University of Groningen

Evaluation of a process for architectural assumption management in software development

Yang, Chen; Liang, Peng; Avgeriou, Paris

Published in:
Science of computer programming

DOI:
[10.1016/j.scico.2018.08.002](https://doi.org/10.1016/j.scico.2018.08.002)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2018

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Yang, C., Liang, P., & Avgeriou, P. (2018). Evaluation of a process for architectural assumption management in software development. *Science of computer programming*, 168, 38-70.
<https://doi.org/10.1016/j.scico.2018.08.002>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Evaluation of a process for architectural assumption management in software development

Chen Yang^{a,b,c}, Peng Liang^{a,*}, Paris Avgeriou^b

^a State Key Lab of Software Engineering, School of Computer Science, Wuhan University, 430072 Wuhan, China

^b Department of Mathematics and Computing Science, University of Groningen, Nijenborgh 9, 9747 AG Groningen, the Netherlands

^c IBO Technology (Shenzhen) Co., Ltd., 518057 Shenzhen, China

ARTICLE INFO

Article history:

Received 27 October 2017

Received in revised form 9 August 2018

Accepted 21 August 2018

Available online 27 August 2018

Keywords:

Software architecture
Architectural assumption
Management process
Case study

ABSTRACT

Context: Architectural assumption management is critical to the success of software development projects. In this paper, we propose an Architectural Assumption Management (AAM) process, comprised of four AAM activities: Architectural Assumption Making, Description, Evaluation, and Maintenance.

Objective: Evaluating the AAM process in architectural assumption management, regarding the ease of understanding and the effort of conducting the AAM process, as well as the effectiveness of using the AAM process to make architectural assumptions explicit and to identify and reduce invalid architectural assumptions.

Method: An explanatory study with 88 first-year master students in software engineering, and an exploratory study with five practitioners from five companies.

Results: (1) the ease of understanding the AAM process is moderate for first-year master students but easy for practitioners; (2) the effort of conducting the AAM process is moderate for first-year master students; (3) Making and Evaluation took the students more time than Description and Maintenance; (4) the practitioners considered Evaluation as the most time consuming activity; (5) the AAM process can help to make architectural assumptions explicit and to identify and reduce invalid architectural assumptions in projects. The majority of the students and practitioners agreed that Architectural Assumption Evaluation is the most helpful activity for all these three aspects. For other activities, there are different opinions about their helpfulness; and (6) there are various factors identified that can impact the aforementioned results. Being aware of and properly adjusting these factors can facilitate the application of the AAM process in projects.

Conclusions: The AAM process aims at systematically managing architectural assumptions in software development. The results of the case studies provide preliminary empirical evidence for the evaluation of the AAM process.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

According to Kruchten et al. [46], 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”. In this paper, we

* Corresponding author.

E-mail address: liangp@whu.edu.cn (P. Liang).

focus on one of the types of architectural knowledge, namely architectural assumptions (AAs). An assumption is defined as “a thing that is accepted as true or as certain to happen, without proof”¹ or as “a fact or statement taken for granted”.² Accordingly we define AA as architectural knowledge taken for granted, or accepted as true without evidence [5,29]. The essence of this definition is that it entails uncertainty of architectural knowledge: stakeholders are not completely sure about, for instance, the correctness, impact, importance, suitability, or other aspects of architectural knowledge.

In addition to being a type of architectural knowledge, assumptions are also a type of artifact. As defined by Kroll and Kruchten [49]: “An artifact is a piece of information that is produced, modified, or used by a process”. Since AAs are produced (made), modified, and used during software development, we advocate treating AAs as a type of software artifact, similarly to requirements, design decisions, etc. The AA concept, its characteristics, and relation to other types of software artifacts are further elaborated in Section 2.

In this paper, we advocate that AAs need to be treated as first class entities in software development, i.e. they are clearly distinguished by other types such as design decisions or requirements. This is corroborated by other studies (e.g., [56,57]), but we provide some additional argumentation as follows:

- (1) AAs are important in both software architecting and the whole software development lifecycle [1,10]. AAs are not only related to requirements or design decisions, but also intertwined with other types of artifacts, such as use cases, components, design models, design patterns, and risks [4,10]; the lifecycle of AAs spans the whole software development lifecycle, instead of only the architecting or design phase [4,10].
- (2) Although AAs are related to different types of artifacts, they differ in nature. The management of other types of artifacts may not be suitable for AA management. We further discuss this in Sections 2.3 and 3.4.
- (3) Most importantly, AAs that are not well managed (e.g., remaining implicit or becoming invalid) could lead to severe issues in software development [4]. One such issue is **undesirable consequences**, including undesirable variations in product families, unpredictable and/or unacceptable program/system/environment behavior, and increased costs. For example, the reuse of an AA from ARIANE 4, which was invalid in ARIANE 5, led to the ARIANE 5 disaster [24]. Another issue is **increased system weaknesses**, including system vulnerability, system failures/errors, integration defects, and runtime errors. For example, ignoring or violating a system security assumption would lead to a security vulnerability [19]. Finally, another important issue is **design problems**, such as architectural mismatches and design violations. For example, inconsistent AAs regarding the context of components can lead to architectural mismatches between these components [20], and subsequently cause the lack of integrability.

Therefore, there is a need for systematic management of AAs in software development. Various AA management approaches have been proposed in literature (e.g., [1,3]), which can be used to support individual AA management activities (e.g., description of AAs). However, there is currently no work that explicitly proposes a general AA management process, comprised of a set of AA management activities.

In this work, we propose an Architectural Assumption Management (AAM) process, comprised of four AA management activities: AA Making, Description, Evaluation, and Maintenance. To validate the process, we conducted an explanatory study with 88 first-year master students in software engineering, and an exploratory study with five practitioners from five companies. Particularly we validated the ease of understanding and the effort of conducting the AAM process, as well as the effectiveness of using the AAM process to make AAs explicit and to identify and reduce invalid AAs.

The rest of the paper is structured as follows: Section 2 introduces the assumption concept in software development. Section 3 details the proposed AAM process. Sections 4 and 5 present the design and results of the student study. Section 6 provides the design and results of the industrial study. The results are further discussed in Section 7. Section 8 describes the threats to the validity of the two studies. Section 9 concludes this work with future directions.

2. Assumptions in software development

This section first introduces assumptions with related work on their classifications (Section 2.1), definitions (Section 2.1), and characteristics (Section 2.2), then discusses how to distinguish assumptions from other types of software artifacts (Section 2.3), and finally provides an example of Assumption Evolution in software development (Section 2.4).

2.1. Classifications and definitions of assumptions

There are a few classifications of assumptions in software development. Three examples are provided: (1) Garlan et al. [20] 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). (2) Lago and van Vliet [40] classified architectural assumptions as technical assumptions, organizational assumptions, and management assumptions. (3) Savolainen and Kuusela [41] classified domain assumptions as mandatory domain assumptions,

¹ Oxford English Dictionary. <http://www.oxforddictionaries.com/definition/english/assumption>.

² Merriam-Webster. <http://www.merriam-webster.com/dictionary/assumption>.

optional domain assumptions, and multiple domain assumptions. The AAM process proposed in this work is independent of classifications, i.e., it can work with any classification.

In addition, there are certain definitions of different types of assumptions. Three examples are provided: (1) we defined AAs in our earlier work [5,29] as architectural knowledge taken for granted or accepted as true without evidence. (2) Tang et al. [38] defined context assumptions as uncertainties and expectations of the context (e.g., motivational reasons) for making design decisions. (3) Haley and Nuseibeh [39] defined assumptions in requirements engineering as assertions of truth. One commonality between all these definitions is their focus on the uncertain nature of assumptions, i.e., every assumption inherently has uncertainty. As already mentioned in the introduction section, in this work we adopt the definition of AA from our previous work.

2.2. Characteristics of assumptions

In addition to the uncertain nature of assumptions, we summarized four main characteristics of assumptions in software development as the following.

Subjective. Many researchers and practitioners pointed out that whether a piece of information is an assumption or not, is rather subjective [1,4,5,29]. This is the major reason that stakeholders may have a different understanding of the assumption concept. As an example, Roeller et al. [1] mentioned: “*From one perspective or stakeholder, we may denote something as an assumption, while that same thing may be seen as a design decision from another perspective.*”

Intertwined with certain types of artifacts. Assumptions are not independent in software development, but intertwined with many types of software artifacts. For example, when managing assumptions in software design (e.g., [2,3]), assumptions are commonly related to requirements, design decisions, components, etc. This is further elaborated in Section 2.3.

Dynamic. Assumptions have a dynamic nature, i.e., they can evolve over time [5,27]. 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. This is further elaborated in Section 2.4.

Context-dependent. Assumptions are context-dependent [5,29]. For example, the same assumption could be valid in one project, and invalid in another project because the context changes; or an assumption in one project is not an assumption in another project.

2.3. Distinguishing assumptions from other types of artifacts

Due to the characteristics of assumptions, especially their subjective nature, many studies mention that it is difficult to draw a line between assumptions and other types of software artifacts. For example, Roeller et al. [1] discussed the difficulty of distinguishing architectural assumptions from requirements, design decisions, and constraints, as they believed that researchers and practitioners often employ such terms interchangeably in software development [1]. Therefore, the same piece of information could be an assumption in one context, and a completely different type of artifact in another context [5,29]. Unless the information is expressed in a specific way (e.g., using phrases like “*It is assumed that*”), it is difficult to judge whether the information is an assumption or not, without considering its context [5].

In this work, we advocate the necessity and importance to denote assumptions explicitly as such, during software development. However, this often does not happen in practice, and stakeholders need to judge whether a piece of information is an assumption or another type of artifact. To assist making this distinction, we further provide six lessons learned based on our previous work on assumptions and their management [4,5,10,29]. These may increase the possibility of classifying something correctly as an assumption but there are no definitive rules for doing so.

- (1) Although other types of artifacts may include uncertainty, uncertainty is not the core (or emphasis) of such artifacts. For example, when talking about a design decision, stakeholders usually do not debate whether it is uncertain, but focus on addressing the related problems. In contrast, assumptions are usually made in order to deal with uncertainties.
- (2) The content of an assumption and other types of artifacts is different. For example, the content of a decision is usually about employing a solution to a problem, while the content of an assumption is more about something is true, correct, important, suitable, etc.
- (3) Look at the context and not just the content. In our systematic mapping study [4], we identified many examples of assumptions claimed by the authors of the studies (e.g., “*The system will be available during normal working hours*” [47]; “*There is a subsystem that is responsible for receiving emergency calls and forwarding them to an available Coordinator*” [2]; “*If thread i holds the lock in read mode, then x cannot be changed by another thread*” [48]). We argue that it is difficult to treat such examples as assumptions by only reading these statements. However, if we dig deeper in the studies, paying attention to the context of those examples, it becomes clear why the authors of those studies considered the examples as assumptions, instead of other types of artifacts, such as requirements.
- (4) Every assumption should be characterized by uncertainty; if something is certain or has strong evidence supporting its validity, it is not an assumption. Despite this inherent uncertainty, an assumption is still taken for granted or accepted as true. Therefore, if such characteristic can be found in a piece of information, it increases the possibility that the information is an assumption. For example, an architect may assume that “*the concurrent users of the system will exceed 1 million*”. In this statement, the architect is not sure about its correctness, but takes it for granted during system

Fig. 1. An example of assumption evolution in software development.

design, which makes it an AA. This AA will exist until its uncertainty is eliminated (number of concurrent users indeed exceeds 1 million). Assumptions are not equal to uncertainties in software development: instead uncertainties can lead to assumptions, i.e., one way to deal with uncertainties is to make implicit or explicit assumptions [44].

- (5) Stakeholders should reach a consistent understanding on what is an assumption in their projects. Only paying attention to the properties of an assumption (e.g., validity and rationale) is not enough, stakeholders need to further consider or explain to the crowd why such a statement is an assumption, instead of another type of artifact.
- (6) Sometimes the line between an assumption and another type of artifact is vague. Instead of struggling with the “correct” answer of a statement (whether the statement is an assumption), stakeholders should focus on reaching an agreement on the statement within the team. This means that if stakeholders agree that a statement is an assumption, it can be managed as an assumption in the project. Later if the stakeholders find more evidence, they can make adjustments accordingly.

2.4. Evolution of assumptions

An assumption can evolve over time, for example, transforming to another type of software artifact [5]. We provide a real example from the petrochemical industry regarding assumption evolution, including transformation between assumptions, requirements, and design decisions (see Fig. 1). The system developed comprises ten subsystems, including a ports management (e.g., dispatching) subsystem, a futures management subsystem, and a customer relationship management subsystem.

In Iteration *i*, the project manager assumed that “The system would be deployed in a (secure-enough) internal environment” (Assumption: AA1). Subsequently, the architect assumed that “It might not be necessary to consider the external security (such as broken access control and cross-site scripting) of the system” (Assumption: AA2), because of AA1.

In Iteration *j*, the project manager considered “Deploying the system directly on Internet”, and AA1 and AA2 became invalid.

In Iteration *k*, AA1 was modified to “It is uncertain whether the system would be deployed directly on Internet”, and AA2 was modified to “External security of the system may need to be considered”.

In Iteration *l*, the customer confirmed AA1 and AA2 turning them into requirements: “The system would be deployed directly on the Internet” (Requirement: R1), and “The external security of the system should be considered” (Requirement: R2).

Table 1
Rationale of selecting AA management activities.

Activity	Rationale
Making	AA Making is the first step of AA management, and a prerequisite for the existence of AAs. This activity includes making both explicit and implicit AAs. In the latter case, stakeholders are not aware of the AAs made by others or even by themselves [29].
Description	Despite being an important type of architectural knowledge, AAs are usually kept undocumented in projects [10]; this is one of the most critical problems in software development, and leads to a multitude of issues (e.g., information erosion) [10]. Therefore, documenting AAs in an explicit way is of paramount significance, and should be a core activity, if not for all AA, at least for the most important ones.
Evaluation	AAs may be invalid in the first place (at the time they are made), because of, for example, the lack of knowledge or information when making these AAs [5]. However, this invalidity may be undetected. Moreover, AAs have a dynamic nature, i.e., they can evolve over time [29]. For example, during software development, a valid AA can turn out to be invalid or vice versa. Finally, the validity of an AA is context-dependent [29], for example, an AA can be valid in one project, while invalid in another project. As invalid AAs are rather harmful (e.g., leading to architectural mismatch) in projects [20], AA Evaluation can be used to identify such invalid AAs, and provides further guidance on how to deal with them.
Maintenance	The identification of invalid AAs motivates the need of AA Maintenance, since invalid AAs are rather harmful in projects, and adapting such AAs is a necessary step in software development. For example, when the context (e.g., project context) changes, an AA may need to be maintained in order to fit the new context.

