10, p. 44f.], [Hof01], [Hof02, p. 1359], [Hof20]). However, according to Richter in [Ric14, p. 129], other authors have demonstrated changes in Hofstede’s results (cf. [FCS+97, p. 52], [SCS01]).

- **Generalized interpretation of the model**
  Hofstede's model gives the impression of making a generalized interpretation possible at the national cultural level. This would mean that a nation's populations always have a homogeneous culture. Singh and Krishnan refer in [SK07] concerning Gupta in [Gup02] in this context to nations like India, which are of corresponding size and whose culture is nevertheless represented uniformly. Soares et al. point out in [SFS07] that Hofstede's model cannot be applied to all cultures. Richter writes in [Ric14, p. 130] regarding Lachmann et al. in [LNH94] and Williamson in [Will02, p. 1386] that it is questionable ... *whether a property that is understood as a fundamental* value *"in a certain nation really represents one in every nation."*
According to Richter in [Ric14, p. 131], the criticism of Hofstede's model is essential for a conceptual nature. Despite the criticism of Hofstede's cultural dimensions outlined above, reasons are given in the literature that speaks in favor of using the model. These are set out below:

- **The relevance in literature**
  Hofstede's model of cultural dimensions is highly relevant in the literature (cf. [Abr09], [AVH17], [FB17], [FCM03, p. 9], [GBS11], [HG11], [Hwa12], [Jon07, p. 1] in [Ric14, p. 125], [MD07], [MHK05b], [Wil02]). According to Anderson et al. in [AAH11], this can be traced back to Straub in [Str94], in particular in the context of information technology. The authors describe in [AAH11] Straub's decision to use Hofstede's model in [Str94] as “… set a standard for cultural studies in information systems research…“. Westwood and Everett describe in [WE87, p. 201] the high relevance of Hofstede's model as follows: „To ignore Hofstede's work is like ignoring the Indian Ocean: it looms too vast for that.”. The use of Hofstede's model, according to Taras et al., also in [TKS10] in studies that deal with intercultural contexts. The authors refer in particular to the publications by Kirkman et al. in [KLG06], Gelfand et al. in [GEA07], and Tsui et al. in [TNO07].

- **The empirical database**
  Several authors in the literature refer to the extensive database of the model (cf. [Bau13, p. 51f.], [ST08]). Richter writes on this in [Ric14, p. 125] that Hofstede's model was “… still, an approach that was almost revolutionary in its time and was based on a collection of data of unprecedented proportions”. Furthermore, the author refers in [Ric14, p. 132] regarding Williamson in [Wil02, p. 1389], “…that due to the many studies carried out separately and in different contexts, there is definitely empirical evidence that these models also are useful.“.

**Considerations for this paper**

The discussion in this chapter is the basis for the decision about the model to be used for the national cultural level. Models such as the GLOBE study, Trompenaars dimensions, or the Hall and Hall model are not used as often in software engineering, as explained above, as the culture dimensions according to Hofstede. Especially in the context of agile software development, Hofstede's model is most widely used in the literature. The relevance of the model in the field of this work is thus proven.

Another reason for using the Hofstede model is the extensive database. This enables validation of the theoretical model to be created here in a large number of countries.
3.2.2.2. Model of the cultural level organizational culture

In the studies identified in the literature search, only the iceberg model, according to Schein, is used in an adapted form in one study [DB15]. The iceberg model according to Schein (cf. [Gre19], [MHK05a], [SG99]), the Competing Values Model (CVM) (cf. [GGX19], [IH07], [SS07], [VNP15]) and the model of national basic values according to Hofstede (see chapter 3.2.2.1) are cultural models that are relevant in software engineering. First, the models adapted for the context of the organizational culture are introduced. These models are the CVM, the adapted iceberg model, according to Schein, and the adapted national basic values, according to Hofstede. Then the three cultural models are discussed. This discussion aims to select one of the three culture models for the present work.

**Competing Values Model (CVM) from Quinn und Rohrbaugh:**

The CVM, according to Quinn and Rohrbaugh, is used to describe organizational cultures. The CVM comprises two dimensions:

1. **Flexibility vs. stability**
   
   This dimension describes the structure of an organization concerning changeability.

2. **Internal focus vs. external focus**
   
   The difference in the focus enables whether the well-being of individual employees or the well-being of the organization as a whole is considered.

The combination of these two dimensions creates four quadrants. According to Denison and Spreitzer in [DS91], these four quadrants can be described as different sub-cultures in an organization: Group culture, hierarchical culture, development culture, and rational culture.

**Adapted iceberg model according to Schein:**

The iceberg model, according to Schein, has been adapted in the context of the organizational culture of Chiavenato in [Chi99] Chiavenato has retained the basic structure of the iceberg model. In [Chi99], he describes a formal, visible part of the organizational culture and cites, for example, the organizational structure of a company, job titles, and descriptions. He also names organizational requirements for this level about the technologies or processes to be used. As in Schein's model, this part of the iceberg is described as the part above the water surface. He also explains an informal and invisible part of the organizational culture. This lies below the surface of the water and, according to
Chiavenato, comprises values, group rules, and norms as well as informal patterns for interactions.

