Architecture Viewpoints for Documenting Architectural Technical Debt

Zengyang Li [a], Peng Liang [b,c], Paris Avgeriou [a]
[a] Department of Mathematics and Computing Science, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands
[b] State Key Lab of Software Engineering, School of Computer, Wuhan University, Luojiashan, 430072 Wuhan, China
[c] Department of Computer Science, VU University Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands

Abstract:
Technical debt has attracted an increasing interest from researchers and practitioners in the software engineering domain. Currently, most approaches to managing technical debt focus on dealing with technical debt at source code level, while few methods deal with technical debt at architecture level. If architectural technical debt (ATD) is not effectively managed in the architecting process, the knowledge about ATD is not made available to involved stakeholders and the impact of ATD is not considered during architecture decision-making. Thus, the system's maintainability and evolvability can be intentionally or unintentionally compromised. As a result, architectures are costly to maintain and new features are difficult to introduce. To facilitate the management of ATD, it needs to be documented so that it becomes explicit to stakeholders. To this end, we propose a set of architecture viewpoints related to ATD (ATD viewpoints in short). Each viewpoint frames a number of concerns related to ATD. These ATD viewpoints together help to get a comprehensive understanding of ATD in a software system, thereby providing support for architecture decision-making. To evaluate the effectiveness of the ATD viewpoints in documenting ATD, we conducted a case study in a large telecommunications company. The results of this case study show that the documented ATD views can effectively facilitate the documentation of ATD. Specifically, the ATD viewpoints are relatively easy to understand; it takes an acceptable amount of effort to document ATD using the ATD viewpoints; and the documented ATD views are useful for stakeholders to understand the ATD in the software project.

Keywords: architectural technical debt, architecture decision, architecture viewpoint, architectural technical debt documentation

1 Introduction

In recent years, there has been an increasing interest in technical debt (TD) in the software engineering community by both practitioners and researchers [1]. TD is a metaphor, coined by Ward Cunningham in 1992 "*for the tradeoff between writing clean code at higher cost and delayed delivery, and writing messy code cheap and fast at the cost of higher maintenance effort once it is shipped* [2, 3]". This metaphor was initially concerned with source code development. Currently, the concept of TD has been extended to the whole software lifecycle, such as software architecture (SA), detailed design, and testing [4, 5].

As Allman pointed out, "*TD is inevitable since the team almost never has a full grasp of the totality of the problem when a project starts* [6]". Thus, it is more realistic to manage TD rather than try to eliminate it completely. Furthermore, in some cases, TD is intentionally incurred to achieve some business advantages by sacrificing certain technical aspects such as sound modularity and encapsulation. This way, TD is not necessarily a "bad thing" if we have full knowledge of the consequences of the TD.

At the architectural level, architectural technical debt (ATD) is caused by architecture decisions that consciously or unconsciously compromise system quality attributes (QAs), particularly maintainability and evolvability [7, 8]. Like all other types of TD, managing ATD is of great essence. Especially, given the fundamental influence of SA in software development, it is of paramount importance to manage ATD, in order to achieve a high-quality SA especially in terms of its maintainability and evolvability.

To facilitate ATD management (ATDM), ATD needs to be documented, so that it becomes explicit and visible to involved stakeholders. If ATD is not documented, architecture decision making is very likely to ignore it and its impact on candidate decisions. Consequently, undocumented ATD items will keep collecting interest (i.e., effort required to fix the corresponding design issues), leading to a prohibitive cost in system maintenance and evolution. To the best of our knowledge, there are no approaches for systematically documenting ATD.

To facilitate the documentation of ATD, we propose to adopt the architecture documentation approach mandated by ISO/IEC/IEEE 42010 [9], which is based on architecture viewpoints. ISO/IEC/IEEE 42010 is an international standard, which defines requirements on the description of system, software and enterprise architectures. ISO/IEC/IEEE 42010 suggests identifying the stakeholders of a system and subsequently eliciting their concerns, so that appropriate viewpoints can be found or constructed to frame those concerns.

To define architecture viewpoints related to ATD (ATD viewpoints in short), we identified a number of stakeholders that are involved in ATDM and the typical concerns of those stakeholders. The identified stakeholders and their concerns were collected during our previous mapping study on TD [1]. Since the concerns are related to different aspects of ATD and cannot be framed by a single ATD viewpoint, we propose six ATD viewpoints, each of which frames a number of concerns related to ATD. This is in line with the guidelines of ISO/IEC/IEEE 42010 [9]. Note that, the verb *frame* used in this chapter has the same meaning as in ISO/IEC/IEEE 42010 standard, where "*frame* is used in its ordinary language sense: to formulate or construct in a particular style or language; to enclose in or as if in a frame; to surround so as to create a sharp or attractive image [9]."

