Integrating Agile Practices into Architectural Assumption Management: An Industrial Survey

Chen Yang[1,2,3], Peng Liang[1*], Paris Avgeriou[2]
[1] School of Computer Science, Wuhan University, 430072 Wuhan, China
[2] Department of Computing Science, University of Groningen, 9747 AG Groningen, The Netherlands
[3] IBO Technology (Shenzhen) Co., Ltd., 518057 Shenzhen, China
cyang@whu.edu.cn, liangp@whu.edu.cn, paris@cs.rug.nl

ABSTRACT

Although managing architectural assumptions can benefit software development in several aspects (e.g., reducing architectural misunderstanding and mismatch), the effort required is a key obstacle towards employing architectural assumption management in practice. One potential solution is to apply agile practices in order to reduce this effort. To this end, we conducted a survey with 91 practitioners to investigate the possibility of integrating agile practices into architectural assumption management in industrial practice. The results offer an overview of which agile practices can be integrated in architectural assumption management and how. Six agile practices were selected by more than half of the subjects: "Backlog", "Iterative and Incremental Development", "Refactoring", "Continuous Integration", "Effective Communication", and "Just Enough Work". Twelve agile practices were further elaborated by the subjects regarding how they can be used in architectural assumption management. Based on the survey results, we developed a classification of agile practices for agile architectural assumption management, which can act as a reference for researchers and practitioners to employ certain agile practices in architectural assumption management.

CCS CONCEPTS

• **Software and its engineering ~ Designing software**

KEYWORDS

Architectural Assumption, Agile Practice, Integration, Survey

ACM Reference Format:

Chen Yang, Peng Liang, and Paris Avgeriou. 2019. Integrating Agile Practices into Architectural Assumption Management: An Industrial Survey. In *EASE'19: 23rd International Conference on Evaluation and Assessment in Software Engineering 2019, April 15-17, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3319008.3319027

* Corresponding author

EASE '19, April 15–17, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-7145-2/19/04…$15.00
https://doi.org/10.1145/3319008.3319027

1 Introduction

According to Kruchten *et al.* [31], the concept of Architecture Knowledge "*consists of architecture design as well as the design decisions, assumptions, context, and other factors that together determine why a particular solution is the way it is*". Thus, Architectural Assumptions (AA[1]) are a type of architectural knowledge. To be more precise, AA are architectural knowledge taken for granted, or accepted as true without evidence [7]. This definition of AA emphasizes the characteristic of *uncertainty* in architectural knowledge: stakeholders believe but cannot prove, for instance, the importance, impact, or correctness of specific architectural knowledge. For example, an architect may assume that "*the concurrent users of the system will exceed 1 million*". This statement is an AA, because the architect is not sure about its correctness. This AA will exist until its uncertainty is eliminated.

In addition to being a type of architectural knowledge, assumptions are also a type of artifact. As defined by Kroll and Kruchten [32]: "*An artifact is a piece of information that is produced, modified, or used by a process.*" Since AA are produced (i.e., made), modified, and used during software development, we advocate treating AA as a type of software artifacts, similarly to requirements, design decisions, etc.

A multitude of problems in software development can be traced directly to not well-managed AA, for example, architectural misunderstanding and mismatch [9]. Though researchers and practitioners recognize the importance and necessity of managing AA in software development [8], the effort required is a key obstacle towards employing AA management in practice [7]. Traditional AA management is considered as resource-intensive, while the return on investment of AA management is debatable [7]. Therefore, there is a need to reduce the investment of managing AA and consequently making it less resource-intensive. One potential way to achieve this, is to integrate agile practices into AA management, as agility aims at reducing heavyweight effort in traditional software development, and embracing changes with a set of agile practices (see e.g., the Agile Manifesto [6]).

In our recent systematic mapping study on the combination of architecture and agility [2], we found that certain agile practices can be used in architecting. In this study, we used these identified twenty-five agile practices to explore the possibility of integrating them into AA management through a survey with 91 practitioners. Note that we do not study the opposite, i.e., how AA or their management can be used in agile development (practices).

The results provide an overview of which agile practices can be integrated in AA management and how. Based on those results, we further proposed a classification of agile practices for agile AA

[1] AA in this paper is used in singular (architectural assumption) as well as plural (architectural assumptions) based on the context.

management, which can act as a reference for researchers and practitioners to employ certain agile practices in AA management.

The rest of the paper is organized as follows: Section 2 provides related work. Section 3 details the survey design. Section 4 presents the results of the survey. Discussion based on the results and the threats to the validity of the study are provided in Section 5 and 6 respectively. Section 7 concludes the survey with future directions.

2 Related Work

We introduce assumptions and their management in the following three aspects.

Characteristics of assumptions. Every assumption has uncertainty. Uncertainty is an important criterion to judge whether a piece of information is an assumption.

The assumption concept is subjective, which is a major reason that stakeholders may have different understandings of the assumption concept [9]. For example, Lago and van Vliet [13] mentioned that it is difficult to draw the line between AA, requirements, and constraints.

Furthermore, assumptions are not independent in software development, but intertwined with certain types of software artifacts. For instance, when managing assumptions in software design (e.g., [18][19]), assumptions are usually related to requirements, design decisions, components, etc.

Though stakeholders can understand the assumption concept, they rarely use the term "*assumption*" in their work [7]. Instead, they mix assumptions with other types of software artifacts, such as requirements, design decisions, and risks [7].

Assumptions have a dynamic nature, i.e., they can evolve over time [14]. For example, during software development, a valid assumption can turn out to be invalid or vice versa, or an assumption can transform to another type of software artifact or vice versa.

Moreover, assumptions are context dependent [9]. For instance, the same assumption could be valid in one project, while invalid in another project because the context changes; or an assumption in one project is not an assumption in another project. The dynamic and context-dependent nature could be one reason that not-well managed assumptions lead to a multitude of problems in software development. As an example, the reuse of an assumption from ARIANE 4, which was not valid in ARIANE 5, led to the ARIANE 5 disaster [15].

Classifications of assumptions. There are various classifications of assumptions in software development from different perspectives. We provide two examples as follows. Garlan *et al*. [16] classified assumptions that can lead to architectural mismatch into four types: the nature of the components, the nature of the connectors, the global architectural structure, and the construction process (development environment and build). Lago and van Vliet [17] classified AA as technical, organizational, and management assumptions.