In Iteration m , based on R2, the architect assumed that “Data validation and data encryption might be enough to support the external security of the system” (Assumption: AA3).

In Iteration n , AA3 was turned into a decision: “Data validation and data encryption will be used for the external security of the system” (Architectural Design Decision: ADD1).

3. Architectural assumption management process

Sections 3.1, 3.2, and 3.3 elaborate on the AAM process. Section 3.4 presents a comparison between AA management, requirements engineering, and architecting. Section 3.5 introduces how AA management can be integrated in architecting. Related work on AA management is provided in Section 3.6.

3.1. Architectural assumption management activities

To understand which activities are required for managing AAs in software development, we previously conducted a systematic mapping study on assumptions and their management in software development [4]. The mapping study analyzed 134 studies from an initial set of 19,428 studies. Twelve assumption management activities were identified and collected in the mapping study: Assumption Making, Description, Evaluation, Maintenance, Tracing, Monitoring, Communication, Reuse, Understanding, Recovery, Searching, and Organization.

While all the twelve activities are meaningful and useful in assumption management, we argue that including them all into a comprehensive process could be heavyweight and resource-intensive to be adopted in practice. For example, traceability between AAs and other types of software artifacts is beneficial in software development, but the effort required could be a critical problem, especially when dealing with a large number of artifacts. Therefore, to keep the overall process relatively simple and lightweight, we selected four assumption management activities as core activities to compose the AAM process: **Making**, **Description**, **Evaluation**, and **Maintenance**. Table 1 further explains why they were selected as the core activities.

We considered the remaining eight assumption management activities as supporting AA management activities; these can assist the core activities if enough project resources are available. There are two main reasons for making such a decision (i.e., meet at least one of the reasons). First, some assumption management activities (e.g., Communication) are general. These activities usually do not exist independently, but are meant to support the core activities (e.g., Making) during the AAM process. Second, these activities (e.g., Recovery) do not frequently take place in assumption management. As an example, in our systematic mapping study [4], we found that although twelve assumption management activities were explored, much effort had been put on Assumption Making (108 out of 134 selected studies, 80.6%), Description (89 out of 134, 66.4%), and Evaluation (83 out of 134, 61.9%). Assumption Maintenance (30 out of 134, 22.4%) received moderate attention. Other assumption management activities were mentioned in less than 15.0% of the selected studies. These supporting AA management activities, although not part of our proposed process, can still be used to support the core activities.

We note that this is one potential classification of core and supporting AA management activities. Other classifications can be used based on different criteria. It is of paramount importance to distinguish between a core part of the AAM process and a supporting (optional) part, in order to allow for a lightweight process, thus increasing the likelihood of its adoption in practice.

Fig. 2. AAM process in AA management activities.

3.2. Design of the architectural assumption management process

As evidenced in literature (e.g., [4,5,29]), AA management is teamwork: architects and designers are the major stakeholders in AA management, while other types of stakeholders should also be involved. Therefore, the main users of the AAM process are architects and designers, but other types of stakeholders can be included when performing specific activities of the process. For instance, during AA Evaluation, stakeholders such as project managers, requirements engineers, and customers can be involved. This is in line with other architectural approaches. As an example, when using ATAM to evaluate an architecture, various types of stakeholders (including customers) are suggested to be involved [50]. We define the core AA management activities and briefly describe the inputs and outputs of each activity below.

AA Making refers to both the actual making of assumptions in software development (including making new AAs and identifying existing AAs) and analyzing the properties of AAs (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 [28]). **Outputs:** Undocumented AAs (i.e., AAs which have been made, but have not been documented).

AA Description aims at describing and recording AAs in certain forms during software development. **Inputs:** Undocumented AAs or maintained AAs (see AA Maintenance below). **Outputs:** Documented AAs. 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 AAs are correct and accurate. **Inputs:** Undocumented AAs or documented AAs. **Outputs:** Evaluated AAs (i.e., the results of evaluation, for example, which AAs turn out to be invalid).

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

Fig. 2 shows the AAM process, comprised of the four core AA management activities. The process needs to be iteratively executed during software development instead of following a “sequential” flow; in Fig. 2 this is depicted through the feedback loop from AA Maintenance to Description. The flow of the AAM process proceeds as follows:

- (1) Stakeholders start with making AAs. Since it is difficult to draw a line between making new AAs and identifying existing AAs, we include both situations in AA Making. This activity can be conducted either individually or as a team. For example, in our previous study [29], we found that some practitioners usually use brainstorming to make AAs collaboratively as a team. After making or identifying AAs, stakeholders continue this activity through analyzing the properties of each AA.
- (2) After AA Making, the next step is AA Description. The aim of this activity is to document details of each AA made, including a basic description of what the AA is, rationale behind the AA, and related AAs or other types of artifacts. In

Fig. 3. Two adapted AAM processes.

our previous work, we proposed a framework for AA Description [5], which can be used as a reference template in this activity. Furthermore, AA Description can be conducted either individually or as a team.

- (3) The AAM process moves to AA Evaluation. The focus of this activity is to verify whether the AAs made and described are correct and precise, especially considering the validity of each AA. Normally AA Evaluation should be performed as a team, and includes various types of stakeholders (e.g., requirements engineers and architects).
- (4) There could be no problems identified from AA Evaluation, and therefore, no further steps would be taken for the evaluated AAs. Otherwise, the AAM process moves from Evaluation to Maintenance. The problems (e.g., an AA was evaluated as invalid or conflicted with a requirement) found in Evaluation are addressed in Maintenance (e.g., modifying invalid AAs to fit their new context).
- (5) As mentioned in literature (e.g., [30–32]), architectural documentation is usually not up-to-date, which leads to a multitude of issues, such as information erosion. Therefore, after AA Maintenance, there is a need to update the related documents, i.e., back to AA Description. Detailed guidelines of the AAM process can be found in Table 2.

The AAM process is flexible and customizable. For example, stakeholders can conduct AA Making and Description at the same time (see Fig. 3); after making AAs, stakeholders can skip AA Description (e.g., recording AAs in a document), and directly evaluate the AAs by discussing them face-to-face in a meeting (see Fig. 3); or they can extend the AAM process by adding supporting AA management activities.

Table 2 lists the guidelines of the AAM process. These guidelines are based on our previous studies on AAs and their management, including [4,5,10], and [29].

3.3. An example of using the architectural assumption management process

In this section, we walk through the usage of the AAM process using a simple but real industrial example from the domain of Internet of Things. Consider a system that manages a number of gas cylinders, trucks used to carry the gas cylinders, and drivers that drive the trucks. Functions of the system include identifying gas cylinders and tracking details of each gas cylinder (e.g., expiry date). There is a component in the system, which is used to identify gas cylinders.

In AA Making, the stakeholders made an AA that gas cylinders can be identified through three different ways: active tags, passive tags, and barcodes. Accordingly, they further came up with three AAs based on the AA made: “Active Tags Identification” (i.e., using active tags is a suitable solution for gas cylinders identification), “Passive Tags Identification” (i.e., using passive tags is a suitable solution for gas cylinders identification), and “Scanning Barcodes” (i.e., using barcodes is a suitable solution for gas cylinders identification), and analyzed the properties of each AA (e.g., rationale).

Next, the stakeholders documented the three AAs according to their discussion, using a MS Word template.

Moving to AA Evaluation, “Scanning Barcodes” and “Active Tags Identification” were identified as invalid, i.e., they were not accepted as true as the identification of gas cylinders with barcodes and active tags proved unsuitable. On the contrary, “Passive Tags Identification” was evaluated as valid.

Later in AA Maintenance, “Scanning Barcodes” and “Active Tags Identification” were decided to be removed. Then the stakeholders further updated the information in the related documents.

Table 2
Guidelines of the AAM process.

Activity	Guidelines
Making	<ol style="list-style-type: none"> (1) Before making AAs, stakeholders in a project should reach a consensus on (a) what AAs are, and (b) how to perform the AAM process. (2) AA Making is the first activity to be performed in the AAM process. (3) When Making AAs, some AAs may be identified as invalid in the first place, and such AAs will not go to the next activity. This ensures keeping the effort lower, since documenting every AA is impractical (the cost would be much larger than the benefit [5]). If such invalid AAs in a latter phase of software development are identified as valid because of, for example a new context, then they are considered as new AAs made, and further proceed with the follow-up activities (e.g., Description). (4) Making explicit AAs (as mentioned in Table 1) is recommended. (5) Conducting AA Making as teamwork is recommended.
Description	<ol style="list-style-type: none"> (1) Describing each valid AA made in AA Making is recommended. (2) As keeping AA Description up-to-date is rather effort-intensive, whether the description, as well as which part of the description should be kept up-to-date depends on project context (e.g., available resources). (3) Stakeholders should at least use a template for AA Description. (4) ID, Name, Description, State (i.e., valid and invalid), Rationale, Related AAs, and Related artifacts are the core elements in AA Description. (5) Each AA has a unique ID. (6) There should be at least two parts in the ID of an AA (e.g., AA1-1). When an AA changes (e.g., from valid to invalid), the second part of the ID should also change (e.g., from AA1-1 to AA1-2). (7) An AA in AA Description could be either valid or invalid. (8) Stakeholders can describe AAs either individually or as a team.
Evaluation	<ol style="list-style-type: none"> (1) Before evaluating AAs, a presentation of the overview of the project is recommended. (2) Evaluating each AA made in AA Making is recommended. (3) After evaluating an AA, there could be four types of results: (a) no problems are found regarding the AA; (b) the AA is problematic (e.g., invalid); (c) the uncertainty of the AA is eliminated (i.e., it is no longer an assumption and to be transformed, see Point (2) of Maintenance in this table); and (d) unable to evaluate the AA because of, for example, the lack of information. (4) The outputs of AA Evaluation should include the decision on whether an AA needs to be maintained. (5) Conducting AA Evaluation as teamwork is recommended.
Maintenance	<ol style="list-style-type: none"> (1) Maintaining problematic AAs is recommended (see Point (3)-(b) of Evaluation in this table). (2) If an AA transforms to another type of artifact, it includes two steps: (a) turning to be invalid, and (b) transforming to another type of artifact. (3) Stakeholders can maintain AAs either individually or as a team.
General	<ol style="list-style-type: none"> (1) The relationships between two AAs or an AA and another type of artifact can be zero, one, or multiple. (2) The relationships between two AAs or an AA and another type of artifact can be unidirectional or bi-directional. (3) AA management needs to be performed from the early phases of software development (e.g., requirements engineering or software design). (4) The lifecycle of AA management spans the whole software development lifecycle. (5) Using a dedicated approach (e.g., an Architectural Assumption Documentation Framework [5] based on ISO/IEC/IEEE Std 42010-2011 [51]) with tool support (e.g., Architectural Assumption Manager [58]: a tool that implements the Architectural Assumption Documentation Framework) to manage AAs is recommended. Such approaches and tools can help to, for example, reduce manual effort, keep documentation up-to-date, ensure the output quality of the AAM activities, or handle the intertwined relationships between AAs and other types of software artifacts. (6) Turning an AA from valid to invalid is not equal to removing the AA, i.e., such invalid AA can still be retained in the system design.

3.4. Architectural assumption management vs. requirements engineering and architecting

AAs are related to different types of artifacts, including requirements and architectural design decisions. However, AAs require their own management process because of the following reasons.

- (1) As discussed in Section 2.3, AAs, requirements, and architectural design decisions are different artifacts with different characteristics, properties, etc. As an example, the essence of an assumption is uncertainty, which is not the emphasis in a requirement or a decision.
- (2) The focus (or concerns) of managing different types of artifacts is diverse, leading to different approaches and tools. These differences include both the tasks that compose the approach or tool, as well as what artifacts are created and consumed and what stakeholders do in a specific task. As an example, Evaluation in requirements engineering, architecting, and AA management has completely different meanings. In requirements engineering, Evaluation means “to go back to the system stakeholders and check if the requirements are what they really need.” [53] In architecting, Evaluation refers to “ensuring that the architectural design decisions made are the right ones, and the candidate architectural solutions proposed in Architectural Synthesis are measured against the architecturally significant requirements collected in Architectural Analysis.” [33] However, in AA management, Evaluation “ensures that the description and analysis of AAs are correct and accurate.” [4] It is possible to integrate AA management into for example architecting, but AAs will always require their own management approaches and tools.

Fig. 4. A general architecting process.

Fig. 5. An extended architecting process.

In summary, we argue that though there are existing processes for certain types of software artifacts (e.g., in requirements engineering and architecting), there is still a need to design a general process for managing AAs in software development.

3.5. Integrating architectural assumption management in architecting

An architecting process is comprised of a set of activities. Hofmeister et al. [33] proposed a general model (as shown in Fig. 4), including three activities that form the architecting process as follows.

Architectural Analysis aims at defining the problems an architecture must solve. The outcome of this activity is a set of architecturally significant requirements.

Architectural Synthesis proposes candidate architecture solutions to address the architecturally significant requirements collected in Architectural Analysis, thus this activity moves from the problem to the solution space.

Architectural Evaluation ensures that the architectural design decisions made are the right ones, and the candidate architectural solutions proposed in Architectural Synthesis are measured against the architecturally significant requirements collected in Architectural Analysis.

Tang et al. [34] extended the architecting process proposed by Hofmeister et al. [33] through adding another two activities (as shown in Fig. 5).

Architectural Implementation realizes the architecture by creating a detailed design.

Architectural Maintenance aims to change an architecture for the purpose of fixing faults.

The aforementioned five activities are core architecting activities (i.e., covering the entire architecting lifecycle), while there is a set of supporting architecting activities (i.e., used to support the core activities) identified in literature (e.g., [54, 55]).

Architectural Recovery is used to extract the current architecture of a system from the system's implementation.