We briefly outline the six viewpoints. First, the ATD Detail viewpoint provides detailed information of ATD items that are incurred by architecture decisions that compromise system evolvability or maintainability [7]. Second, the ATD Decision viewpoint deals with the relationship between architecture decisions and ATD items, showing which ATD items were incurred or repaid by which architecture decisions. Third, the ATD Stakeholder Involvement viewpoint addresses the responsibilities of stakeholders in ATDM during the architecting process, showing who took what actions on the ATD items during the current architecture iteration. Fourth, the ATD Distribution viewpoint deals with the distribution of the amount of the ATD over ATD items of a software system and the change of the ATD amount between milestones. Fifth, the ATD-related Component viewpoint deals with the relationship between system components and ATD items. Last, the ATD Chronological viewpoint addresses the evolution of ATD items across time.

To validate the effectiveness of the proposed ATD viewpoints in a real-life environment, we carried out a case study in which the ATD viewpoints are used to document ATD in an industrial project. The case is an information system in a large telecommunications company. The system mainly analyzes test data in various formats of telecommunications equipment and generates test reports about the quality of the tested equipment. The results of this case study show that the documented ATD views can effectively facilitate the documentation of ATD. Specifically, the ATD viewpoints are relatively easy to understand; it takes an acceptable amount of effort to document ATD using the ATD viewpoints; and the documented ATD views are useful for stakeholders to understand the ATD in the software project.

The main contributions of this chapter are threefold. First, we identified a set of stakeholders and their concerns on ATD, building on the results of our recent systematic mapping study. Second, six architecture viewpoints were proposed to address stakeholders' concerns on ATD. Third, we provide evidence from an industrial case study regarding the effectiveness of the proposed ATD viewpoints in documenting TD.

The rest of this chapter is organized as follows: Section 2 introduces the background and related work on ATD and its management as well as TD documentation; Section 3 presents the typical stakeholders involved in the ATDM process and their concerns regarding ATD; Section 4 describes the proposed ATD viewpoints including an example view for each of the viewpoints; Section 5 presents a case study which evaluates the effectiveness of the proposed ATD viewpoints in an industrial software project; and Section 6 concludes this chapter with future research directions.

2 Background and related work

In this section, we elaborate the concept of ATD, and then examine the related work on TD documentation.

2.1 Architectural technical debt

TD is essentially invisible to end users: they are not aware of the existence of TD when they are using a software system that delivers on its features. Conceptually, TD concerns the technical gaps between the current solutions and the ideal solutions, which may have negative influence on the system quality, especially maintainability and evolvability [10]. ATD is a type of TD at the architecture level [1]. It is mainly incurred by architecture decisions that result in immature architectural artifacts that compromise maintainability and evolvability of a software system. In contrast, code-level TD is concerned with the quality of the code and is usually incurred by the poor structure of the code and non-compliance with coding rules as well as violations of coding best practices (i.e., bad code smells).

Maintainability and evolvability are the two main system QAs that are compromised when ATD is incurred. Maintainability is defined in the ISO/IEC 25010 standard [11], in which quality models for systems and software are defined. According to ISO/IEC 25010, maintainability includes the following sub-characteristics (i.e., QAs): modularity, reusability, analyzability, modifiability, and testability. Evolvability is not defined in either ISO 9126 or ISO/IEC 25010. We define software evolvability as the ease of adding new functional and non-functional requirements. Typical ATD includes violations of best practices, or the consistency and integrity of software architectures, or the adoption of immature architecture techniques (e.g., architecture frameworks). A concrete example is the creation of architecture dependencies that violate the strict layered architectural pattern, i.e., a higher layer having direct dependencies to layers other than the one directly below it; this compromises modularity, a sub-characteristic of maintainability. Another example of ATD is the adoption of Microsoft .NET 2.0 as running environment for a software system, which would hinder the implementation of new features that are well supported by an updated .NET version (e.g., .NET 4.5); thus, this compromises evolvability. In summary, ATD essentially results from the compromise of modularity, reusability, analyzability, modifiability, testability, or evolvability during architecting.