Assumption management. In our recent systematic mapping study on assumptions and their management [9], we identified twelve assumption management activities from the selected studies, namely Making, Description, Evaluation, Maintenance, Tracing, Monitoring, Communication, Reuse, Understanding, Recovery, Searching, and Organization.

Moreover, assumptions can be managed both manually and automatically. In the latter case, formal approaches and tools are used to automatically manage assumptions (e.g., assumptions are usually made and documented by such approaches or tools instead of stakeholders). A representative example of such management is assume-guarantee reasoning in system verification (e.g., [22][23][24][25]). Considering that there is a need to evaluate whether a system satisfies a property, instead of verifying the whole system, assume-guarantee reasoning decomposes the system into components (i.e., which are smaller and easier to verify), generates assumptions for a component that guarantees the property, checks whether the other components satisfy the assumptions, and then comes up with the results. On the other side, manual assumption management refers to manual work by stakeholders. For example, stakeholders use approaches and tools to make and document assumptions in their projects (e.g., [20][21][26][27]).

Agile development aims at stripping away, as much as possible, the effort-intensive activities in software development [28], and focuses on quick response to various changes of a project [28]. Agile practices are general practices used to support agile development. An example of such an agile practice is "Iterative and Incremental Development", which focuses on small releases and a planning strategy based on a release plan and an iteration plan [29]. We note that some of the agile practices can also be used in other development processes, such as the aforementioned "Iterative and Incremental Development". One may argue that those practices are general, instead of specific for agile development. In this work, similar to our systematic mapping study [2], we consider "agile practices" as the practices that can be used in agile development. Agile practices in principle adhere to the values proposed in the Agile Manifesto [6]. Further related work on agile practices in the context of software architecture is presented in [30].

Relation to our previous work on AA and their management. We clarify the relationship between this work and our previous work on AA and their management with their contributions and contribution types in Table 1.

Table 1. Relation to our previous work on AA and their management

Reference	Contribution	Contribution Type
[2]	State of the research regarding the architecture-agility combination in software development from literature.	State-of-research
[26]	A simplified conceptual model with a lightweight approach for AA Documentation in agile development, and preliminary evaluation of the approach.	Conceptual model
[33]	State of the practice regarding AA Identification and Documentation from practitioners' perception.	State-of-practice
[9]	State of the research regarding assumptions and their management in software development from literature.	State-of-research
[8]	State of the practice regarding architectural assumptions and their management from architects' perception.	State-of-practice
[7]	An Architectural Assumption Documentation Framework (i.e., an AA Documentation approach AADF), an Excel template, and industrial evaluation of the approach.	Documentation framework

[12]	An architectural assumption management process and evaluation of the process.	Management process
This work	A survey to investigate the possibility of integrating agile practices into architectural assumption management in industrial practice.	State-of-practice

Compared to the related work, our work has a different focus: it explores the possibility of introducing agile practices into AA management in industrial practice. To the best of our knowledge, there is no other similar work. However, the work presented in this section has provided a number of insights and inputs in our work. As an example, we developed the introduction page of our survey based on several aforementioned references on AA and their management.

3 Survey Design

3.1 Goal and Research Questions

According to the guidelines for selecting empirical methods in software engineering research [3], surveys aim to "*collect quantitative but subjective data (concerning individual's opinions, attitudes and preferences) and objective data such as demographic information for example a subject's age and educational level*". We did not aim at exploring what happens when practitioners use agile practices to manage architectural assumptions (in which situation a case study could be employed [3]), or studying correlation or causality of variables related to using agile practices in architectural assumption management (in which situation an experiment is more appropriate [3]).

We used the Goal-Question-Metric approach [1] to formulate the objective of this work: to **analyze** AA management **for the purpose of** exploration **with respect to** integrating agile practices into AA management **from the point of view of** software engineers **in the context of** software development in industry. According to this objective, we decided to conduct a survey.

The Research Questions (RQs) of this study are presented below.

RQ1: Which agile practices can be used in AA management?

There are a multitude of agile practices that can be used in the context of architecture. This RQ helps to investigate the feasibility of employing certain agile practices in AA management.

RQ2: How to use agile practices in AA management?

Even if it is feasible to use certain agile practices in AA management, there is a lack of guidance regarding how to use them. This RQ helps to fill this gap.

3.2 Survey Design

We employed an online self-administrated questionnaire [4] in this survey. The type of the survey is cross sectional [4], i.e., we asked the subjects to fill in an online questionnaire at a fixed point in time.

3.3 Population and Sample

Stakeholders may manage AA either explicitly or implicitly. As evidenced in literature (e.g., [7][8][9]), implicit AA management means: (1) stakeholders manage (e.g., making) AA in their work without being aware of them; and (2) they rarely use the AA term and concept, but treat assumptions as, for example, a force of other types of artifacts (e.g., design decisions) or other types of artifacts themselves (e.g., requirements per se).

We used Non-Probabilistic Sampling methods [4], i.e., convenience sampling and snowball sampling. The target population of this study consists of practitioners who (1) have experience in using agile practices; (2) explicitly or implicitly managed AA in their work; and (3) use Chinese as native language. The reason for selecting Chinese-speaking subjects is that Chinese respondents are easily accessible to two of the researchers of this survey. The population of software engineers in China is one of the biggest in the world, so it makes good sense to sample from this population.

We invited potential subjects from both our own social networks in industry and popular IT websites (e.g., GitHub[2] and CSDN[3]). Furthermore, according to the guidelines [4], since the potential subjects are easily accessible to the first two authors, we used both Convenience Sampling method (i.e., getting responses from people who are available or willing to participate) and Snowballing Sampling method (i.e., asking the subjects to invite people who are available or willing to participate from their contacts) in the survey to get a valid sample.

3.4 Data Collection

We used an online self-administrated questionnaire to collect data. The questionnaire comprises seven general questions (as shown in Table 4), five AA-related questions (as shown in Table 5), and three specific questions to the RQs (as shown in Table 6).

General questions focus on background information of the subjects. AA-related questions aim to validate whether the subjects have a fair understanding of the AA concept as well as managed AA in their work. This validation is necessary because practitioners may not explicitly use the term 'AA' and thus may confuse it with other concepts.