Architectural Description is used to describe the architecture with a set of architectural elements.

Architectural Understanding is used to comprehend the architectural elements and their relationships in an architecture design.

Architectural Impact Analysis is used to identify the architectural elements, which are affected by a change scenario.

Table 3

Employing AA management in the architecting process.

	Architectural Analysis	Architectural Synthesis	Architectural Evaluation	Architectural Implementation	Architectural Maintenance
Making	++	++	+	+	+
Description	++	++	+	+	++
Evaluation	+	+	++	+	+
Maintenance	+	+	+	+	++

Architectural Reuse aims at reusing existing architectural design elements, such as architecture frameworks, decisions, and patterns, in the architecture of a new system.

Architectural Refactoring aims at improving the architectural structure of a system without changing its external behavior.

AA management is not a standalone process, but should be connected with existing software development activities (e.g., architecting). Since AA is architectural in nature, Table 3 shows an example regarding the integration of the AAM process within the general architecting process [33,34]. “++” means that the related AAM activity should be conducted along with the respective architecting activity, and “+” denotes that the related AAM activity could be optional.

AA Making and Description should be conducted together with Architectural Analysis and Synthesis. For example, when identifying architectural significant requirements or making architectural design decisions, stakeholders can come up with a set of AAs that the requirements and decisions are based on.

Architectural Evaluation ensures the correctness of the decisions made in Architectural Synthesis; thus AAs can be assessed within Architectural Evaluation. As an example, consider using ATAM [50] to evaluate an architecture: besides sensitivity points, tradeoff points, and risks, another output of ATAM can be validity of AAs.

All the AAM activities (e.g., making AAs) are optional in Architectural Implementation. As an example, stakeholders can still make new AAs or revisit existing AAs when implementing the architecture.

Finally, when maintaining the architecture, stakeholders can decide which AAs should be adapted as well as solutions for maintenance (i.e., AA Maintenance), and then update the related documents (i.e., back to AA Description).

As mentioned in Table 2, a dedicated approach (e.g., a documentation framework) is recommended for managing AAs. For example, we proposed the Architectural Assumption Documentation Framework in our previous work [5]. The framework provides a template to document details (e.g., pros and cons) of each AA, supports stakeholders to get an overview of relationships between AAs, traces AAs to other types of software artifacts (e.g., requirements and design decisions), and monitors the lifecycle of AA in software development.

3.6. Related work on architectural assumption management

According to the results of our previous work [4], all the identified assumption management approaches are related to individual assumption management activities (e.g., Description). However, there is no work that explicitly proposes a general assumption management process, comprised of a set of assumption management activities. In this section, we discuss three types of representative assumption management approaches. More details about related work on assumptions and their management can be found in [4].

Assume-guarantee reasoning is designed specifically for system verification. Considering there is a need to verify whether a system satisfies a property, instead of verifying the entire system, assume-guarantee reasoning decomposes the system into components (i.e., which are smaller and easier to verify), generates an assumption for a component that guarantees the property, checks whether the other components satisfy the assumption, and then comes up with the results of the verification. There are different variations of assume-guarantee reasoning. For example, Nam et al. [25] mentioned the following process of assume-guarantee reasoning: decomposing a system S into components $M1, M2, \dots, Mn$; identifying an assumption A_i ($i = 1, 2, \dots, n$) for each component; verifying whether the assumptions are appropriate for proving the satisfaction of the property by system S .

Assumption Description framework and language. There are several frameworks designed for Assumption Description. For example, Van Landuyt and Joosen [3] proposed a meta-model and an instantiation strategy based on quality attribute scenarios and use cases to document AA. Another way to document assumptions in software development is to use specific languages. For example, Seater et al. [26] used Alloy (a modeling language) to express assumptions.

Assumption management system. Assumption management systems usually support one or several assumption management activities. However, such systems are not accompanied by an explicit assumption management process, i.e., they do not specify the rationale, flows, inputs, and outputs of the process. For example, Lewis et al. [27] proposed an assumption management system in software development with the focus on managing assumptions from source code. Their assumption management system is related to six assumption management activities: Assumption Making, Description, Evaluation, Maintenance, Searching, and Organization. Nevertheless, the assumption management process employed behind the assumption management system is implicit.

4. Case study

We followed the guidelines proposed by Runeson and Höst [6] to design and report on this case study.

4.1. Goal and research questions

The goal of the case study is to **analyze** the AAM process **for the purpose** of evaluation **with respect to** ease of understanding and effort of conducting the AAM process, effectiveness of using the AAM process to make AAs explicit, identify and reduce invalid AAs **from the point of view of** junior software engineers **in the context** of requirements engineering and software architecture design. The five aspects are detailed below:

Ease of understanding the AAM process. This aspect includes both the ease of understanding individual AA management activities and the entire AAM process. If the AAM process is not easy to understand and instead requires a steep learning curve, practitioners may hesitate to adopt it.

Effort of conducting the AAM process. This aspect includes both the effort of conducting individual AA management activities and the entire AAM process. Conducting the AAM process in software development requires certain effort. However, stakeholders usually need to balance the effort spent on different tasks (e.g., requirements engineering and architecting tasks), and the resources for AA management are usually limited. It would be a problem if stakeholders have to spend much effort on using the AAM process.

Effectiveness of using the AAM process to make AAs explicit. AAs are usually implicit in projects; in fact stakeholders may make AAs without even being aware of them. This may cause many problems (e.g., architectural misunderstanding), thus making AAs explicit is of paramount importance in software development. In this case study, making AAs explicit means that the relevant stakeholders in a group (i.e., those that need to know, such as the development team) are aware of the AAs made.

Effectiveness of using the AAM process to identify invalid AAs. Invalid AAs may negatively impact software development, as evidenced in our systematic mapping study [4], in which we identified eight types of problems caused by invalid assumptions, such as system vulnerabilities and system failures. In this case study, we aimed to evaluate whether the AAM process can help to identify invalid AAs.

Effectiveness of using the AAM process to reduce invalid AAs. Since invalid AAs may negatively impact software development, it is necessary to reduce invalid AAs in projects. However, when an AA is identified as invalid, it may not be addressed immediately. Therefore, we considered identifying and reducing invalid AAs as two different aspects. In this case study, we aimed to evaluate whether the AAM process can help to reduce invalid AAs.

It may be obvious that the AAM process can help to make AAs explicit, and identify and reduce invalid AAs, but we still need evidence. Otherwise it is only an argument of the authors. Moreover, the last three aspects include a deeper analysis, for example, identifying the most helpful activity for making AAs explicit.

The research questions (RQs) of this study, which are formulated based on the aforementioned five aspects, are presented below.

RQ1: How easy is it to understand the AAM process?

RQ2: What is the effort required for conducting the AAM process?

RQ3: Does the AAM process effectively help to make AAs explicit?

RQ4: Does the AAM process effectively help to identify invalid AAs?

RQ5: Does the AAM process effectively help to reduce invalid AAs?

Our data collection procedure (e.g., how to measure the effectiveness) is further presented in Section 4.3.

4.2. Case and subject selection

As mentioned in Section 2.2, the AA concept is rather subjective. This means that AA management (e.g., AA Making) is to some extent subjective in software development. Therefore, it is difficult to conduct a comparative study in which certain groups use the AAM process, while the other groups use an ad-hoc approach in AA management. Specifically, we argue that it is unrealistic to compare quantitative results among different groups in the context of the course project (i.e., each group had a different project) as well as the RQs defined in this work. For example, metrics such as the number of invalid AAs identified by each group cannot be used for indicating the effectiveness of using the AAM process to identify invalid AAs. Instead, we focused on presenting and analyzing both quantitative and qualitative data (e.g., using Constant Comparison for the analysis of qualitative data, see Section 4.4). Furthermore, this case study is explanatory [6], because it focuses on evaluating the AAM process in AA management based on the five aspects mentioned in Section 4.1.

4.2.1. Case and subjects description

A case is a contemporary phenomenon in its real-life context [7]. The borderline between the types of studies is not always distinct, as the term “case study” is used for various kinds of studies [22]. In this work, one case (i.e., a course project) was conducted at Wuhan University in China with 22 groups of first-year master students in software engineering (4 students per group, 88 students in total), who took the requirements engineering and software architecture course given by the second author during the autumn semester from 20/09/2016 to 29/11/2016. All students were Chinese speakers (the

Fig. 6. An example of role playing.

related threat is discussed in Section 8). We considered this study as a preliminary evaluation of the AAM process, which is necessary and beneficial for further evaluation (e.g., a longitudinal study) [12].

We used a questionnaire to collect background information of the subjects (see Section 4.3). Most of the subjects (74 out of 88, 84.1%) had been involved in at least one academic software engineering project (e.g., course and bachelor projects). 33 subjects (out of 88, 37.5%) had participated in at least one industrial project (i.e., working in a team for industry to develop software-intensive systems). Furthermore, we also asked the subjects if they had worked in open source software community. 6 subjects (out of 88, 6.8%) stated that they had such experience.

The majority of the subjects (65 out of 88, 73.9%) had taken software engineering courses. 18 subjects (out of 88, 20.5%) attended software development competitions (e.g., working in a team and developing a software system in the domain of Finance³), and 3 subjects (out of 88, 3.4%) acquired the certification of Software Designer (in Chinese).⁴ Furthermore, we also asked the subjects whether they had ever received any professional training (i.e., excluding the courses at universities) related to software engineering. 7 subjects (out of 88, 8.0%) stated that they had such experience.

To summarize, the vast majority of the subjects (80 out of 88, 90.9%) were educated in software engineering at the undergraduate level and had a certain level of project experience. Only 8 subjects (out of 88, 9.1%) had never participated in any software project (i.e., academic, industry, or open source projects), and had never attended any course of software engineering. To ensure that these 8 students would not affect the results, we grouped the students according to their student numbers, and allocated these 8 subjects to different groups.

In this study, each group was asked to develop a project document (including a requirements engineering, architecture design, and AA management document) as the final deliverable of the course project; some of these student projects continued at a business incubation center funded by Wuhan University after finishing the course. The course project taken by each group was proposed by another group in terms of role-playing. For example, as shown in Fig. 6, Group C proposes Project 1, and Group A makes a bid for this project. Thus for Project 1, Group C acts as the customer team, while Group A plays the role of the development team. There were 22 different projects in this case study. Since the focus of this study is AA management, background information of each project (e.g., introduction of each project) is not included in this paper. Furthermore, the deliverables of each project were checked by the first and second author every two weeks (i.e., four iterations in total) to help the subjects reduce misunderstandings regarding the templates provided by the researchers (see Table 15 in Appendix A) and to give each group a grade of their reports (i.e., four grades per group in total). In order to not bias the subjects, we had introduced all the details of the AAM process before the case study (i.e., the course project), and we did not suggest anything to the subjects regarding how to apply the AAM process during the case study. We focused on grading as well as improving readability of their reports. For example, some groups put two versions of the same AA in two sections of the report; we asked them to put different versions of an AA into the same section. Moreover, during the study, we did not change anything of the AAM process design. The subjects did not know their grades until they finished their projects.

4.2.2. Treating students as subjects

The research community has long debated on the scientific value of treating students as subjects in experiments, case studies, etc. [13]. Sjøberg et al. [18] conducted a survey on the state of how software engineering controlled experiments were conducted and reported (103 out of 5453 papers selected from 12 leading software engineering venues). There were 113 controlled experiments and 5488 subjects from the selected papers, and 87% of the subjects were students. In the domain of software engineering, there could be a significant difference in experience between practitioners and students.

³ http://www.citigroup.com/china/csts/EducationProgram/AboutEducation_CFCcn.html.

⁴ <http://www.ruankao.org.cn/>.

However, as mentioned by Host et al. [9], for some tasks (e.g., in the context of project impact assessment and risk management), software engineering students can be considered equal to practitioners. As an example, in the context of the Personal Software Process (i.e., a structured process for personalized software development), Runeson [14] found that there was a difference between freshmen students and graduate students, but there were no conclusive results on the differences between graduate students and practitioners. Berander [15] also pointed out that for some tasks (e.g., identifying trends and behaviors), students could be suitable subjects. For example, the author found that the requirements prioritizations made by students and practitioners could be similar under certain conditions. Galster et al. [17] conducted three studies with graduate students in software architecture courses. Their findings indicate that there are some challenges when treating students as subjects. However, graduate students were able to perform architecting activities, and empirical studies with graduate students could be considered as useful and valid research approaches.

Among the studies that treated students as subjects, there are two studies similar to our study, i.e., including architecting tasks. Van Heesch et al. [11] conducted a comparative multiple case study with four teams of senior software engineering students who took the software factory course. They mentioned that involving students in empirical studies could lead to issues (e.g., generalizability), and they used the guidelines proposed by Carver et al. [12] to alleviate the related risks. Tofan et al. [16] conducted a case study with 7 graduate students and 13 practitioners, and an experiment with 113 graduate and undergraduate students who took the software architecture course. They also discussed the problems of treating students as subjects, and they used the same checklist proposed by Carver et al. [12]. Similarly, in this work, we conducted a case study with 88 first-year graduate students in software engineering, who took the requirements engineering and software architecture course. We also applied the checklist from Carver et al. [12] to ensure that the case study contributed to the students' education and the students were committed to the study. The checklist can be found in Appendix D.

4.2.3. Case study procedure

The procedure of the case study comprises three steps with five tasks.

Step 1: Preparation

T1 Each group proposed a software project, and made a bid for a project proposed by another group. The proposed projects were reviewed by the first and second author based on their industrial and teaching experience, to make sure that the projects were sufficiently different from each other and non-trivial.

T2 The researchers prepared the case study design and documents of the tutorial, and collected background information from each subject through a questionnaire (see Section 4.3).

T3 The researchers gave a tutorial to help the subjects understand the concept of AA, the AA management activities, and the AAM process; had a discussion with the subjects regarding these three topics after the tutorial to ensure that the subjects had a sufficient basis to conduct the case study. The researchers also elaborated the tasks that the subjects needed to perform and the schedule of the case study.

Step 2: Execution

T4 The subjects used the AAM process to manage AAs in their projects. Since there were no dedicated tools for the AAM process, we asked the subjects to use MS Word. We also provided the subjects a template for their AAM reports (see Table 15 in Appendix A). Except for the template, we did not offer them any other approach for conducting individual AA management activities. Every two weeks the groups were asked to submit a new version of their AAM reports.