As Steve McConnell pointed out, TD is classified in two basic types: the TD that is incurred unintentionally and the TD that is incurred intentionally [12]. Accordingly, ATD can be classified into intentional and unintentional ATD. The former ATD is incurred by strategic compromises of maintainability and evolvability in architecture decision making. The latter can be incurred by poor architecture decisions during architecting or violations of architecture rules and conformance during detailed design and coding. Both types of ATD need to be managed in the software lifecycle [7].

ATD can be seen as an important type of risk for a software system in the long term, but ATD is often ignored by the architecture and management teams. The main reason is that ATD is concerned with the cost of the long-term maintenance and evolution of a software system instead of the short-term business value that can be easily observed. However, ATD cannot be ignored forever; as the ATD in a software system accumulates incrementally, sooner or later problems will arise: maintenance tasks become hard to conduct, new features become difficult to introduce, system QAs are challenging to meet, etc.

2.2 Technical debt documentation

Not every type of TD needs to be documented. For instance, the code-level TD that can be automatically detected and measured by tools, does not necessarily have to be documented, since we can monitor the change of this type of TD by running the supporting tools. In contrast, the TD that cannot be automatically identified by tools needs to be systematically documented by other means; if not documented, this type of TD tends to be ignored by related stakeholders and, thus, it becomes invisible and cannot be managed. Most ATD is very difficult to identify and measure, as this cannot be automated. Therefore, once identified, this kind of ATD should be documented for further management.

There is little work on TD documentation. In our recent mapping study on TD [1], we only found four studies ([13-16]) that proposed to use TD items to represent and document TD. A TD item is a unit of TD in a software system. An example TD item is a "God" class with information about its location, estimated cost and benefit, responsible developer, intentionality, and TD type (design TD in this case). The TD in a software system is comprised of multiple TD items. The four aforementioned approaches ([13-16]) provided their own templates to document single TD items. All four TD item templates contain the following common fields: ID, location, responsible developer, TD type, and description [1]. Furthermore, each template also contains part of the following fields: principal, interest, interest probability, interest standard deviation, name, context, intentionality, correlation with other TD items, and propagation rules [1]. The last two fields (correlations and propagation rules) deserve further attention as they are helpful in analyzing the impact of TD items. In [14], Guo and Seaman proposed to record the correlations between TD items, but they did not specify the kinds of correlations between two TD items (of the same type or different types). In [13], Holvitie and Leppänen proposed to document so-called "propagation rules", which refer to implementation parts (e.g., packages, classes, and methods) that propagate TD. We consider that the propagation rules are important for managing TD since this information can be helpful in measuring TD and coming up with solutions to resolve TD.

The approaches proposed in the four aforementioned studies fall short in a number of ways compared with the approach proposed in this chapter. First, none of those four studies systematically extracted stakeholders' concerns on TD; therefore, there is no evidence that the documented TD items using those approaches (i.e., TD item templates) cover all necessary information interesting to related stakeholders. Second, all those approaches document individual TD items without showing the relationships between TD items, the holistic view of all TD items, and the evolution of the TD. Third, none of those TD item templates is dedicated to documenting TD at the architecture level (ATD). To the best of our knowledge, the only dedicated work on documenting ATD is the template for recording ATD items proposed in our previous work [7]. This ATD item template was adapted in the ATD Detail viewpoint in this chapter (see Table 4).

3 Typical stakeholders and concerns

We provide definitions of four core concepts used in this chapter before going into the details of stakeholders and concerns for the ATD viewpoints. These definitions are adopted as is from ISO/IEC/IEEE 42010 [9]:

- **Stakeholder:** "individual, team, organization, or classes thereof, having an interest in a system [9]."
- **Concern**: "interest in a system relevant to one or more of its stakeholders [9]."
- **Architecture view**: "work product expressing the architecture of a system from the perspective of specific system concerns [9]."
- **Architecture viewpoint**: "work product establishing the conventions for the construction, interpretation, and use of architecture views to frame specific system concerns [9]."

We identified a number of stakeholders that have interests in ATD and the typical concerns of those stakeholders. The identified stakeholders and their concerns were collected during our recent mapping study on TD [1] (see Section 3.2), in which we analyzed all available peer-reviewed scientific papers on TD. These stakeholders and their concerns are described in Sections 3.1 and 3.2, respectively. The ATD viewpoints are presented in Section 4.