Specific questions collect answers to the RQs. In our systematic mapping study on the architecture-agility combination [2], we identified 41 agile practices that can be used with architecture. In this survey, we chose 25 out of the 41 agile practices, with each practice mentioned in at least two of the selected primary studies. The subjects might also come up with additional agile practices that are not covered in the 25 pre-provided agile practices. Furthermore, we asked the subjects to explain how their selected agile practices can be used in AA management.

A set of inclusion and exclusion criteria were used to select valid responses as shown in Table 2. We used Boolean "AND" to connect alternatives in the inclusion criteria and Boolean "OR" to connect alternatives in the exclusion criteria. The authors reached an agreement on the understanding of these criteria before selecting valid responses.

Table 2. Inclusion and exclusion criteria to select valid responses

Inclusion criteria
I1: The subject has experience in agile practices.
I2: The subject managed AA in her/his work.
Exclusion criteria
E1: If two or more responses have the same IP (Internet Protocol) address, only the first response is further evaluated.
E2: If two or more responses have the same Email address, only the first response is further evaluated.

[2] https://github.com/
[3] http://www.csdn.net/

E3: The subject does not provide an AA example.
E4: The subject does not understand the AA concept.
E5: There are inconsistencies (e.g., conflicts) between the answers to different questions in the questionnaire.
E6: The answers to the questions are meaningless, i.e., the answers to several or all the open questions in the questionnaire are gibberish or logically senseless.

3.5 Data Analysis

We used descriptive statistics for quantitative data analysis and Constant Comparison [5] for qualitative data analysis. Constant Comparison (the core of the Grounded Theory approach) is a systematic approach used for qualitative analysis, and a continuous process for verifying the generated concepts and categories [5]. In this survey, Constant Comparison was iteratively performed by the first and second author, and the codes and their relationships were refined in each iteration. The third author was consulted to resolve any inconsistency during the iterations.

3.6 Pilot Study

To increase the validity of the survey, we conducted a pilot study with five experts (including two native Chinese) within the domain of empirical software engineering, software architecture, and agile software development. The subjects were subsequently interviewed to obtain feedback and further improve the design of the survey. The data points from the pilot study were not included in the results of the survey.

4 Results

In this section, we present the results of this survey including the demographic results and the results of the two RQs, without our interpretation (this happens in Section 5).

4.1 Demographic Results

In total 351 responses were received of which 91 (25.9%) valid responses were included according to the criteria and further analyzed. The subjects are located in 26 cities in China: Beijing (26 out of 91, 28.6%) and Shenzhen (17 out of 91, 18.7%) are the most popular cities. As shown in Fig. 4, most of the subjects are involved in development (65 out of 91, 71.4%) and architecture design (59 out of 91, 64.8%). Three subjects provided additional answers regarding their tasks: process improvement, quality management, and consultant. Note that in this question, the subjects could select one or more options as their main tasks in their companies.

Regarding experience in IT industry and software architecture of the subjects, most of them have at least three years of working experience in IT industry (70 out of 91, 76.9%), but less than three years of working experience in software architecture (55 out of 91, 60.4%), as shown in Fig. 4. Furthermore, all the subjects (91 out of 91, 100%) claimed that they understood agile practices, and most of them (74 out of 91, 81.3%) used certain agile practices in at least one project.

We also asked the subjects regarding the size of their companies. Most of the subjects (65 out of 91, 71.4%) worked in a company where the number of employees was less than 500.

4.2 Results of RQ1

As shown in Fig. 1, in summary, six agile practices (out of 25, 24.0%) were selected by more than half of the subjects: "Backlog" (65 out of 91, 71.4%), "Iterative and Incremental Development" (60 out of 91, 65.9%), "Refactoring" (57 out of 91, 62.6%), "Continuous Integration" (49 out of 91, 53.8%), "Effective Communication" (49 out of 91, 53.8%), and "Just Enough Work" (47 out of 91, 51.6%).

Moreover, question SQ2 asked subjects to provide agile practices not listed in Table 7. However, most of the answers are more or less similar to the listed agile practices (although formulated differently).

4.3 Results of RQ2

In this RQ, we asked the subjects to detail how to use their selected agile practices (from RQ1) in AA management. We did not mandate that the subjects should elaborate on all agile practices; consequently the subjects only detailed one or several agile practices but not all. This was mostly because of time pressure (see Section 5.1 for a more detailed explanation).

As mentioned in Section 3.5, we used Constant Comparison to analyze their answers, and the results of the analysis are shown in Table 3.

Table 3. Ways to integrate agile practices in AA management

Agile practice	Description
Backlog	List AA in backlogs. For example, AA can be treated as a type of product backlog items. Stakeholders should be aware of the AA listed in backlogs. AA should be classified in backlogs. As an example, AA should be prioritized in backlogs.
Iterative and Incremental Development	In each iteration of software development, (a) investigate how existing AA can impact software architecture; (b) keep AA under control (e.g., avoiding AA explosion [9]); and (c) make plans regarding AA management. Important AA should be identified in early iterations of software development. Each iteration of AA management should have a clear endpoint. Though AA should be iteratively managed, there is a need to make clear which AA should be managed in each iteration of software development (or AA management). As an example, when the number of concurrent visitors is over a threshold, we may need to employ specific strategies (e.g., refactoring the architecture). However, it does not mean that we should manage an AA regarding concurrency from the very beginning of development. Such an AA can be managed later when necessary. Evaluate AA in short and fast iterations of software development (or AA management).
Refactoring	Refactoring should be an iterative process, and stakeholders can review the source code during each refactoring iteration to identify invalid AA.
Effective Communication	This includes effective communication of AA within a development team, as well as between the development team and customer team.