Step 3: Data collection

T5 In the last week of the course, each subject was asked to fill in a questionnaire (see Section 4.3) and attend a focus group, which were used to answer the RQs. As mentioned by Kontio et al. [23], the typical number of subjects in a focus group is between 3 and 12, and therefore, we organized eleven focus groups (i.e., two project groups of the subjects per focus group).

Table 4 shows relationships among the tasks, who were involved, inputs and outputs of each task, and the related RQs.

4.3. Data collection procedure

There are certain techniques to collect data in empirical studies [52]. As an example, Shull et al. [52] classified the techniques into three types: direct, indirect, and independent techniques. Considering the context of this study, we decided to use three data collection methods: questionnaire, focus group, and documentation analysis to answer the RQs. Interview was not included because it would be extremely resource-intensive (88 subjects), which could not be achieved in a realistic time frame. Collecting data from the three data sources to answer the RQs helped to improve the validity of the case study through data source triangulation [22].

Questionnaire: We asked each subject in the preparation phase to fill in Questionnaire A (see Table 16 in Appendix B and Table 18 in Appendix C), which was used to collect background information of the subjects. We also asked each subject in the last week of the course to fill in Questionnaire B (see Table 17 in Appendix B and Table 20 in Appendix C), which is structured according to the RQs.

Focus group: The focus groups aimed at encouraging the subjects to discuss further the AAM process in AA management. The researchers organized the discussion according to the RQs.

Table 4

Relationships among the tasks, who were involved, inputs and outputs of each task, and the related RQs.

Part	Task	Who	Inputs	Outputs	RQs
Part 1	T1	Researchers and subjects	(1) Documents of project proposal	(1) Documents of project selection	N/A
	T2	Researchers and subjects	(1) Documents of the AAM process (e.g., the AAM process design and a questionnaire for collecting background information)	(1) Case study design (2) Tutorial materials of the AAM process (3) Results of background information	N/A
	T3	Researchers and subjects	(1) Case study design (2) Tutorial materials of the AAM process	(1) Questions and answers regarding the course	N/A
Part 2	T4	Researchers and subjects	(1) Tutorial materials of the AAM process (2) Documents of project selection	(1) AAM reports (2) Questions and answers regarding the course	RQ3, RQ4, RQ5
Part 3	T5	Researchers and subjects	(1) Questionnaires and focus groups for the RQs	(1) Results of the questionnaires (2) Results of the focus groups	RQ1, RQ2, RQ3, RQ4, RQ5

Table 5

Relationships among the data collection methods, data analysis methods, and RQs.

Data collection method	Data analysis method	RQs
Questionnaire	Descriptive statistics and Constant Comparison	Background information, RQ1, RQ2, RQ3, RQ4, RQ5
Focus group	Constant Comparison	RQ1, RQ2, RQ3, RQ4, RQ5
Documentation analysis	Descriptive statistics and Constant Comparison	RQ2, RQ4

Documentation analysis: The AAM report is comprised of five parts (see Table 15 in Appendix A and Table 19 in Appendix C). To quantitatively measure the effort of conducting individual AA management activities in the case study, the subjects were asked to explicitly record the time (per subject) they spent on each AA management activity in their AAM reports. Furthermore, we also provided the subjects a template for AA Description (see Table 14 in Appendix A).

4.4. Units of analysis and data analysis procedure

The case is the course project. We initially checked the quality of their reports to ensure that the subjects had understood the process well enough to provide a reliable opinion. The reports were not analyzed further, as we focused on the opinions of the subjects on the AAM process. To answer the RQs, we used questionnaire, focus group, and documentation analysis for data collection; the students were not only the subjects, but also the units of analysis. Therefore, the study was a single and embedded case study [6].

We used descriptive statistics for quantitative data (e.g., background information of the subjects) and Constant Comparison [8] to analyze qualitative data (i.e., iteratively generating concepts and categories to answer the RQs). Constant Comparison provides a systematic way to generate, for example, concepts from incidents, and a continuous process of verification of the generated concepts and categories [8,35]. In this study, we followed the guidelines proposed by Adolph et al. [35] to iteratively perform the process. We coded the extracted data as incidents, compared these incidents to each other to generate concepts, and further performed comparison among incidents and concepts to generate categories. MAXQDA⁵ was used as the tool for qualitative data analysis. The first author was responsible for the whole analysis process, while in each iteration, the second author checked the data (including the generated incidents, concepts, and categories). In the end of each iteration, the first and second author had a meeting to address conflicts (e.g., whether a piece of data is correctly coded), and then the first author maintained the data according to the review and discussion. In the last iteration, the third author was involved to check the final results.

The relationships among the data collection methods, data analysis methods, and RQs are shown in Table 5.

We had explicitly mentioned to the subjects that all the data items using Likert Scales could be considered as interval variables. For example, in a five-point interval Likert Scale, the difference between 1 and 2 is exactly the same as the difference between 4 and 5. This allows us to calculate means and standard deviations, which would not be possible with an ordinal scale. Showing means and standard deviations helps to better understand the results for a large audience [37]. This practice has been observed in several studies that considered Likert Scale items as interval variables (e.g., [36,42,43,45]).

⁵ <http://www.maxqda.com/>.

Fig. 7. Ease of understanding the AA management activities and process.

5. Results

This section presents the results of the RQs.

5.1. Results of RQ1 – easy to understand

As shown in Fig. 7 (1 = “Very difficult” and 5 = “Very easy”), the majority of the subjects thought that the ease of understanding individual AA management activities and the entire AAM process were moderate (i.e., “3”), except for AA Evaluation (34 subjects chose “4”, while 33 subjects chose “3”). Table 6 shows the means and standard deviations of the results in Fig. 7. Considering the number of the subjects who chose “4” or “5” in Fig. 7, understanding AA Description (45 out of 88, 51.1%) and Evaluation (43 out of 88, 48.9%) are easier than AA Making (29 out of 88, 33.0%) and Maintenance (i.e., 33 out of 88, 37.5%); to some extent, understanding the AAM process is easy, i.e., 30 subjects (out of 88, 34.1%) who chose “4” or “5” versus 14 subjects (out of 88, 15.9%) who chose “1” or “2”.

The subjects mentioned a couple of reasons that hamper understandability of the AAM process. First, AAs and their management (e.g., usage of the AA concept, distinguishing between AAs and other types of software artifacts, and understanding the motivation of managing AAs) are not commonly applied in software development. For example, one subject stated that: “AA is a very abstract concept and needs certain effort to understand it. Some terms related to AA seem to be directly translated from English to Chinese, which are not easy to understand.” Second, the subjects stated that there was a lack of practice and experience for managing AAs in projects, and materials regarding learning AAs as well as their management. For example, one subject stated that: “except for the course materials (e.g., the PPT slides), we didn’t find much information of AAs and their management in Chinese from other sources (for example the Internet), which increased the difficulty of understanding the AAM process.”

Table 6

Means and standard deviations of ease in understanding the AA management activities and process.

	Making	Description	Evaluation	Maintenance	AAM process
Mean	3.15	3.53	3.45	3.27	3.18
Standard deviation	0.90	0.90	0.86	1.00	0.74

Fig. 8. Effort of conducting the AA management activities and process.

5.2. Results of RQ2 – required effort

As shown in Fig. 8 (1 = “Very time-consuming” and 5 = “Not time-consuming”), most of the subjects agreed that the effort of conducting individual AA management activities and the entire AAM process was moderate (i.e., “3”). Table 7 shows the means and standard deviations of the results in Fig. 8. Considering the number of the subjects who chose “4” or “5” in Fig. 8, the required effort of AA Description (34 out of 88, 38.6%) and Maintenance (30 out of 88, 34.1%) are less than AA Making (13 out of 88, 14.8%) and Evaluation (i.e., 19 out of 88, 21.6%); to some extent, conducting the AAM process is time-consuming, i.e., 36 subjects (out of 88, 40.9%) who chose “1” or “2” versus 9 subjects (out of 88, 10.2%) who chose “4” or “5”.

The subjects identified a number of reasons regarding the required effort of using the AAM process and how it could be improved:

- (1) The communication within a development team as well as with their customer team was not always efficient and effective. For example, as one subject put it: “My group members always had different opinions regarding AAs, and the communication between us was rather inefficient. Therefore, we had to spend more time to reach a consensus.”

Table 7

Means and standard deviations of effort in conducting the AA management activities and process.

	Making	Description	Evaluation	Maintenance	AAM process
Mean	2.64	3.23	2.82	3.05	2.64
Standard deviation	0.86	0.92	0.86	1.02	0.75

- (2) There was a lack of specific approaches for each AA management activity. For example, as one subject stated: *“Though we understood what AA Making and Evaluation are, we did not know how to make or evaluate AAs exactly in the first place, so it took us certain time to come up with an approach for AA Making and Evaluation.”*
- (3) The output quality of the AA management activities could be low. For example, as one subject mentioned: *“AA Making and Description were the basis of AA management: sometimes we neither made AAs clearly nor described AAs in detail, which made, for example, AA Evaluation difficult to perform.”*
- (4) Before conducting the AAM process, the subjects needed to spend certain time on understanding the process (including the AA concept). For example, as one subject put it: *“The effectiveness of conducting the AAM process depended on the understanding of the AA concept. Without a fair understanding of the AA concept, it made AA Making difficult to perform, and it was difficult for the group members to reach an agreement in AA Evaluation.”*
- (5) Several groups had a large number of AAs in their projects, which took them certain effort to manage the AAs. For example, as one subject stated: *“Because we had a lot of AAs to manage, it took us much effort in, for example, AA Maintenance.”*
- (6) AAs were intertwined with other types of software artifacts (e.g., requirements and design decisions) in the projects, and the effectiveness of managing these artifacts influenced the effort of conducting the AA management activities. As one subject mentioned: *“Other types of software artifacts were not always well managed, which led to the increased effort of conducting the AAM process.”*
- (7) Some subjects did not have much experience (e.g., architecting and project experience). As one subject explained: *“Without much project experience, it took us more time to understand requirements, architecture, etc. when we were making AAs.”*

We provide the actual time spent by each group in Appendix E. Concerning the most time-consuming AA management activities according to Fig. 10 in Appendix E, AA Making (10 of 21, 47.6%) is ahead of the other activities, followed by Description (8 out of 21, 38.1%), Evaluation (6 out of 21, 28.6%), and Maintenance (6 out of 21, 28.6%). A group can spend equal time on different AA management activities (e.g., Group 6 spent equal time on AA Making and Description), and therefore, the total number of the most time-consuming AA management activities (i.e., 30) is larger than the number of groups (i.e., 21).

5.3. Results of RQ3 – making AAs explicit

As we noted in Section 4.1, making AAs explicit means that the relevant stakeholders in a group (i.e., those that need to know, such as the development team) are aware of the AAs made; we made this clear to the subjects. We asked the subjects whether the AAM process was helpful in making AAs explicit in their projects, and most of the subjects (81 out of 88, 92.0%) answered positively to this question: 53 subjects (out of 88, 60.2%) chose “1” (i.e., “Very helpful”) and 28 subjects (out of 88, 31.8%) chose “2” (i.e., “Helpful”).

Furthermore, we also asked the subjects to list the most helpful AA management activities in making AAs explicit. We clearly mentioned to the subjects that they could choose zero, one, or multiple activities when answering this question. As shown in Table 8, AA Evaluation is far ahead from the other AA management activities. For example, as one subject mentioned: *“I think AA Evaluation was the most helpful AA management activity for us to make AAs explicit because during Evaluation, if you want to evaluate an AA correctly and efficiently, you have to first get a fair understanding of this AA. Otherwise, you cannot evaluate it.”*

For those subjects who agreed that AA Making, Description, and Maintenance are the most helpful AA management activities in making AAs explicit, their reasons are various. For example, a typical answer is: *“AA Description helped me to understand the AAs made, because Description aims to decrease the abstract nature of AAs, and make AAs clear.”*

On the other hand, there are subjects that disagreed that certain AA management activities can help them in making AAs explicit. For example, one subject who did not choose AA Description stated: *“we described AAs independently and then discussed with other members. However, the description of the AAs from other members was vague and I could not understand some of them. This did not help to make AAs explicit.”* Furthermore, efficient communication with group members and the customer team when conducting the AAM process is of paramount importance in making AAs explicit: *“If there is a lack of communication between us, it would make information sharing rather difficult.”*

5.4. Results of RQ4 – identifying invalid AAs

We asked the subjects whether the AAM process was helpful in identifying invalid AAs in their projects, and most of the answers to this question (78 out of 88, 88.6%) are positive: 44 subjects (out of 88, 50.0%) considered that the AAM process

Table 8

The most helpful AA management activities in making AAs explicit.

Making	Description	Evaluation	Maintenance
47 (53.4%)	50 (56.8%)	78 (88.6%)	37 (42.0%)

Table 9

The most helpful AA management activities in identifying invalid AAs.

Making	Description	Evaluation	Maintenance
13 (14.8%)	25 (28.4%)	87 (98.9%)	43 (48.9%)

Table 10

The most helpful AA management activities in reducing the number of invalid AAs.

Making	Description	Evaluation	Maintenance
29 (33.0%)	25 (28.4%)	72 (81.8%)	45 (51.1%)

was very helpful (i.e., “1”) while 34 subjects (out of 88, 38.6%) thought it was helpful (i.e., “2”). The number of the AAs made as well as the invalid AAs identified by the groups can be found in Appendix F.

Furthermore, we also asked the subjects to list the most helpful AA management activities in identifying invalid AAs in their projects. We explicitly mentioned to the subjects that they could choose zero, one, or multiple activities when answering this question. As shown in Table 9, all the subjects except one agreed that AA Evaluation was the most helpful for them to identify invalid AAs, while AA Maintenance is in the second position. One of the subjects stated very explicitly: “The nature of AA Evaluation is to identify invalid AAs.”

There are also subjects that chose AA Making, Description, and Maintenance as the most helpful AA management activities in identifying invalid AAs. For example, one subject mentioned: “During AA Making, I could already find some invalid AAs.”