**C.H.I.D.D.I. according to Siakas und Siakas (based on Hofstede’s cultural dimensions):**

In [SS07], Siakas and Siakas present a model for describing the organizational culture. They refer to this model as “C.H.I.D.D.I. Typology of organizational cultures”. The basis for this model is the cultural dimensions of power distance and the avoidance of uncertainty from the model of Cultural dimensions, according to Hofstede (see Chapter 3.2.2.1). With the help of the two cultural dimensions, the authors create two dimensions in [SS07]. Based on the characteristics of the two cultural dimensions, they locate four quadrants.

These four quadrants represent the types of culture described by Siakas and Siakas in [SS07]: Clan (High PDI / Low UAI), hierarchical (High PDI / High UAI), democratic (Low PDI / Low UAI), and disciplined (Low PDI / High UAI)

**Considerations for this paper**

The iceberg model adapted from Chiavenato in [Chi99], according to Schein for the organizational culture, has a descriptive character. This is based, in particular, on the distinction between visible and invisible aspects. Although these aspects can be used to describe specific organizational cultures, there is no distinction between specific cultures.

The C.H.I.D.D.I. the typology of organizational cultures, according to Siakas and Siakas in [SS07], is based, as explained above, on two cultural dimensions, according to Hofstede (see Chapter 3.2.2.1). These cultural dimensions are power distance and uncertainty avoidance. On this basis, the authors explain four different types of culture and describe them in [SS07] as: “These four cognitive types of culture should be considered as ideal types.” The problem with this model is the fact that Hofstede introduces the model based on national culture (see Chapter 3.3.2). Siakas and Siakas describe the motivation to use the dimensions in [SS07] as follows: “By explaining complex entities using metaphors and simplifications, such as typologies and other kinds of simplified measurements and by using nationality as the basic unit, cross-cultural comparisons are facilitated.” Although four types of organizational culture are described in the model, it is doubtful whether the national culture is the appropriate basis for this. After all, Hofstede developed the cultural dimensions used to understand and compare national cultures.
The C.H.I.D.D.I. the typology of organizational cultures, according to Siakas and Siakas in [SS07], is based, as explained above, on two cultural dimensions, according to Hofstede (see Chapter 3.2.2.1). These cultural dimensions are power distance and uncertainty avoidance. On this basis, the authors explain four different types of culture and describe them in [SS07] as: „These four cognitive types of culture should be considered as ideal types.“. The problem with this model is the fact that Hofstede introduces the model based on national culture. Siakas and Siakas describe the motivation to use the dimensions in [SS07] as follows: „By explaining complex entities using metaphors and simplifications, such as typologies and other kinds of simplified measurements and by using nationality as the basic unit, cross-cultural comparisons are facilitated.“. Although four types of organizational culture are described in the model, it is doubtful whether the national culture is the appropriate basis for this. After all, Hofstede developed the cultural dimensions used to understand and compare national cultures.

As explained above, it is essential to take into account any existing subcultures in companies. Although the author shares the opinion of the authors Tolfo et al. in [TWF+11], Robbins in [Rob98], and Schein in [Sch09, p. 147ff.], that a dominant (organizational) culture can influence existing subcultures, it makes sense to choose a model that is based on organizational cultures describes specific sub-cultures. The CVM is such a model.

Another aspect is that the CVM is often used to describe the influences of the organizational culture in software engineering (see above and cf. [DA13], [NN03]). The CVM is also widely used outside the field of this work (cf. [DLN00], [Lam03], [PFP00], [YW09]). In addition, it has been validated in various studies (cf. [How98], [KW04], [OB12]).

For these two reasons, the CVM is used as a model for the cultural level of the organizational culture in this work.

### 3.3. Model of Cultural Impact on Agile Elements (MoCloAE)

In this chapter, the Model of Cultural Impact on Agile Elements (MoCloAE) is introduced. The model formally describes the influence of different cultural levels on agile elements. First, the modeling form of the causal and impact models is explained and the reasons why this modeling is suitable for the MoCloAE. Then the metamodel and the elements of the MoCloAE are introduced. Based on this, a hypothetical version of the MoCloAE is presented.
3.3.1. MoCIOAE metamodel

The MoCIOAE metamodel describes the cultural influence on agile elements (see Figure 1). For this purpose, classes and relationships are used to map these causal relationships between these two dimensions.

![Metamodel of the MoCIOAE](image)

Figure 1: Metamodel of the MoCIOAE

The individual components and their relationships are explained below:

**Classes:**

- *Agile elements* can become concrete in the type of practices or roles (see definition 1 in chapter 2.1).
- *Cultural levels* (see definition 2 in chapter 2.2) can be national cultures or organizational cultures. Culture models are used to concretize these cultural levels. For the national culture, the model of the national fundamental values, according to Hofstede, is used (see chapter 3.2.2.1). The Competing Values Model (CVM) is used as a culture model for the organizational culture (see chapter 3.2.2.2).
- *The impact* is defined as follows:

  **Definition 3 (Impact)** Impact is understood to mean influence on agile elements. The influence of a specific cultural level can have a positive or negative effect on agile elements.