Retrospective	Retrospective can help stakeholders reflect on their own AA (e.g., validity), as well as be aware of the AA made by others. In retrospective, consider rationale of each AA made (e.g., why we made such an AA).
Face-to-Face Communication	Face-to-face communication is helpful for identifying implicit AA. AA are significantly impacted by customers. Face-to-face communication with customers is therefore of significant importance in AA management (especially in the early stages of development).
Test-Driven Development	AA should be tested in development (e.g., using test cases). Testing helps to identify implicit and invalid AA.
Prototyping	We may not always have on-site customers. Prototyping in the early phases of development helps to reduce invalid AA as well as transform AA to other types of software artifacts. Provide a prototype that includes explicit AA for customers to get their feedback.
Quick and Early Feedback	Get customers' feedback regarding potential problems of existing AA in short iterations and the early phases of software development.
Collaborative Software Development Process	Reach agreements regarding AA management (e.g., understanding and awareness of AA) within a team. Reach consistent understandings regarding AA with customers. This could reduce invalid AA that may further cause problems in projects.
User Story	Use user stories to describe AA. The described AA should be understandable. When using user stories to describe AA, it helps stakeholders to reach a consistent understanding of existing AA.
Pair Programming	Encourage discussions regarding AA and their management between pairs of programmers during pair programming.