On the other hand, there are subjects that disagreed that certain AA management activities can help them in identifying invalid AAs. For example, one subject who did not choose AA Making and Description stated: “You cannot completely understand an AA during Making and Description. Only during Evaluation and Maintenance, you can decide whether the AA is invalid or not.” Moreover, understanding of the AAs made as well as the project context can influence identifying invalid AAs: “If I did not understand an AA as well as the related context of the AA, such as related artifacts, I would be hesitating to decide whether the AA was invalid or not.” The subjects further emphasized that efficient communication when conducting the AAM process can help to reduce the risk of incorrect judgments on AAs: “Involving everyone of the team in the discussion helped us to understand an AA from different perspectives, and therefore, helped us to make an informed and more reasonable decision about the validity of the AA.”

5.5. Results of RQ5 – reducing invalid AAs

Most of the subjects (77 out of 88, 87.5%) stated that using the AAM process could help in reducing the number of invalid AAs in their projects: 37 subjects (out of 88, 42.0%) considered that the AAM process was very helpful (i.e., “1”) and 40 subjects (out of 88, 45.5%) thought it was helpful (i.e., “2”). We explicitly mentioned to the subjects that they could choose zero, one, or multiple activities when answering this question.

For the most helpful AA management activities in reducing invalid AAs, AA Evaluation is the dominant one, followed by Maintenance, as shown in Table 10. For example, as one subject stated: “During AA Evaluation, we would identify some invalid AAs, and then as a natural step, we would focus on maintaining these invalid AAs.” For those subjects who chose AA Making and Description as the most helpful AA management activities in reducing the number of invalid AAs, one typical answer is: “When I was making AAs, I tried to be careful and spend more time on thinking; this helped me to reduce invalid AAs in AA Making.” On the other hand, there are subjects that disagreed that certain AA management activities could help them in reducing invalid AAs. For example, one subject who did not choose AA Making explained: “Whether an AA was invalid could not be decided during AA Making, and therefore, Making did not help to reduce invalid AAs.” As discussed in Sections 5.3 and 5.4, efficient communication is also of significant importance in reducing the number of invalid AAs when conducting the AAM process: “Communication with team members can help us to discuss whether an AA is invalid or not and how to maintain these invalid AAs.”

6. An industrial evaluation through focus groups

6.1. Study design

To further validate the process in an industrial setting, we conducted a shorter, exploratory study with five practitioners from five companies. They had at least one year of working experience in IT industry (ranging from one year to three years).

Table 11

Ease of understanding each AA management activity (1 = “Very difficult” and 5 = “Very easy”).

Subject	Making	Description	Evaluation	Maintenance
S1	5	5	5	5
S2	4	5	4	5
S3	5	5	4	5
S4	3	5	5	4
S5	5	5	5	3

Table 12

Expected effort of each AA management activity.

Subject	Most time consuming – Least time consuming
S1	Evaluation and Making; Description; Maintenance
S2	Evaluation; Making; Maintenance; Description
S3	Evaluation; Description; Maintenance; Making
S4	Evaluation; Description; Making and Maintenance
S5	Maintenance, Evaluation, Description, Making

As a new approach in the field of AAs and their management, it was difficult to convince companies to directly employ such an approach throughout the entire software development lifecycle in their projects. Therefore, we decided to perform the evaluation in three steps, i.e., (1) an explanatory case study with a large number of students that employed the AAM process in their course projects; (2) an exploratory case study with practitioners using focus groups; and (3) an explanatory case study with practitioners that employ the AAM process throughout the software development lifecycle (as future work, see Section 9).

In this study, we used similar research questions to those described in Section 4.1. The only difference is that we modified RQ2 from “required” to “expected”, as the industrial study is exploratory in nature and the subjects did not actually use the AAM process in the software development lifecycle.

The study was conducted in Groningen, the Netherlands with five practitioners through two focus groups for data collection (3 practitioners and 2 practitioners respectively; 1.5 hours per focus group). Each focus group is comprised of a presentation of the AAM process (0.5 hour; the same as we used for the student study) and a discussion with the practitioners (1.0 hour). The researchers organized the discussion according to the RQs. The practitioners were not only the subjects, but also the units of analysis.

6.2. Results

Results of RQ1 – easy to understand

All the subjects considered that the AAM process is generally easy to understand. Regarding each activity in the process, as shown in Table 11, Description was mentioned as the easiest to understand, followed by Evaluation. The subjects pointed out that it was not difficult to understand either the activities themselves (e.g., their definition) or the connections between the activities; but understanding the AA concept requires some effort (e.g., what an AA looks like, and its characteristics and properties). Additionally, the subjects considered that the term “AA Making” may confuse stakeholders, since it includes two parts: making new AAs and identifying existing AAs. This could be addressed by either adapting the term “AA Making” to “AA Making and Identification” or separating them into two activities.

Results of RQ2 – expected effort

Regarding the expected effort of each AA management activity, Table 12 shows the results. Four subjects (out of 5, 80%) considered Evaluation as the most time consuming AA management activity. As one subject mentioned: “*Evaluation means really thinking a lot. Description is typing. You can do that quite fast.*”

It is not clear which AA management activity requires the least effort according to Table 12, since the subjects had different opinions. As an example regarding AA Making, one subject explained: “*I am not sure whether Making or Evaluation takes the most effort, because in Making, you talk about the type of knowledge that is not ready in your head. So you have to do deep thinking to figure it out. It's not something that you can easily do like asking what did you assume. It's something at the circumstance, really have to go back like what did I do and how; why did I do this. So it can be quite a lot of effort.*” Another example that acts as a counterpoint from a different subject: “*Making assumptions is sort of human nature, so it's not time consuming.*”

The subjects also mentioned several specific ways to reduce the effort of managing AAs. As an example, a subject suggested using predefined classifications and tags to mark each AA, and said that this could facilitate not only AA Description, but also AA Searching and Organization.

Results of RQ3 – making AAs explicit

All subjects considered that following the AAM process can help to make AAs explicit. Regarding the most helpful AA management activities, the subjects mentioned that it depends on when stakeholders share their knowledge in managing AAs. According to the guidelines in Table 2 as well as their experience, this should happen mostly in Making and Evaluation. As one subject mentioned: “*When do you share the information? To me the only one you share information is either Making or*

Evaluation. If you do Making in a group, you discuss and try to identify AAs. All the evaluation is about discussing the AAs you have written down."

One subject explained this from another angle: *"The step of evaluating an assumption comes after you have an assumption. If you want to raise the awareness, you first have to make people be aware what is an assumption, and what is not. So this makes Description the most important."* Though Description is not a sharing activity per se, it acts as a basis for disseminating AAs to different stakeholders.

Results of RQ4 – identifying invalid AAs

All subjects agreed that following the AAM process can help to identify invalid AAs. Considering the most helpful AA management activities, four subjects (out of 5, 80.0%) pointed out that it should be Evaluation. As one subject put it: *"Evaluation is meant to do this."* However, one subject mentioned that Making should be the most helpful activity in identifying invalid AAs, followed by Evaluation: *"You could identify invalid AAs in the first place. This happens in Making."*

Results of RQ5 – reducing invalid AAs

All subjects agreed that following the AAM process can help to reduce invalid AAs. Regarding the most helpful AA management activities, the subjects considered that both Evaluation and Maintenance could contribute to it, though they had an argument on which one is more helpful in the first place. As one subject put it: *"Maintenance is when you make the decisions, when you tell what is what, and what is changing. You do this in Maintenance."* Another subject disagreed: *"Description and Maintenance are something like documentation activities, and the really hard work happens here, in this case, Evaluation. Evaluation is the thinking activity."* This lack of consensus was attributed by the participants themselves to two factors: (1) how they understood Evaluation and Maintenance, especially what stakeholders do in these two activities; (2) whether preparation for reducing invalid AAs contributes to the helpfulness. As one subject put it: *"In Evaluation, we identify invalid AAs, and decide whether we should maintain these invalid AAs. This should count as helpful for reducing invalid AAs."* Moreover, the subjects mentioned that although Evaluation or Maintenance can help to reduce invalid AAs, it is also critical to identify them in the first place, i.e., in Making.

7. Discussion

This section first presents the interpretation of the results of the RQs, and subsequently discusses the implications of the results for researchers and practitioners. "subjects" in this section refers to both the students and practitioners.

7.1. Interpretation of results

As discussed in Section 4.2, the quantitative results among different groups in the student study are not comparable. For example, how many AAs the groups made, highly depended on how the students treated the AA concept and the context of the projects (some projects may be prone to have more AAs than the others).

RQ1 – Ease of understanding the AAM process

The majority of the students thought that the ease of understanding the AAM process was moderate, and all the practitioners considered that the process was easy to understand. The process seems rather straightforward to both students and practitioners. Though some subjects mentioned that for example AA Making is more difficult to understand than the other activities, we attribute this not to the activity itself or connections between the activities, but to understanding the different concepts (e.g., the AA concept itself). Regarding the difference between the students and practitioners, we attribute this mainly to experience, since all the practitioners were experienced in architecting, and therefore had fewer problems in understanding the AAM process. Furthermore, the results of RQ1 also show that the standard deviation of each AA management activity and the AAM process ranges from 0.74 to 1.00, which implies that the opinions of most of the students on this RQ are uniform.

RQ2 – Effort of conducting the AAM process

We measured the required effort for conducting the AAM process through both Likert Scale scores and logging the time spent on individual AA management activities in the student study. The former shows that most of the students considered that the required effort for conducting the AAM process was moderate. The effort was impacted by various factors, for example, the students stated that it took them certain time to understand the AAM process. A complete summary of the factors can be found in Table 13. Furthermore, the standard deviation of each AA management activity and the AAM process ranges from 0.75 to 1.02, which implies that the opinions of most of the students on this RQ are uniform.

On the other hand, the results of logging the time spent on individual AA management activities show that the average time (per group) of conducting AA Description (8.97 hours/group) and Maintenance (7.23 hours/group) are less than AA Making (11.22 hours/group) and Evaluation (9.08 hours/group); this is consistent with the results from using Likert Scale scores (see the first paragraph of Section 5.2). One potential reason for the increased effort in AA Making could be that during the first iteration of the project, the students had to spend certain time on understanding the AAM process while making AAs.

Since the practitioners did not actually perform the AAM process in the software development lifecycle, the results regarding effort from the students and practitioners are not comparable. However, we noticed that most of the practition-

Table 13

Factors for applying the AAM process.

Factor	Subfactor	Description	RQs
Concepts understanding		A fair understanding of the elements (e.g., the AA concept) in the AAM process can help to understand the process better.	RQ1
Learning approach		Various approaches were suggested to facilitate the understanding of the AAM process, including frequent communication with other stakeholders, learning from tutorial materials, following well-designed guidelines, and searching on the Internet.	RQ1
Quality of learning materials		It is straightforward that the higher quality of learning materials, the easier to learn the AAM process.	RQ1
Practice		More practice of the AAM process, for example, using the process in different projects, can improve the understanding of the process.	RQ1
Experience in managing AAs		Previous experience in managing AAs can help to understand the AAM process.	RQ1
AAM process	Understanding of the AAM process	A fair understanding of the AAM process can reduce the effort of performing the process and help to identify and reduce invalid AAs.	RQ2, RQ4, RQ5
	Approaches and tools	Dedicated approaches and tools for specific AA management activities can reduce the effort of conducting the AAM process, help to make AAs explicit, and facilitate identifying and reducing invalid AAs. For example, the importance of communicating with team members and customers in an efficient way (e.g., following guidelines or using dedicated tools) was frequently mentioned by the subjects.	RQ2, RQ3, RQ4, RQ5
	Quality of the outputs of individual AA management activities	The quality of the outputs of each AA management activity can influence the effort of applying the AAM process, making AAs explicit, and identifying and reducing invalid AAs. For example, a clear and concise description of AAs can facilitate the evaluation of the AAs.	RQ2, RQ3, RQ4, RQ5
	Number and type of participants	AA management activities may be conducted as teamwork. The number and type of participants could have an impact on the effort of applying the AAM process, making AAs explicit, and identifying and reducing invalid AAs. For example, if there are too many people in a room, it is difficult to get things done (e.g., making AAs).	RQ2, RQ3, RQ4, RQ5
	Division of labor	Division of labor can impact making AAs explicit. As an example, if a subject is responsible for only a single AAM activity, s/he may not be aware of every AA made in the group.	RQ3
	Time spent on AA management	AA management takes a certain amount of time. If a subject only spends little time on managing AAs, s/he may not be aware of every AA made in the group.	RQ3
Project context	Software artifacts	AAs are intertwined with certain types of software artifacts, and managing AAs is significantly impacted by the management of these artifacts. For example, if requirements change frequently over time, the effort of managing related AAs will be increased.	RQ2, RQ3, RQ4, RQ5
	Number of AAs	Obviously more AAs in a project, more effort is required to manage them.	RQ2
Experience		Experience (e.g., project experience and architecture experience) can impact not only the understanding of the AAM process, but also the effort of using the AAM process, making AAs explicit, and identifying and reducing invalid AAs.	RQ2, RQ3, RQ4, RQ5
Time spent on AA arguments		The students pointed out that there were many arguments on AAs within their development team as well as between their development team and customer team when conducting the AAM process. The students had to spend much time on addressing these arguments.	RQ2

ers agreed that Evaluation would require the most effort, while they had different opinions on the least time consuming AA management activity (e.g., Making). The critical factor here is whether an activity requires much thinking. There was consensus in considering Evaluation as the thinking activity.

RQ3 – Effectiveness of using the AAM process to make AAs explicit

AA Evaluation was identified as the most helpful AA management activity for making AAs explicit in the student study. One reason could be that AA Evaluation had the highest involvement of the students in all the AA management activities. As we found from the results of the questionnaires and focus groups, the development teams employed various strategies in conducting AA Making, Description, and Maintenance. For example, the students might make and describe AAs independently and then discuss with each other in their teams; or they could make and describe AAs together as teamwork. However, this is not the case when performing AA Evaluation, since all the students from a development team and their customer team were involved in this activity, which led to the highest participation rate. We did not limit the students regarding which strategies they should use when conducting individual AA management activities (e.g., conducting AA Evaluation individually or collaboratively). Another reason for AA Evaluation being considered as the most helpful activity for making AAs explicit could be that a precondition of evaluating an AA is to have a fair understanding of this AA. Therefore the students would need, prior to evaluation, to understand AAs well enough, which made the AAs explicit.