**Relations:**
• The association impacts connect cultures with agile elements. It represents a direct 1:1 relationship between these two classes. The relationship describes the influence of precisely one culture on an agile element. As set out in definition 3, this influence can be positive or negative.

• The inheritance hierarchies of the classes Culture, Agile Elements, and Impact primarily serve to clarify the model. They also offer a content limitation, for example, in the context of the cultural levels.

The MoCIoAE metamodel describes, in general, the positive or negative influences of cultures on agile elements.

### 3.3.2. Example of the MoCIoAE

To illustrate this, Figure 2 shows an example of the MoCIoAE. The example of the MoCIoAE shows the influence of three different cultures on five agile elements in the form of directed arrows. The type of influence, whether it is positive or negative, is shown by a “+” or “-” on the respective arrow. The cultures are shaped in the form of a development and group culture according to the CVM as well as the cultural dimension of the power distance of the national fundamental values.

![Figure 2: Example of the MoCIoAE](image)

For example, it can be seen that a high power distance has a negative effect on a daily standup meeting. The description of the effect resulting from the respective influence is shown in a separate form. The MoCIoAE only represents the type of influence of culture on an agile element.

With the help of MoCIoAE, the results of future studies of cultural influences on agile elements can be formally mapped. The model can also be used as a basis for assumptions regarding cultural influences on agile elements.
4. Limitation of the study and future work

The central aspects of the MoCloAE are the created list of agile elements (see Chapter 3.1) and the cultural models discussed for this paper (see Chapter 3.2).

The list of agile elements is to be interpreted as a living artifact. It does not claim to be complete, nor is it to be regarded as final in any way. The past two decades have shown that new agile elements are continually being developed. In this respect, a possible expansion of the list of agile elements is taken into account at this point.

The cultural models selected for this paper based on the discussion are based on the decision only to consider the cultural levels of national and organizational culture. There is a possibility that other models are better suited for the development of this dimension of the MoCloAE when considering other cultural levels. Also, it must be taken into account that further cultural models will be developed and published in the future. For these reasons, the discussion of the cultural models for the respective research context should be restarted in case of doubt. Based on this, other cultural models can then also be selected accordingly.

The MoCloAE takes these aspects into account. It was deliberately designed in the form of a causal impact model. This form leaves sufficient leeway concerning the specific characteristics of the limitations described above. No fundamental adjustments to the MoCloAE are therefore necessary in these cases. Instead, it can be expanded with additional agile elements (such as artifacts) or cultural levels, for example.

It is also conceivable that other influences on agile elements can be integrated into the model. The focus of this paper is on cultural influences. For this reason, the integration of other types of influences has not been included.

Although the model is based on current findings in the literature, it should be noted that the MoCloAE has not yet been validated. This validation is planned for the future. In the form of a qualitative multiple case study, hypotheses based on the MoCloAE are to be tested.
5. Conclusion

A theoretical model was developed to be able to examine and understand the influences of different cultures (cultural levels) on agile elements. This model is intended to depict the connections between cultural influences and agile software development. The basis of the theoretical model is two dimensions: agile elements and cultures.

To be able to create the dimension of the agile elements, it is necessary to generate an overview of these elements (see chapter 3.1). There are three main reasons for this. There are overviews in the literature, but they differ in terms of content. Also, the research goals of the underlying work are partly different. Likewise, under identically labeled agile elements, sometimes different things are defined and understood. For the reasons mentioned, there is a high diversity in terms of the number and understanding of agile elements. Therefore, a new overview of agile elements was created when creating the theoretical model. This is based on the systematic extraction of agile elements (see Chapter 3.1.2). The basis for this extraction is five overviews from the literature, the frameworks of the three procedural models Scrum, XP, and Kanban, as well as the glossary of the Agile Alliance. The sources for the new overview of agile elements are compared to remove redundancies. The overview contains 69 agile elements.

Cultures describe the second dimension of the model. For this purpose, a delimitation based on culture levels must first be carried out (see Chapter 3.2). For the present work, the cultural levels of national and organizational culture are taken into account. As explained in Chapter 3.2, there are various models for describing cultures in the literature. Models are, therefore, being discussed for the two cultural levels mentioned above. For the cultural level of the national culture, the model of the national fundamental values, according to Hofstede, is used. The competing values model is used in the context of the organizational culture.

The theoretical model is called the Model of Cultural Impact on Agile Elements (MoCIOAE). The MoCIOAE is designed in the form of a causal impact model. Causal impact models are used in software engineering as well as in the more specific context of agile software development to be able to map and explain the influences of different things (see Chapter 3.3). The MoCIOAE is introduced as a metamodel in Chapter 3.3.1. A concrete, definitive version of the model is presented in section 3.3.2.
References


[KLG06] Kirkman, B. L.; Lowe, K. B.; Gibson, C. B.: A quarter century of Culture’s Consequences: a review of empirical research incorporating