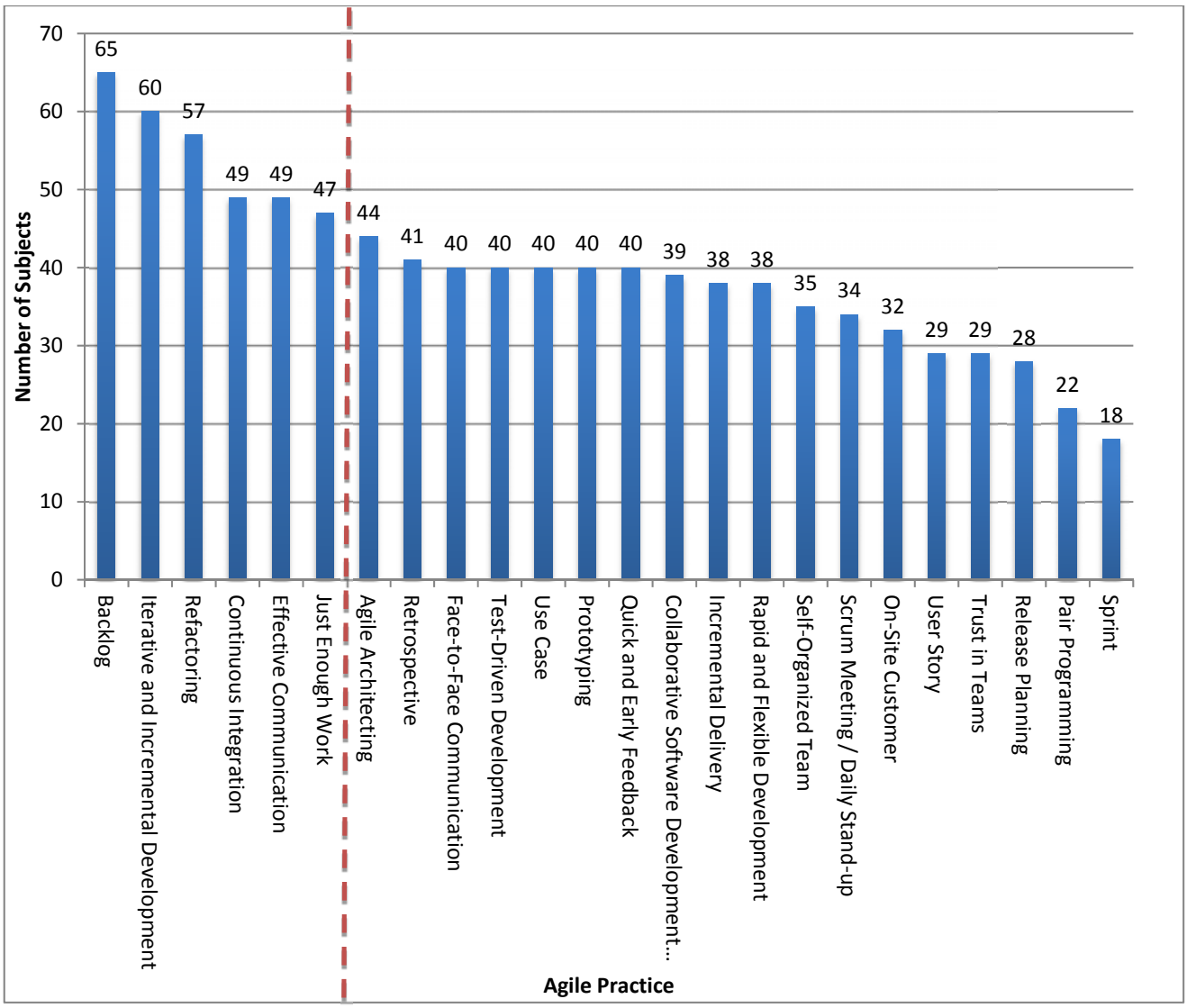


Fig. 1. Agile practices that can be integrated in AA management

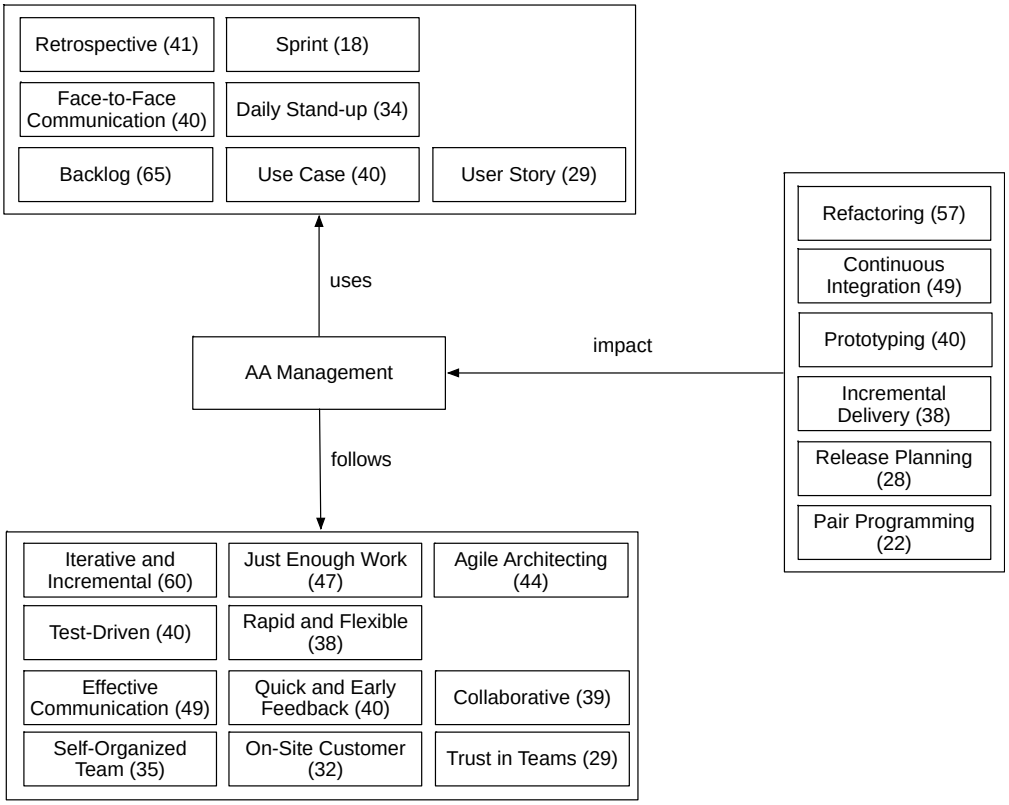


Fig. 2. A classification of the identified agile practices for AA management

5 Discussion

In this section, we first discuss the quality of the responses, and then present a classification of agile practices for AA management identified from the survey results.

5.1 Quality of the Responses

Compared with our previous industrial survey on AA (101 valid responses out of 213 responses, 47.4%), the proportion of valid responses in this survey is comparatively low (91 valid responses out of 351 responses, 25.9%). We consider two reasons: (1) Many responses did not meet the selection criteria. As an example, in the responses we found that many subjects answered "*No*" to questions "*Have you ever used agile practices in your work?*" and "*Have you ever managed AA in your work?*". (2) The subjects had probably little interest or insufficient knowledge in this topic. As an example, many subjects we invited were recruited from online resources (e.g., registered developers on GitHub); thus, we did not have enough information to judge whether a subject was qualified when sending the invitation.

Furthermore, we noticed that not every selected agile practice was elaborated by the subjects. This can be explained by two main reasons: (1) We used an online questionnaire, and many subjects were recruited using online resources. According to our experience, such types of subjects usually do not tend to spend much time on filling in a questionnaire, while open questions are the most time-consuming questions, compared with other types of questions such as multiple-choice questions. This may have led to some subjects not paying enough attention to the open questions. As an example, one subject merely mentioned: "*I think, for example, Continuous Integration is helpful for AA management.*" (2) To improve the validity of the survey, we developed an introduction page as well as certain questions to verify if the subjects are qualified for this study, including asking them to give an AA example they had managed in their work. Reading the introduction page and answering these questions already cost the subjects certain time, which might reduce their willingness to spend further time to answer the follow-up questions.

Finally, some subjects provided further lessons learned for AA management in general instead of specifically to agile AA management. As an example, one subject mentioned: "*The effectiveness of AA management depends on not only approaches and tools used, but also other contextual factors, such as whether the company considers design important and how much effort has been spent on design.*" Though the identified lessons learned could benefit AA management in general, we excluded them in this paper as they are out of the scope.

5.2 A Classification of Agile Practices for AA Management

We identified three types of agile practices for AA management from the answers of the subjects:

Agile practices that can be directly used in AA management activities. For example, stakeholders can include AA in the backlog or stakeholders can discuss AA and their management face-to-face.

Agile practices that have an impact on AA management. For instance, AA do not need to be refactored. However, when refactoring other types of software artifacts (e.g., software architecture), it helps stakeholders to identify invalid AA.

Agile practices that act as guidelines for AA management. As an example, the agile practice "Iterative and Incremental Development" can be treated as a guideline for AA management, i.e., AA should be iteratively and incrementally managed in software development.

We further classified the 25 pre-provided agile practices according to the three types as shown in Fig. 2. The numbers in Fig. 2 represent the popularity of each agile practice in agile AA management according to the subjects. This may help stakeholders not only to understand how to use certain agile practices in AA management, but also to pay more attention to the popular ones for potential use in their projects. As an example, if a project does not have on-site customers, it is natural not to use such a practice in AA management.

Below we provide an example of agile practices used in AA management to further elaborate the classification.

AA management is comprised of a set of activities [9]. A typical AA management process [12] is comprised of AA Making, Description, Evaluation, and Maintenance (see Fig. 3). These four activities with their inputs and outputs are elaborated below.

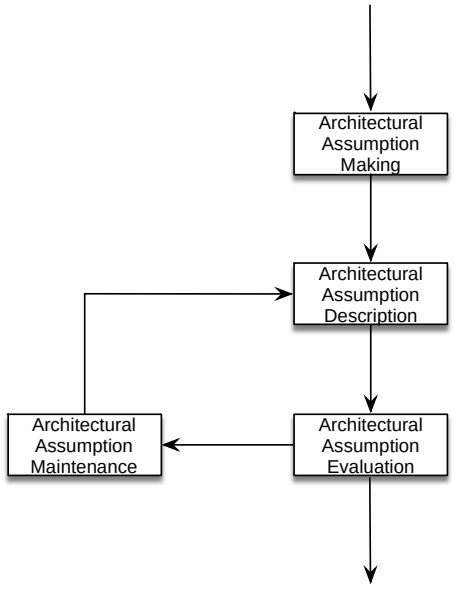


Fig. 3. A Typical AA management process

AA Making refers to both the actual making of assumptions in software development (including making new AA and identifying existing AA) and analyzing the properties of AA (e.g., validity, pros and cons). **Inputs**: Potentially all software artifacts, including project management materials (e.g., project description and business context), requirements (i.e., functional and non-functional requirements), and design (e.g., architectural design decisions). These artifacts can be either explicit (e.g., in documentation, source code comments, and bug repositories) or implicit (e.g., design decisions often remain in stakeholders' heads). **Outputs**: Undocumented AA (i.e., AA which have been made, but have not been documented).

AA Description aims at describing and recording AA in certain forms during software development. **Inputs**: Undocumented AA or maintained AA (see AA Maintenance below). **Outputs**: Documented AA. We note that the level of documentation can vary greatly depending on, for example, project context, from a one-sentence description to a full-fledged document for an AA.