The results of this RQ from the student study are consistent with the findings from the industrial study. The practitioners also pointed out that the most helpful activity in making AAs explicit depends on when stakeholders share their knowledge. This should happen in the activities that are conducted as teamwork (i.e., in the student study, it is Evaluation).

RQ4 – Effectiveness of using the AAM process to identify invalid AAs

AA Evaluation is far ahead from the other three activities for identifying invalid AAs in both the student and industrial study. One reason is that the nature of AA Evaluation is to ensure that the description and analysis of AAs are correct and accurate, driving the subjects to identify invalidity issues in AAs, for example, incorrect descriptions. Furthermore, as we explained in the interpretation of the results of RQ3, all the groups in the student study adopted the same strategy when performing AA Evaluation: the entire development team as well as their customer team was involved in this activity. This is another reason that the majority of the students thought that AA Evaluation was the most helpful to identify invalid AAs in projects.

RQ5 – Effectiveness of using the AAM process to reduce invalid AAs

The majority of the students thought that AA Evaluation is the most helpful AA management activity for reducing invalid AAs, followed by AA Maintenance. The practitioners also considered that both Evaluation and Maintenance are helpful. This is in line with our intent when designing the process: we designated Maintenance as the activity used to reduce invalid AAs. On the other side, the benefit of Evaluation for reducing invalid AAs is illustrated in two ways: (1) it had the highest participation in the student study; (2) the practitioners considered that in Evaluation, stakeholders *prepare* for reducing invalid AAs (e.g., deciding whether an AA needs to be maintained).

The practitioners also mentioned that it is important to reduce invalid AAs in the first place, i.e., in Making, which is consistent with our guidelines (see Section 3.2). However, they admitted that it is rather difficult, and that is why they did not consider Making as the most helpful activity for this RQ.

Summary of factors for applying the AAM process

As shown in Table 13, based on the analysis of the qualitative data, we identified a set of factors for applying the AAM process mapped to the RQs. Being aware of and properly adjusting these factors can facilitate the application of the AAM process in projects (e.g., helping to identify invalid AAs).

7.2. Implication for researchers

Students as subjects and experience: We conducted a student study as well as an industrial study, and found out that the students were able to understand the AAM process (including the AA concept) and apply the process in their projects, though they might need more time than practitioners for learning. Therefore, this study provides further evidence in treating specific types of students (e.g., first-year master students in software engineering) as valid subjects in software engineering case studies; it is of course important to follow certain guidelines (e.g., the checklist in [12]). Furthermore, the factors (e.g., “Learning approaches”) we identified in the case study (see Section 7.1) can impact not only the application of the AAM process, but also treating students as subjects in case studies. Being aware of and properly adjusting these factors can improve the validity of such studies. Finally, as mentioned by the students in the case study, there may be a significant difference between junior and senior software engineers in applying the AAM process. It is valuable for researchers to investigate how experience impacts the understanding as well as the usage of the AAM process, and further develop dedicated guidelines regarding the AAM process for junior and senior software engineers respectively.

AA Making and Identification: In this study, we merged making new AAs and identifying existing AAs in a single activity, i.e., AA Making, because it is difficult to draw a line between them. However, as mentioned by the practitioners, in some specific cases, it may be better to separate them in two activities. This requires more investigation.

AA Evaluation and Maintenance: In this study, we proposed that Evaluation entails deciding whether an AA needs to be maintained; the actual AA rework takes place during Maintenance. Though some practitioners agreed to this design, they mentioned that there are other options: (1) stakeholders only identify problems in Evaluation, but move all the other work in Maintenance; (2) stakeholders decide during Evaluation not only whether an AA needs to be maintained, but also how to maintain them, while Maintenance could be just simple documentation work merged in Description. We encourage researchers to find the best practice regarding what stakeholders should do in AA Evaluation and Maintenance.

Group and individual: In our guidelines, we recommended to conduct AA Making and Evaluation as teamwork, while Description and Maintenance could be done by either a group or individuals. Such guidelines should be further refined, since there are still some unknowns, such as whether making AAs as teamwork is always the best practice, or when stakeholders should maintain AAs as a team or individually.

Approach and tool support: We did not propose any dedicated approach or tool for conducting the individual AA management activities. As mentioned by the subjects, dedicated approaches and tools for specific AA management activities can reduce the effort of conducting the AAM process, help to make AAs explicit, and facilitate identifying and reducing invalid AAs. As an example, the subjects mentioned that after writing down the AAs made, if a tool could send a notification to related stakeholders, it would be helpful for raising the awareness of the AAs. On the other side, one subject mentioned that though MS Word or PowerPoint is not suitable for managing AAs, Excel could be a good choice, as it is well accepted in industry. This result is against our previous findings [5], i.e., practitioners do not accept Excel as a suitable tool for AA management, which requires further investigation. Furthermore, the effectiveness of performing the AA management activities is of significant importance in managing AAs in software development. How to ensure for example, the quality of the inputs and outputs of the AAM process is another interesting direction.

AAs and other types of software artifacts: As evidenced in this study, managing AAs is significantly impacted by the management of certain types of artifacts. For example, if requirements change frequently over time, the effort of managing related AAs will be increased. This is consistent with our previous conclusions (e.g., AAs are intertwined with certain types of artifacts). Therefore, there is a need to investigate how AAs are related to those artifacts, as well as how AAs and their management can be integrated into the software development lifecycle.

7.3. Implication for practitioners

Understanding of AA: As evidenced in literature (e.g., [4]), assumptions and their management has been a broad topic in software engineering, and AAs are important and should be managed in both architecting and software development. However, the terms as well as the concepts related to AAs are not commonly used in industry. We suggest that practitioners acquire a fair understanding of what AAs are, the problems caused by not well-managed AAs, etc. before applying the AAM process in their work.

AAM process: Implicit and invalid AAs could lead to various problems in software development, while AAs are usually not well managed in industrial practice. In the two case studies, we provide evidence regarding how the AAM process helps to deal with implicit and invalid AAs with acceptable effort of learning and conducting it. It is of course significant to integrate the AAM process into existing software development activities (e.g., architecting), and adapt the AAM process to specific contexts of projects (e.g., agile software development).

Predefined classifications and tags: As suggested by the subjects, stakeholders could use predefined classifications or tags in AA Description, Searching, and Organization. Though we agree to the suggestion on using tags, which could be beneficial in AA management, we conjecture that a predefined classification may also have challenges: (1) it may increase the complexity of AA management; (2) it may lead to inconsistencies and misunderstandings among stakeholders (e.g., regarding whether an AA should belong to which type); (3) there is no well-accepted AA classification.

Communication and teamwork: Managing AAs is teamwork. As mentioned by the subjects in the case study, effective and efficient communication among team members as well as between the development and customer teams is of paramount importance in managing AAs. We suggest that practitioners encourage not only architects, but also other types of related stakeholders (e.g., requirements engineers, project managers, and customers) to be involved in AA management.

Experience: It is not surprising that experience was considered as an important factor for the ease of understanding and effort of conducting the AAM process, making AAs explicit, and identifying and reducing invalid AAs. For example, senior practitioners may make less AAs than junior practitioners in projects, because it would be easier for senior practitioners to identify whether a piece of information is an AA or an AA is invalid at the time it was made based on their knowledge and experience. However, since there are various situations in projects (e.g., junior practitioners are responsible for AA management, only involved in AA management, or do not need to consider AAs), the impact of experience on AA management would be different. As an example, when junior practitioners are responsible for AA management (especially in AA Making and Evaluation), we suggest that junior practitioners frequently discuss with senior practitioners regarding AA management in order to, for example, identify and reduce invalid AAs in their projects. We also encourage senior practitioners to share their knowledge and experience of managing AAs (e.g., the reasons of considering an AA as invalid) with other stakeholders, for example, through internal discussions or documentation within an organization.

8. Threats to validity

We discuss the threats to the validity of this case study based on the guidelines proposed by Runeson and Höst [6]. This work does not study causality, so internal validity is not considered.

Construct validity reflects to what extent the research questions and the studied operational measures are consistent [6]. All the threats in construct validity have been addressed and reduced.

A potential threat is that the subjects used the AAM process without a basic understanding of the process (including the AA concept). To mitigate this threat, regarding the student study, (1) we gave a tutorial and had a discussion on the AAM process (e.g., what are AAs) with the students before they managed AAs in their projects; and (2) we organized a presentation session half-way through the course which allowed each group to present their projects (including explaining three AAs selected from their AAM reports). As discussed in Section 4.2.1, we did not focus on “technical” problems of their reports (e.g., whether a statement was an assumption or not). Instead, the purpose was to grade and improve readability of their reports. Regarding the industrial study, we gave a tutorial and had a discussion on the AAM process (e.g., what AAs are) with the practitioners before digging into the RQs.

Additionally, grading the AAM reports may lead to a risk that the students tried to compliment the AAM process in order to please the course lecturers (i.e., the authors) through the questionnaires and focus groups. However, as discussed in Section 4.2.2, we integrated the AAM process into requirements engineering and architecture design as a natural part of the course, instead of telling the students that we would evaluate a new approach (i.e., the AAM process). The students did not know that the authors designed the AAM process. Furthermore, we explicitly mentioned to the students that the questionnaires they filled in and the focus group they attended would not be counted in their final grade. Finally, the data extracted and analyzed from the AAM reports is only related to RQ2 and RQ4, i.e., the required effort and effectiveness in identifying invalid AAs. Regarding RQ4, the purpose of Fig. 11 and Fig. 12 is to provide an overview of the AAs made as well as the invalid AAs identified by each group, but the numbers of the 22 groups within these two figures are not comparable. Hence the data analysis of the AAM reports did not impact the results of RQ1, RQ3, RQ4, and RQ5. We admit that the risk was not completely eliminated, but had been reduced.

Whether the collected data can answer the RQs is another threat. To reduce this threat, we iteratively refined the RQs and the data collection strategies used in this work. For example, we decided not to use interview because there were 88 students, and the resource (e.g., time) of the course was limited; it was not realistic to conduct interviews in the case study. Moreover, we employed data triangulation, a technique of using various angles towards the studied object in order to provide a broader picture of the study [6], through three data collection methods: questionnaire, focus group, and documentation analysis.

8 students (out of 88, 9.1%) had neither participated in any software project (i.e., academic, industry, or open source projects), nor attended any course of software engineering. To maximize the randomness, we grouped the students according to their student numbers, and we ensured that these 8 students had been allocated to different groups; therefore the threat of having inexperienced subjects has been mitigated.

There is a risk that the time logged by the students for conducting individual AA management activities is not accurate and reliable. For example, the students might exaggerate the actual time spent in order to get a favorable impression from their teachers. To reduce the risk, we explicitly emphasized to the students not to exaggerate the time; we also measured the effort of conducting the AAM process (including each AA management activity) through Likert Scale scores (see Questionnaire B of Appendix B), in order to compare with the logged time.

Moreover, there is a potential threat that the answers of Questionnaire B (see Table 17 in Appendix) by a student were influenced by her/his group members. To mitigate this threat, in the last week when they filled in Questionnaire B, we arranged the students in two classrooms and set four researchers (two at each classroom) as invigilators.

Finally, since all the students are Chinese and the course was taught in Chinese (including the tutorial materials, questionnaires, and the AAM report used in the course), there is a threat that the translation of the tutorial documents (e.g., the slides and user guide) from English to Chinese as well as the translation from Chinese to English regarding the collected data from questionnaires, focus groups, and documentation analysis, could have resulted in information vaporization and “lost in translation” issues. To mitigate this threat, we followed a set of strategies:

- (1) We had conducted a survey [10] and several case studies (e.g., [5,29]) on the topic of AAs in industry with different practitioners (e.g., architects), who are Chinese speakers. In this study, we reused certain materials from those studies to ensure that the used terminology was mature.
- (2) In our previous studies on AAs, 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 the threat in this study.
- (3) Even though the tutorial materials are in Chinese, the authors also provided the students with an English version of the materials as references.
- (4) Translation of the tutorial materials: (a) The first author developed an initial Chinese version of the materials based on the English version and our previous studies (e.g., [5,10,29]). (b) The second author reviewed the Chinese version, and had a discussion with the first author to address conflicts. (c) Then the first author maintained the Chinese version based on the review and discussion. This process (review – discussion – maintenance) was conducted iteratively.

- (5) Translation of the data collected: (a) The first author was responsible for the translation. (b) The second author reviewed both the data in Chinese and English, while the third author reviewed the data in English, and all the authors had a discussion to address conflicts. (c) Then the first author maintained the translation based on the review and discussion. This process (review – discussion – maintenance) was conducted iteratively.
- (6) The discussion (including Q&A) with the subjects as well as the presentation in the middle of the study further helped to reduce this threat. During the case study, even though some subjects mentioned that they could not find much information of AAs and their management in Chinese from other sources (for example the Internet), no subject complained that s/he could not understand the concepts.

Since the industrial study was conducted by using English, we did not have the aforementioned threat in the industrial study.

External validity concerns the generalization of the findings [6]. The student study was conducted at Wuhan University with first-year master students in software engineering as the subjects. There is a risk that the findings of this study cannot be generalized into other contexts (and especially industry). We collected background information of the students, and found that the vast majority of the students were educated in software engineering at the undergraduate level and had project experience (i.e., academic, industry, or open source projects). Most of the students were close to their professional careers as software engineers. Moreover, we also conducted an industrial study to complement the student study. However, since the practitioners did not actually use the AAM process in their work, whether the results of the case studies can be generalized still needs more evidence. Replication of this work through a longitudinal study can mitigate the risk.

Reliability focuses on whether the study would yield the same results when other researchers replicate it [6]. To mitigate personal bias, the case study protocol had been reviewed and discussed by the three authors iteratively. We also presented the case study design in a meeting with eight software engineering researchers, and collected feedback from them to improve the design. Furthermore, to reduce the risk of low quality deliverables submitted by the students, the first and second authors checked their AAM reports every two weeks (i.e., four iterations in total) during the course project. Finally, the focus groups were recorded through audio recording devices to reduce the threat of information vaporization.

9. Conclusions and future work

AAs are an important type of architectural knowledge in both architecting and software development, and there is a need to improve the management of AAs. In this work, we propose an AAM process, comprised of four AA management activities: AA Making, Description, Evaluation, and Maintenance.