AA Evaluation ensures that the description and analysis of AA are correct and accurate. **Inputs**: Undocumented AA or documented AA. **Outputs**: Evaluated AA (i.e., the results of evaluation, for example, which AA turn out to be invalid).

AA Maintenance adapts AA to fit the context of software development (e.g., eliminating outdated or invalid AA, modifying conflicting AA, and transforming AA to other types of software artifacts, such as requirements or decisions). **Inputs**: Evaluated AA. **Outputs**: Maintained AA (e.g., modifying invalid AA to fit their new context).

Regarding the first type of agile practices (i.e., agile practices that can be directly used in AA management activities), we provide the following two examples.

(1) In AA Description, stakeholders can create a backlog to list the AA that need to be evaluated. In AA Evaluation, stakeholders can evaluate AA based on the created backlog and maintain the backlog (e.g., removing evaluated AA) in AA Maintenance or Description.

(2) In AA Making, stakeholders can use daily stand-up meetings to identify existing AA and make new AA.

Regarding the third type of agile practices (i.e., agile practices that act as guidelines for AA management), we provide the following three examples.

(1) The architectural assumption process should be conducted iteratively and incrementally in most projects. For instance, when making AA using the AA management process, some AA may be identified as invalid in the first place, and such AA should not go to the next activity (i.e., "Just Enough Work" and "Agile Architecting"). This will ensure keeping the effort lower as documenting every AA is impractical (the cost would be much higher than the benefit). If such invalid AA in a latter phase of software development are identified as valid because of, for example, a new context, then they are considered as new AA made, and further proceed with the follow-up activities (e.g., Description).

(2) Since teamwork is rather important in AA management [9], teams in a project should be more collaborative with effective communications in those AA management activities. This can help to make AA explicit as well as identify and reduce invalid AA [9].

(3) The essence of AA is uncertainty. One major reason that stakeholders may have many AA in their projects is due to the lack of precise and correct information. If quick and early feedback can be employed in AA management, it would help especially in eliminating the uncertainty of AA, and evolving them to certain things.

The reason that we excluded the second type of agile practices (i.e., agile practices that have an impact on AA management) is that the second type is related not only to AA management, but also management of other types of artifacts. Though AA is not stand-alone, but intertwined with other types of software artifacts [9], integrating agile AA management in the software development lifecycle is not the focus of this work (see future work in Section 7).

6 Threats to Validity

The threats to the validity of this study are discussed based on the guidelines proposed by Kitchenham and Pfleeger [4].

Content validity is a subjective evaluation of the survey instrument [4]. To improve the content validity (e.g., whether the questions in the questionnaire can be understood clearly and correctly by the subjects), we invited one external reviewer to review the protocol of the survey, and conducted a focus group with seven researchers who are experts in empirical software engineering, software architecture, and agile software development to discuss the

design of the survey. Moreover, we also conducted a pilot study with five experts before the formal study to reduce this threat.

Criterion validity is to what extent the survey instrument can distinguish subjects belonging to different groups [4]. There is one threat that invalid responses could not be filtered out from valid responses. To mitigate this threat, we used two inclusion and six exclusion criteria to select valid responses.

Construct validity concerns to what extent the survey instrument measures the concept it is designed to measure [4]. One of the main concepts in the study is the AA concept, which is rather subjective [9]. Though practitioners make AA frequently in their daily work, and can understand the AA concept, they rarely use the AA concept and term in their projects [7]. A threat is that the subjects answered the questionnaire without a fine understanding of the AA concept. To mitigate the threat, we used five AA-related questions to select valid responses. Furthermore, we provided an introduction regarding AA and their management in the first page of the questionnaire.

Another threat is whether the researchers biased the subjects by providing the introduction page. This study aims to explore the possibility of integrating agile practices into AA management, instead of understanding the AA concept or how to manage AA in projects. The introduction page does not contain any information related to agile AA management. Furthermore, we explicitly mentioned in the introduction page: "*stakeholders may have different understandings of the AA concept, and we do not require the subjects to have the exactly same understanding of the AA concept as the researchers*". Finally, all the statements in the introduction page are supported by references. Therefore, we argue that the bias was reduced.

We also considered that the subjects might not truthfully answer the questions in the questionnaire. To minimize this threat, we explicitly mentioned that the participation of the survey is voluntary and anonymous.

Finally, since the subjects used Chinese as native language, there is a threat that the translation from English (i.e., the protocol) to Chinese (i.e., the questionnaire) and from Chinese (i.e., the collected data from the questionnaire) to English was not precise and correct. In our previous studies on AA, we also had to deal with similar issues regarding translation between Chinese and English, and therefore, we have certain practice and experience, which helps to reduce this threat in this study. To further mitigate this threat, the first and second author, who are native Chinese speakers, were responsible for the translation, and the third author reviewed the results (in English). We conducted the translation iteratively. Furthermore, the pilot study also helped to mitigate this threat.

External validity is the extent that the findings of the survey can be generalized. There is one threat that the subjects in the survey are not representative, i.e., the sample was biased. To mitigate this threat, we not only used our social contacts, but also invited subjects from popular IT websites to maximize randomness, i.e., including subjects with various backgrounds. We argue that the sample in this study is representative for at least practitioners who use Chinese as native language, have experience in agile practices, and managed AA in their work.

Reliability is the extent of measurement errors in the survey, and to what extent the replication of the survey can get the same results [4]. To improve the reliability of the study, the authors reached a consensus on the understanding of the protocol (e.g., the inclusion and exclusion criteria used to select valid responses) as well as the answers to the survey questions. The focus group and pilot study (see content validity) also helped to mitigate the threat to reliability.

7 Conclusions and Future Work

Through this survey, we gained insight on agile AA management. Although all the twenty-five agile practices identified from our recent mapping study on the architecture-agility combination [2] could be used in AA management, there is a significant difference regarding the popularity of the agile practices (e.g., "Backlog" vs. "Sprint"). Researchers and practitioners working in and employing AA management could pay more attention to those popular agile practices.

We also collected data regarding how to integrate a specific agile practice in AA management, and we developed a classification of agile practices for agile AA management. Such outputs act as a reference for researchers and practitioners to use agile practices in AA management.