We conducted an explanatory study with 88 first-year master students in software engineering, and an exploratory study with five practitioners from five companies. Particularly we validated the ease of understanding and the effort of conducting the AAM process, as well as the effectiveness of using the AAM process to make AAs explicit and to identify and reduce invalid AAs.

The results indicate that (1) the ease of understanding the AAM process is moderate for first-year master students but easy for practitioners; (2) the effort of conducting the AAM process is moderate for first-year master students; (3) Making and Evaluation took the students more time than Description and Maintenance; (4) the practitioners considered Evaluation as the most time consuming activity; (5) the AAM process can help to make architectural assumptions explicit and to identify and reduce invalid architectural assumptions in projects. The majority of the students and practitioners agreed that Architectural Assumption Evaluation is the most helpful activity for all these three aspects. For other activities, there are different opinions about their helpfulness; and (6) there are various factors identified that can impact the aforementioned results. Being aware of and properly adjusting these factors can facilitate the application of the AAM process in projects.

Future work includes (1) investigating how the identified factors in this work influence the application of the AAM process in software development; (2) investigating which types of AAs should be managed in projects; (3) developing detailed guidelines for individual AA management activities of the AAM process; (4) developing guidelines for different stakeholders (e.g., junior and senior software engineers) to manage AAs; (5) investigating how AA management can be integrated into software development processes; and (6) evaluating the AAM process in an industrial context with a longitudinal study.

Acknowledgements

The authors wish to thank the students who participated in the case study during the requirements engineering and software architecture course of Autumn Semester 2016 at Wuhan University in China, and the practitioners who participated in the focus groups in the Netherlands. We would also like to thank Daniel Feitosa for reviewing an earlier version of the paper. This work is partially sponsored by the NSFC under Grant No. 61472286 and the Ubbo Emmius scholarship program by the University of Groningen.

Appendix A. Assumption management process report

Table 14
Architectural Assumption Description template.

Element	Type of answers
ID	Text
Name	Text
Description	Text
State	Valid/Invalid
Rationale	Text
Invalidity Reason	Text
Related architectural assumptions	Text
Related software artifacts	Text

Table 15
Architectural assumption management process report.

Section	Description				
Section 0	Division of labor and time log				
	<table><tr><th>Subject</th><th>Tasks (AA management activity)</th><th>Time (hour)</th><th>Date</th></tr></table>	Subject	Tasks (AA management activity)	Time (hour)	Date
Subject	Tasks (AA management activity)	Time (hour)	Date		
Section 1	AA Making				
Section 1.1	Overview of the AAs made (e.g., number)				
Section 1.2	Problems and solutions in conducting AA Making				
Section 2	AA Description (see Table 14)				
Section 2.1	Description of AA1				
Section 2.2	Description of AA2				
...					
Section 2- <i>n</i>	Description of AA <i>n</i>				
Section 2- <i>n</i> + 1	Problems and solutions in conducting AA Description				
Section 3	AA Evaluation				
Section 3.1	Results of AA Evaluation				
	(1) What AAs have been evaluated?				
	(2) What issues have been found on an evaluated AA?				
	(3) What are the plans for the problematic AAs?				
Section 3.2	Problems and solutions in conducting AA Evaluation				
Section 4	AA Maintenance				
Section 4.1	Results of AA Maintenance				
	(1) What AAs have been maintained?				
	(2) What are the actions (e.g., remove) for the maintained AAs?				
Section 4.2	Problems and solutions in conducting the AA Maintenance activity				

Appendix B. Questionnaires

Table 16
Questionnaire A – background information of the subjects.

Question	Type of answers
Projects experience	
How many academic software engineering projects (e.g., course and bachelor projects) were you involved in?	Integer ≥ 0
Can you briefly describe the biggest academic project that you were involved in (e.g., your role, the duration, the team size, the size of the project (in Lines of Code), and a short description of the project)?	Text
How many industry projects (i.e., working in a team to develop software-intensive systems for industry) were you involved in?	Integer ≥ 0
Can you briefly describe the biggest industry project that you were involved in (e.g., your role, the duration, the team size, the size of the project (in Lines of Code), and a short description of the project)?	Text
Except for academic and industry projects, how many open source software projects (i.e., working in an open source community) were you involved in?	Integer ≥ 0
Can you briefly describe the biggest open source software project that you were involved in (e.g., your role, the duration, the team size, the size of the project (in Lines of Code), and a short description of the project)?	Text
Expertise	
What activities (e.g., courses, conferences, competitions, and projects) related to software engineering did you attend?	Text
What is your current level of knowledge about software engineering?	Beginner/Mediate/Advanced/Expert
Why do you think you are beginner/mediate/advanced/expert in software engineering?	Text
Have you ever received any professional training (i.e., excluding the courses at university) related to software engineering?	Yes/No

Table 17

Questionnaire B – answers of the RQs.

Question	Type of answers	RQ
How easy was it to understand AA Making?	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
How easy was it to understand AA Description?	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
How easy was it to understand AA Evaluation?	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
How easy was it to understand AA Maintenance?	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
How easy was it to understand the AAM process?	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
What factors could impact the understanding of the AAM process?	Text	RQ1
How much effort did you need to conduct AA Making?	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
How much effort did you need to conduct AA Description?	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
How much effort did you need to conduct AA Evaluation?	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
How much effort did you need to conduct AA Maintenance?	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
How much effort did you need to conduct the AAM process?	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
What factors could impact the effort of conducting the AAM process?	Text	RQ2
Did the AAM process effectively help to make AAs explicit (i.e., every member in the group are aware of the AAs made)?	Five-point Likert Scale (i.e., 1 = “Very helpful” and 5 = “Very helpless”)	RQ3
What AA management activities were the most helpful in making AAs explicit?	AA management activities	RQ3
What factors could impact the effectiveness of making AAs explicit?	Text	RQ3
Did the AAM process effectively help to identify invalid AAs?	Five-point Likert Scale (i.e., 1 = “Very helpful” and 5 = “Very helpless”)	RQ4
What AA management activities were the most helpful in identifying invalid AAs?	AA management activities	RQ4
What factors could impact the effectiveness of identifying invalid AAs?	Text	RQ4
Did the AAM process effectively help to reduce the number of invalid AAs?	Five-point Likert Scale (i.e., 1 = “Very helpful” and 5 = “Very helpless”)	RQ5
What AA management activities were the most helpful in reducing the number of invalid AAs?	AA management activities	RQ5
What factors could impact the effectiveness of reducing the number of invalid AAs?	Text	RQ5

Appendix C. Data items collected

Table 18

Data items related to the subjects.

Data item	Scale type	Unit	Range
Number of academic software engineering projects	Ratio	Project	≥ 0
Description of an academic software engineering project	N/A	N/A	Text
Number of industry projects	Ratio	Project	≥ 0
Description of an industry project	N/A	N/A	Text
Number of open source projects	Ratio	Project	≥ 0
Description of an open source project	N/A	N/A	Text
Activities related software engineering	N/A	N/A	Text
Level of knowledge about software engineering	N/A	N/A	Beginner/Mediate/Advanced/Expert
Reasons of the level of knowledge about software engineering	N/A	N/A	Text
Professional training (i.e., excluding higher education) related to software engineering	N/A	N/A	Yes/No

Table 19

Data items related to AAM report.

Data item	Scale type	Unit	Range	RQ
Time spent on AA Making	Ratio	Hour	≥ 0	RQ2
Time spent on AA Description	Ratio	Hour	≥ 0	RQ2
Time spent on AA Evaluation	Ratio	Hour	≥ 0	RQ2
Time spent on AA Maintenance	Ratio	Hour	≥ 0	RQ2
Overview of the AAs made	N/A	N/A	Text	RQ1–RQ5
Problems in conducting AA Making	N/A	N/A	Text	RQ1–RQ5
Solutions for the identified problems in conducting AA Making	N/A	N/A	Text	RQ1–RQ5
Number of AAs described	Ratio	AA	≥ 0	RQ1–RQ5
Details of AAs (ID, Name, Description, State, Rationale, Invalidity reason, Related architectural assumptions, and Related software artifacts)	N/A	N/A	Text	RQ1–RQ5
Problems in conducting AA Description	N/A	N/A	Text	RQ1–RQ5
Solutions for the identified problems in conducting AA Description	N/A	N/A	Text	RQ1–RQ5
Number of invalid AAs identified	Ratio	AA	≥ 0	RQ4
Results of AA Evaluation	N/A	N/A	Text	RQ1–RQ5
Problems in conducting AA Evaluation	N/A	N/A	Text	RQ1–RQ5
Solutions for the identified problems in conducting AA Evaluation	N/A	N/A	Text	RQ1–RQ5
Results of AA Maintenance	N/A	N/A	Text	RQ1–RQ5
Problems in conducting AA Maintenance	N/A	N/A	Text	RQ1–RQ5
Solutions for the identified problems in conducting AA Maintenance	N/A	N/A	Text	RQ1–RQ5

Table 20

Data items collected for answering RQs.

Data item	Scale type	Unit	Range	RQ
Ease of understanding AA Making	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
Ease of understanding AA Description	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
Ease of understanding AA Evaluation	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
Ease of understanding AA Maintenance	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
Ease of understanding the AAM process	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very difficult” and 5 = “Very easy”)	RQ1
Factors for understanding the AAM process	N/A	N/A	Text	RQ1
Effort for AA Making	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
Effort for AA Description	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
Effort for AA Evaluation	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
Effort for AA Maintenance	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
Effort for the AAM process	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very time-consuming” and 5 = “Not time-consuming”)	RQ2
Factors for using the AAM process	N/A	N/A	Text	RQ2
Effectiveness of the AAM process in making AAs explicit	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very helpful” and 5 = “Very helpless”)	RQ3
Most helpful AA management activities in making AAs explicit	N/A	N/A	AA management activities	RQ3
Factors for the effectiveness of making AAs explicit	N/A	N/A	Text	RQ3
Effectiveness of the AAM process in identifying invalid AAs	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very helpful” and 5 = “Very helpless”)	RQ4
Most helpful AA management activities in identifying invalid AAs	N/A	N/A	AA management activities	RQ4
Factors for the effectiveness of identifying invalid AAs	N/A	N/A	Text	RQ4
Effectiveness of the AAM process in reducing the number of AAs	Interval	N/A	Five-point Likert Scale (i.e., 1 = “Very helpful” and 5 = “Very helpless”)	RQ5
Most helpful AA management activities in reducing the number of AAs	N/A	N/A	AA management activities	RQ5
Factors for the effectiveness of reducing the number of AAs	N/A	N/A	Text	RQ5

Appendix D. Checklist used in the case study

(1) Ensure adequate integration of the study into the course topics

The objective of the case study is to validate the effectiveness of the AAM process in the context of requirements engineering and architecture design, while the goal of the course is to help students to acquire knowledge of requirements engineering and architecture design, and practice the learned knowledge in the course project. As evidenced in the systematic mapping study [4], assumptions should be managed from early stages (e.g., requirements engineering or software design [2,3,21]) of software development [4], and AAs are related to various types of artifacts (e.g., requirements and design decisions) [4,10]. Therefore, the case study is consistent with the topics of the course, and the course project is reasonable for meeting both the research and course objectives. We integrated AAs and their management (i.e., the AAM process) into the course. We also motivated the students by elaborating on the value of the case study from a pedagogical point of view (e.g., why AAs are important and need to be managed). Furthermore, we integrated the AAM report as part of their project report (i.e., an important part for their final grades of the course).

(2) Integrate the study timeline with the course schedule

The course started on 20/09/2016 and ended on 29/11/2016. The case study started on 27/09/2016 and ended on 29/11/2016. Except for the first week (i.e., 20/09/2016–26/09/2016, preparation), the students spent nine weeks (not full-time) on their projects, including using the AAM process to manage AAs. The case study timeline was integrated with the course schedule. Furthermore, the students were asked to submit a project report, including an AAM report, every two weeks (i.e., four iterations in total) during the course. In the AAM report, we asked each group to log the time (per subject) they spent on individual AA management activities. We also created a discussion group using a chat tool QQ⁶ for all the students to discuss issues they had regarding the course topics and provide feedback to them. Furthermore, we clearly stated that each student should spend at least five hours per week on their projects, including at least one hour on communicating with their customer teams. Thus, we partially reduced the risk that the quality of their work in the case study was impacted by other courses the students had to attend.

(3) Reuse artifacts and tools where appropriate

This case study is a further step of our previous work on AAs and their management (e.g., [4,5,10]), and therefore we reused some artifacts (e.g., documents) from those studies.

(4) Write up a protocol and have it reviewed

We iteratively designed, developed, and reviewed a protocol of the case study, which followed the guidelines proposed by Runeson and Höst [6]. We also presented the case study design in a meeting with eight software engineering researchers, and collected feedback from them to improve the design.

(5) Obtain subjects' permission for their participation in the study

We obtained the permission of all the students before the case study (i.e., the course project), explained the plan of the course and the tasks the students need to perform at the beginning of the course, and stated that the data collected for the course (e.g., their background information) would be kept confidential.

(6) Set subject expectations

We explained the purpose of the course as well as the benefits for attending the course to the students. We made sure to not mention any information to the students that could bias the results of the case study. For example, we integrated the AAM process into requirements engineering and architecture design as a natural part of the course, instead of telling the students that we would evaluate a new approach (i.e., the AAM process). Furthermore, we clearly mentioned the time the students need to spend on their project, i.e., at least five hours per week. Finally, the grading criteria for the project report (including the AAM report) were clearly specified and provided to the students.

(7) Document information about the experimental context in detail

We collected background information of the students through a questionnaire (see Section 4.3). We also presented to the students the goals of the course, the included topics, and the teaching methods employed in the course.

(8) Implement policies for controlling/monitoring the experimental variables

Since we did not compare approaches or tools (e.g., two or more groups of students that use different approaches or tools), the educational value for the students was the same. When groups of students were formed, we did not allow self-grouping, because it may cause problems such as creating an unrealistic context [12]. We grouped the students according to their student numbers to maximize the randomness. Furthermore, we elaborate on data collection and analysis procedures in Sections 4.3 and 4.4 respectively. Data collection was performed in a non-invasive way (i.e., as part of the students deliverables).