For future work, we see a number of interesting directions. First, the data of the survey is based only on one region (i.e., China), thus replication of the study in other regions would be desirable. Second, those agile practices that can be used in AA management may have different levels of importance. Also different types of stakeholders may need to employ different agile practices in AA management. Thus, it would be interesting to devise a model connecting types of stakeholders, AA management, and agile practices. Third, although we developed a classification of agile practices for agile AA management according to the survey results, it needs further evaluation (e.g., usefulness). Fourth, the aim of integrating agile practices in AA management is to address the resource-intensive problem in traditional AA management. This work is a first step, i.e., investigating the possibility of integrating agile practices into AA management in industrial practice. A sound and operable approach for agile AA management is needed for practitioners to employ agile AA management in industry, as well as evaluation of the approach (e.g., whether the approach can address the resource-intensive problem in AA management).

ACKNOWLEDGEMENTS

This work is partially sponsored by the NSFC under Grant No. 61472286 and the Ubbo Emmius scholarship by the University of Groningen. We would like to thank Zhuang Xiong, Tianqing Liu, and Tianlu Wang who helped in disseminating the survey questionnaires, as well as all the participants who contributed their valuable knowledge and insights to this industrial survey.

REFERENCES

[1] V. Basili, G. Caldiera, and D. Rombach. The Goal Question Metric Approach, in Encyclopedia of Software Engineering, J.J. Marciniak, Editor, John Wiley & Sons: New York, NY. 1994.

[2] C. Yang, P. Liang, and P. Avgeriou. A systematic mapping study on the combination of software architecture and agile development. Journal of Systems and Software, 111(1): 157-184, 2016.

[3] F. Shull, J. Singer, and D.I.K. Sjøberg. Guide to Advanced Empirical Software Engineering. Springer, 2008.

[4] B.A. Kitchenham and S.L. Pfleeger. Personal Opinion Surveys. In: Guide to Advanced Empirical Software Engineering. Springer, pp. 63-92, 2008.

[5] B.G. Glaser and A. L. Strauss. The Discovery of Grounded Theory: Strategies for Qual-itative Research. New York: Aldine Publishing, 1967.

[6] K. Beck et al. Manifesto for Agile Software Development. http://www.agilemanifesto.org/, accessed on 2018-04-28.

[7] C. Yang, P. Liang, P. Avgeriou, U. Eliasson, R. Heldal, P. Pelliccione, and T. Bi, An industrial case study on an Architectural Assumption Documentation Framework. Journal of Systems and Software, 134(12): 190-210, 2017.

[8] C. Yang, P. Liang, P. Avgeriou, U. Eliasson, R. Heldal, and P. Pelliccione, Architectural assumptions and their management in industry - an exploratory study. In: Proceedings of the 11th European Conference on Software Architecture (ECSA), Canterbury, UK, pp. 191-207, 2017.

[9] C. Yang, P. Liang, and P. Avgeriou. Assumptions and their management in software development: A systematic mapping study. Information and Software Technology, 94(2): 82-110, 2018.

[10] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén. Experimentation in Software Engineering. Springer Science & Business Media, 2012.
[11] A. Fink. The Survey Handbook. Sage. 2003.
[12] C. Yang, P. Liang, and P. Avgeriou. Evaluation of a process for architectural assumption management in software development. Science of Computer Programming, 168(12): 38-70, 2018.
[13] P. Lago and H. van Vliet. Explicit assumptions enrich architectural models. In: Proceedings of the 27th International Conference on Software Engineering (ICSE), St Louis, Missouri, USA, pp. 206-214, 2005.
[14] G.A. Lewis, T. Mahatham, and L. Wrage. Assumptions Management in Software Development. Technical Report, CMU/SEI-2004-TN-021, 2004.
[15] ARIANE 5 Flight 501 Failure Report by the Inquiry Board. 1996. http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html
[16] D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch: Why reuse is still so hard. IEEE Software, 26(4): 66-69, 2009.
[17] P. Lago and H. van Vliet. Observations from the recovery of a software product family. In: Proceedings of the 3rd International Conference on Software Product Lines (SPLC), Boston, MA, USA, pp. 214-227, 2004.
[18] D.V. Landuyt, E. Truyen, and W. Joosen. Documenting early architectural assumptions in scenario-based requirements. In: Proceedings of the 2012 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), Helsinki, Finland, pp. 329-333, 2012.
[19] D.V. Landuyt and W. Joosen. Modularizing early architectural assumptions in scenario-based requirements. In: Proceedings of the 17th International Conference on Fundamental Approaches to Software Engineering (FASE), Grenoble, France, pp. 170-184, 2014.
[20] C.B. Haley, R.C. Laney, J.D. Moffett, and B. Nuseibeh. Using trust assumptions with security requirements. Requirements Engineering, 11(2): 138-151, 2006.
[21] R. Roeller, P. Lago, and H. van Vliet. Recovering architectural assumptions. Journal of Systems and Software, 79(4): 552-573, 2006.
[22] D. Giannakopoulou, C.S. Păsăreanu, and H. Barringer. Component verification with automatically generated assumptions. Automated Software Engineering, 12(3): 297-320, 2005.
[23] J.M. Cobleigh, G.S. Avrunin, and L.A. Clarke. Breaking up is hard to do: An evaluation of automated assume-guarantee reasoning. ACM Transactions on Software Engineering and Methodology, 17(2): Article No. 7, 2007.
[24] S. Chaki, E. Clarke, N. Sharygina, and N. Sinha. Verification of evolving software via component substitutability analysis. Formal Methods in System Design, 32(3): 235-266, 2008.
[25] C. de la Riva and J. Tuya. Automatic generation of assumptions for modular verification of software specifications. Journal of Systems and Software, 79(9): 1324-1340, 2005.
[26] C. Yang and P. Liang. Identifying and recording software architectural assumptions in agile development. In: Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE), Vancouver, Canada, pp. 308-313, 2014.
[27] A. Tang, Y. Jin, and J. Han. A rationale-based architecture model for design traceability and reasoning. Journal of Systems and Software, 80(6): 918-934, 2007.
[28] J. Erickson, K. Lyytinen, and K. Siau. Agile modeling, agile software development, and extreme programming: The state of research. Journal of Database Management, 16(4):88-100, 2005.
[29] S. Augustine. Managing Agile Projects. Prentice Hall: Upper Saddle River, NJ. 2005.
[30] Integrating Agile Practices into Architectural Assumption Management: An Industrial Survey: Complementary Material. https://tinyurl.com/y7pf7zpm
[31] P. Kruchten, P. Lago, and H. van Vliet. Building up and reasoning about architectural knowledge. In: Proceedings of the 2nd International Conference on Quality of Software Architectures (QoSA), Västerås, Sweden, pp. 43-58, 2006.
[32] P. Kroll and P. Kruchten. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. Addison-Wesley Professional, 2003.
[33] C. Yang, P. Liang, and P. Avgeriou. A survey on software architectural assumptions. Journal of Systems and Software, 113(3): 362-380, 2016.