(9) Plan follow-up activities

At the end of the course, we discussed with the students regarding the objective of the case study as well as the potential threats to the validity, answered all the questions (e.g., related to the case study) from the students, and collected their feedback on this study. The discussion was also helpful for the students, because they acquired knowledge about, for example, empirical research.

(10) Build or update a lab package

⁶ <http://im.qq.com>.

Fig. 9. Time spent on the AAM process by the groups.

Fig. 10. Time spent on individual AA management activities by the groups.

We developed a lab package (in Chinese) for the course, which contains all the details of the case study, and can be used for future work.

Appendix E. Time spent on the AAM process by the groups

All the subjects logged the time (per subject) they spent on individual AA management activities in their AAM reports (see Section 0 of Table 15 in Appendix A). The results are shown in Fig. 9 (i.e., the time spent on the AAM process) and Fig. 10 (i.e., the time spent on each AA management activity). The subjects in Group 22 did not provide complete time information in their AAM report, as they claimed that they forgot to count the time spent on AA Evaluation and Maintenance, and therefore, we excluded the time data of this group in the answers to this RQ. Most of the groups (12 out of 21, 57.1%) spent 20 to 40 hours on conducting the AAM process during the course project, while three groups spent more than 120 hours, i.e., 30 hours per subject in average. We consider these three groups as outliers and further discussed it in Section 8. Moreover, Table 21 shows the means and standard deviations of hours that the groups spent on the AA management activities and process (excluding the aforementioned three groups).

Appendix F. Number of AAs made and invalid assumptions identified by the groups

In their AAM reports, all the groups identified a number of invalid AAs, ranging from 2 to 29 (average: 8; standard deviation: 6.06), as shown in Fig. 11. An AA can evolve over time, and therefore, an AA may turn to be invalid zero, one, or several times during its lifecycle. For example, consider the following iterations:

Iteration 1: A valid AA is made and described as, for example, AA1-1.

Iteration 2: AA1-1 is evaluated as invalid (i.e., AA1-2).

Iteration 3: AA1-2 is maintained and turns to be valid (i.e., AA1-3).

Iteration 4: AA1-3 is further evaluated as invalid (i.e., AA1-4).

This is why, for example, Group 11 only described 27 AAs in their AAM report, but 29 invalid AAs were identified.

Table 21

Means and standard deviations of hours the groups spent on the AA management activities and process.

	Making	Description	Evaluation	Maintenance	AAM process
Mean	11.22 hours	8.97 hours	9.08 hours	7.23 hours	36.50 hours
Standard deviation	7.26	4.79	6.93	3.61	11.45

Fig. 11. Number of invalid AAs identified by the groups. **Fig. 12.** Number of AAs described by each group in the AAM reports.

Fig. 12 shows the number of AAs described by each group in the AAM reports, ranging from 8 to 32 (average: 16; standard deviation: 6.00). Most of the groups (15 out of 22, 68.2%) had 11 to 17 AAs, 3 groups (out of 22, 13.6%) described less than 10 AAs, and 4 groups (out of 22, 18.2%) documented more than 20 AAs in their projects. The proportion of the invalid AAs identified to the AAs described in the AAM reports of the 22 groups ranges from 9.4% (i.e., Group 12) to 107.4% (i.e., Group 11).

The purpose of Fig. 11 and Fig. 12 is to provide an overview of the AAs made as well as the invalid AAs identified by each group. The numbers of the 22 groups in these two figures are not comparable, since the AA concept is subjective (as discussed in Section 4.2) and the project context differs from each other (as discussed in Section 4.2).

References

- [1] R. Roeller, P. Lago, H. van Vliet, Recovering architectural assumptions, *J. Syst. Softw.* 79 (4) (2006) 552–573.
- [2] D.V. Landuyt, E. Truyen, W. Joosen, Documenting early architectural assumptions in scenario-based requirements, in: *Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture, WICSA, and European Conference on Software Architecture, ECSA, Helsinki, Finland, 2012*, pp. 329–333.
- [3] D.V. Landuyt, 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, 2014*, pp. 170–184.
- [4] C. Yang, P. Liang, P. Avgeriou, Assumptions and their management in software development: a systematic mapping study, *Inf. Softw. Technol.* 94 (2) (2018) 82–110.
- [5] C. Yang, P. Liang, P. Avgeriou, U. Eliasson, R. Haldal, P. Pelliccione, T. Bi, An industrial case study on an Architectural Assumption Documentation Framework, *J. Syst. Softw.* 134 (12) (2017) 190–210.
- [6] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, *Empir. Softw. Eng.* 14 (2) (2009) 131–164.
- [7] R.K. Yin, *Case Study Research: Design and Methods*, 3rd edition, Sage, London, 2003.

- [8] B.G. Glaser, A.L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Aldine Publishing, New York, 1967.
- [9] M. Höst, B. Regnell, C. Wohlin, Using students as subjects – a comparative study of students and professionals in lead-time impact assessment, *Empir. Softw. Eng.* 5 (3) (2000) 201–214.
- [10] C. Yang, P. Liang, P. Avgeriou, A survey on software architectural assumptions, *J. Syst. Softw.* 113 (3) (2016) 362–380.
- [11] U. van Heesch, P. Avgeriou, A. Tang, Does decision documentation help junior designers rationalize their decisions? A comparative multiple-case study, *J. Syst. Softw.* 86 (6) (2013) 1545–1565.
- [12] J.C. Carver, L. Jaccheri, S. Morasca, F. Shull, A checklist for integrating student empirical studies with research and teaching goals, *Empir. Softw. Eng.* 15 (1) (2010) 35–59.
- [13] M. Kubickova, H. Ro, Are students “real people?” The use of student subjects in hospitality research, http://scholarworks.umass.edu/gradconf_hospitality/2011/Poster/87/, 2011.
- [14] P. Runeson, Using students as experiment subjects – an analysis on graduate and freshmen student data, in: *Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering, EASE*, Keele University, UK, 2003, pp. 95–102.
- [15] P. Berander, Using students as subjects in requirements prioritization, in: *Proceedings of the 3rd International Symposium on Empirical Software Engineering, ISESE*, Redondo Beach, CA, USA, 2004, pp. 167–176.
- [16] D. Tofan, M. Galster, I. Lytra, P. Avgeriou, U. Zdun, M.A. Fouche, R. de Boer, F. Solms, Empirical evaluation of a process to increase consensus in group architectural decision making, *Inf. Softw. Technol.* 72 (4) (2016) 31–47.
- [17] M. Galster, D. Tofan, P. Avgeriou, On integrating student empirical software engineering studies with research and teaching goals, in: *Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering, EASE*, Ciudad Real, Spain, 2012, pp. 146–155.
- [18] D.I.K. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N.K. Liborg, A.C. Rekdal, A survey of controlled experiments in software engineering, *IEEE Trans. Softw. Eng.* 31 (9) (2005) 733–753.
- [19] J.D. Arthur, R.E. Nance, A. Bazaz, O. Balci, Mitigating security risks in systems that support pervasive services and computing: access-driven verification, validation and testing, in: *Proceedings of the 22nd IEEE International Conference on Pervasive Services, ICPS*, Istanbul, Turkey, 2007, pp. 109–117.
- [20] D. Garlan, R. Allen, J.M. Ockerbloom, Architectural mismatch: why reuse is still so hard, *IEEE Softw.* 26 (4) (2009) 66–69.
- [21] D.V. Landuyt, W. Joosen, On the role of early architectural assumptions in quality attribute scenarios: a qualitative and quantitative study, in: *Proceedings of the 5th International Workshop on the Twin Peaks of Requirements and Architecture, TwinPeaks*, Florence, Italy, 2015, pp. 9–15.
- [22] P. Runeson, M. Host, A. Rainer, B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*, John Wiley & Sons, 2012.
- [23] J. Kontio, L. Lehtola, J. Bragge, Using the focus group method in software engineering: obtaining practitioner and user experiences, in: *Proceedings of the 3rd International Symposium on Empirical Software Engineering, ISESE*, Redondo Beach, CA, USA, 2004, pp. 271–280.
- [24] ARIANE 5 flight 501 failure report by the inquiry board, <http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html>, 1996.
- [25] W. Nam, P. Madhusudan, R. Alur, Automatic symbolic compositional verification by learning assumptions, *Form. Methods Syst. Des.* 32 (3) (2008) 207–234.
- [26] R. Seater, D. Jackson, R. Gheyi, Requirement progression in problem frames: deriving specifications from requirements, *Requir. Eng.* 12 (2) (2007) 77–102.
- [27] G.A. Lewis, T. Mahatham, L. Wrage, *Assumptions Management in Software Development*, Technical Report, Carnegie Mellon University, 2004.
- [28] A. Jansen, *Architectural Design Decisions*, PhD Thesis, University of Groningen, 2008.
- [29] C. Yang, P. Liang, P. Avgeriou, U. Eliasson, R. Haldal, 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, 2017, pp. 191–207.
- [30] R.C. de Boer, R. Farenhorst, P. Lago, H. van Vliet, V. Clerc, A. Jansen, Architectural knowledge: getting to the core, in: *Proceedings of the 3rd International Conference on Quality of Software Architectures, QoSA*, Medford, MA, USA, 2007, pp. 197–214.
- [31] D. Turk, R. France, B. Rumpe, Assumptions underlying agile software development processes, *J. Database Manag.* 16 (4) (2005) 62–87.
- [32] D. Rost, M. Naab, C. Lima, C. von Flach Garcia Chavez, Software architecture documentation for developers: a survey, in: *Proceedings of the 7th European Conference on Software Architecture, ECSA*, Montpellier, France, 2013, pp. 72–88.
- [33] C. Hofmeister, P. Kruchten, R.L. Nord, H. Obbink, A. Ran, P. America, A general model of software architecture design derived from five industrial approaches, *J. Syst. Softw.* 80 (1) (2007) 106–126.
- [34] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, M.A. Babar, A comparative study of architecture knowledge management tools, *J. Syst. Softw.* 83 (3) (2010) 352–370.
- [35] S. Adolph, W. Hall, P. Kruchten, Using grounded theory to study the experience of software development, *Empir. Softw. Eng.* 16 (4) (2011) 487–513.
- [36] U. van Heesch, P. Avgeriou, Naive architecting – understanding the reasoning process of students A descriptive survey, in: *Proceedings of the 4th European Conference on Software Architecture, ECSA*, Copenhagen, Denmark, 2010, pp. 24–37.
- [37] D. Tofan, M. Galster, P. Avgeriou, Difficulty of architectural decisions – a survey with professional architects, in: *Proceedings of the 7th European Conference on Software Architecture, ECSA*, Montpellier, France, 2013, pp. 192–199.
- [38] A. Tang, Y. Jin, J. Han, A rationale-based architecture model for design traceability and reasoning, *J. Syst. Softw.* 80 (6) (2007) 918–934.
- [39] C.B. Haley, B. Nuseibeh, Bridging requirements and architecture for systems of systems, in: *Proceedings of International Symposium on Information Technology, ITSIM*, Kuala Lumpur, Malaysia, 2008, pp. 1–8.
- [40] P. Lago, 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, 2004, pp. 214–227.
- [41] J. Savolainen, J. Kuusela, Consistency management of product line requirements, in: *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, Toronto, Ontario, Canada, 2001, pp. 40–47.
- [42] T.R. Knapp, Treating ordinal scales as interval scales: an attempt to resolve the controversy, *Nurs. Res. (N.Y.)* 39 (2) (1990) 121–123.
- [43] H.N. Boone, D.A. Boone, Analyzing Likert data, *J. Ext.* 50 (2) (2012) 1–5.
- [44] H. Ziv, D. Richardson, R. Klösch, *The Uncertainty Principle in Software Engineering*, Technical Report, 1997.
- [45] A.O.S. Barreto, A.R. Rocha, Analyzing the similarity among software projects to improve software project monitoring processes, in: *Proceedings of the 7th International Conference on the Quality of Information and Communications Technology, QUATIC*, Porto, Portugal, 2010, pp. 441–446.
- [46] P. Kruchten, P. Lago, 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, 2006, pp. 43–58.
- [47] V. Page, M. Dixon, I. Choudhury, Security risk mitigation for information systems, *BT Technol. J.* 25 (1) (2007) 118–127.
- [48] C. Flanagan, S.N. Freund, S. Qadeer, Thread-modular verification for shared-memory programs, in: *Proceedings of the 11th European Symposium on Programming, ESOP*, Grenoble, France, 2002, pp. 262–277.
- [49] P. Kroll, P. Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison–Wesley Professional, 2003.
- [50] R. Kazman, M. Klein, P. Clements, ATAM: Method for Architecture Evaluation, Technical Report, 2000, http://resources.sei.cmu.edu/asset_files/TechnicalReport/2000_005_001_13706.pdf.
- [51] ISO, ISO/IEC/IEEE Std. 42010–2011, *Systems and Software Engineering – Architecture Description*, 2011.
- [52] F. Shull, J. Singer, D.I.K. Sjøberg, *Guide to Advanced Empirical Software Engineering*, Springer, 2008.
- [53] I. Sommerville, Integrated requirements engineering: a tutorial, *IEEE Softw.* 22 (1) (2005) 16–23.

- [54] Z. Li, P. Liang, P. Avgeriou, Application of knowledge-based approaches in software architecture: a systematic mapping study, *Inf. Softw. Technol.* 55 (5) (2013) 777–794.
- [55] C. Yang, P. Liang, P. Avgeriou, A systematic mapping study on the combination of software architecture and agile development, *J. Syst. Softw.* 111 (1) (2016) 157–184.
- [56] D. Garlan, Software engineering in an uncertain world, in: *Proceedings of the 18th FSE/SDP Workshop on Future of Software Engineering Research*, FoSER, Santa Fe, New Mexico, USA, 2010, pp. 125–128.
- [57] T.M. Hesse, B. Paech, Supporting the collaborative development of requirements and architecture documentation, in: *Proceedings of the 3rd International Workshop on the Twin Peaks of Requirements and Architecture*, TwinPeaks, Rio de Janeiro, Brazil, 2013, pp. 22–26.
- [58] C. Yang, P. Liang, P. Avgeriou, T. Liu, Z. Xiong, Industrial evaluation of an architectural assumptions documentation tool: a case study, submitted for publication.