Table 4. Questions on background information of subjects

ID	Question	Types of answers
GQ1	Which city are you working in?	City
GQ2	What are your main tasks in your company?	Multiple choice: Project Management / Requirements Engineering / Architecture Design / Detailed Design / Coding / Testing / Others
GQ3	What is your experience (in years) in IT industry?	Integer >= 0
GQ4	What is your experience (in years) in software architecting or software design?	Integer >= 0
GQ5	Have you ever used agile practices in your work?	No / Understand but Not Used in Projects / Used in One Project / Used in Multiple Projects
GQ6	What is the size of your company (number of employees)?	<100 / 100-500 / 501–1000 / > 1000
GQ7	(Optional) Your email address?	Text

Table 5. Questions on AA and their management

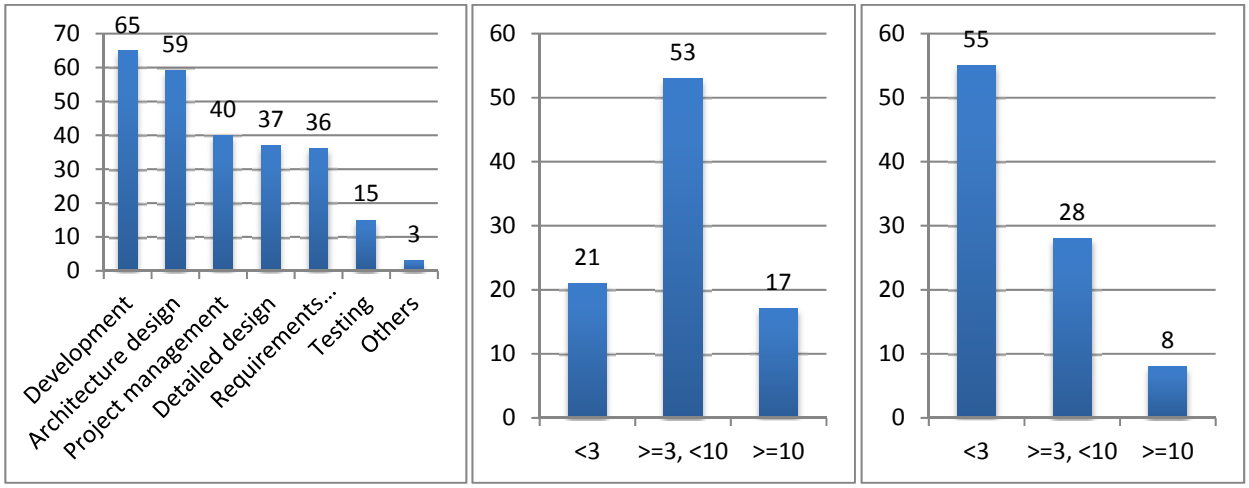
ID	Question	Types of answers
AQ1	To what extent do you agree with the examples, characteristics, and motivation of AA management provided in the Introduction page?	Agree / Partly Agree / Disagree
AQ2	(Optional) Would you please point out which part you disagree?	Free text
AQ3	After reading the Introduction page, to what extent can you state that you understand the AA concept?	Completely Understand / Understand / Moderate / Not Understand / Completely Not Understand
AQ4	Have you ever managed AA in your work?	Yes / No
AQ5	Would you please provide an example of AA you managed in your work?	Free text

Table 6. Questions on the RQs

ID	Question	Types of answers
RQ1. Which agile practices can be used in AA management?		
SQ1	Which agile practices in the following list can be used in AA management? (We added examples or explanations for several agile practices. However, in order to not bias participants, all the examples and explanations are not related to AA. For example, in "Just Enough Work", we added two examples regarding architecting and design. Note that the examples and explanations are only used to help participants clarify the agile practices. Please always consider and answer the questions in the context of AA management.)	Multiple choice: Agile practices listed in Table 7
SQ2	(Optional) Is there any other agile practice that in your opinion can be used in AA management?	Free text
RQ2. How to use agile practices in AA management?		
SQ3	How the agile practices you chose in SQ1 or described in SQ2 can be used in AA management?	Free text

Table 7. Agile practices used in the questionnaire

Backlog (e.g., product backlog)
Iterative and Incremental Development
Just Enough Work (e.g., just enough anticipation of architecting or just enough up-front design)
Sprint
Agile Architecting (i.e., making architecting agile, for example, using lightweight approaches in architecting, a little up-front architectural planning, or only the artifacts which directly contribute to the current iteration are created)
Continuous Integration
Scrum Meeting / Daily Stand-up
Incremental Delivery
User Story
Rapid and Flexible Development (e.g., rapid delivery of working products and embracing changes in development)
Refactoring
Retrospective (reflecting on how the team is performing and what can be done for improvements)
Face-to-Face Communication (encouraging communication between stakeholders through a face-to-face interaction)
Collaborative Software Development Process (focusing on collaboration between stakeholders, for example, developers and customers)
Test-Driven Development
Quick and Early Feedback
Use Case
Metaphor (a simple evocative description of how things work)
Release Planning (aims at committing to a plan for product delivery)
Effective Communication (e.g., between customers and developers)
On-Site Customer (customers being on-site with the development team during the whole lifecycle of the project)
Pair Programming
Trust in Teams (building trust within teams, for example, development teams)
Prototyping
Self-Organized Team (i.e., the members manage work themselves rather than being controlled by others outside the team, for example, leaders in their company)



(a) Main tasks of the subjects (b) Experience in IT industry (c) Experience in Software architecture

Fig. 4. Details of the subjects