3.1 ATD stakeholders

ATD stakeholders are those who perform ATDM activities, and who are directly affected by the consequences of ATD. The ATDM process includes five main activities: ATD identification, measurement, prioritization, monitoring, and repayment [7]. Architects, the development team, and architecture evaluators perform ATDM activities, such as ATD identification and ATD repayment. Project managers, customers, the development team, and architects are directly influenced by the consequences of ATD. The ATD stakeholders are described in detail as follows:

- **Architects** are concerned with all aspects of ATD incurred by architecture decisions. They are responsible to manage ATD explicitly and effectively to keep the architecture healthy enough. They perform all the five aforementioned ATDM activities in the ATDM process [7].
- **Architecture evaluators** take the ATD incurred by architecture decisions into account to assess the impact of the ATD on the quality of architecture. They can consider the known ATD as input and identify the existing but yet-unknown ATD as part of output during architecture evaluation. They conduct the ATD identification, measurement, and prioritization in the ATDM process [7].
- **Project managers** are mainly concerned with the consequences of the ATD which may cause a delayed release, changed release plan, or decreased quality of the product in the end. They are also concerned with assigning appropriate development team members to addressing different pieces of ATD. They are involved in ATD prioritization in the ATDM process [7].
- **Development team** is concerned with the cost of ATD in terms of the maintenance and evolution effort to a project. Development team members mainly include requirements engineers, designers, developers, maintainers, and testers. They are involved in ATD identification, measurement, and repayment [7].
- **Customers** are concerned with the impact on software product quality, the total cost of repaying ATD, and the time to market of new releases.

3.2 Concerns on ATD

We came up with the concerns on ATD in the following two ways: (1) concerns derived or adapted from generic concerns on TD that were identified during our mapping study on TD [1]; (2) the concerns derived from the ATDM activities in the ATDM process proposed in our previous work [7]. The ATD concerns are listed in Table 1. The details on how we came up with the ATD concerns are described in Appendix A.

Most of the ATD concerns are self-explanatory and, thus, we only describe two concerns in more detail: The concerns C16 and C17 are about the change rates of ATD benefit and cost, which are defined as the increased or decreased ATD benefit and cost in current iteration compared with the previous iteration. The proposed ATD viewpoints frame all the identified concerns. One concern can be framed by multiple ATD viewpoints, e.g., concerns C12 and C13 are framed by both the ATD Detail viewpoint and the ATD Decision viewpoint. The ATD viewpoints addressing each ATD concern are presented in Table 1. An "X" denotes that the viewpoint in the corresponding column addresses the concern in the corresponding row.

Table 1. Concerns related to ATD and their corresponding viewpoints

ID	Concern Description	ATD Detail viewpoint	ATD Decision viewpoint	ATD-related Component viewpoint	ATD Distribution viewpoint	ATD Stakeholder Involvement viewpoint	ATD Chronological viewpoint
C1	What ATD items have been incurred?	X					
C2	How much ATD does a software system have?				X		
C3	How much is the benefit of ATD item *A*?	X			X		
C4	How much is the cost of ATD item *A*?	X			X		
C5	How much is the interest of ATD item *A*?	X					
C6	What is the priority of ATD item *A* to be repaid?	X					
C7	What is the impact of ATD item *A* on software quality?	X					
C8	Which stakeholders were involved in ATD item *A*?	X				X	
C9	What ATD items affect stakeholder *SH*?	X				X	
C10	Which elements in the architecture design does ATD item *A* relate to?	X		X			
C11	What is the rationale for incurring ATD item *A*?	X					

C12	What is the architecture decision that incurs ATD item *A*?	X	X				
C13	What architecture decision(s) are made to repay ATD item *A*?	X	X				
C14	When does ATD item *A* change?	X					X
C15	When should ATD item *A* be repaid?	X					
C16	How fast is the total ATD benefit and cost of a software system changing?				X		
C17	How fast are the benefit and cost of ATD item *A* changing?				X		X
C18	What ATD items have changed since Iteration *I*?				X		
C19	What change scenarios are impacted by ATD item *A*?	X					
C20	How does an ATD item *A* propagate and accumulate in development?	X			X		
C21	Is ATD in a software system under acceptable level?				X		

We assign the ATD concerns to different types of stakeholders according to their roles. Table 2 shows the stakeholders of the ATD viewpoints and their concerns. Architects are concerned with all aspects of the ATD in a software system because architects need to have full knowledge of an architecture. Architecture evaluators are concerned with the aspects that are related to the architecture rationale, how the architecture satisfies the requirements of a project, and what the risks on the architecture quality are. Project managers are concerned with the aspects that are related to project management, such as cost of software maintenance and evolution, risks on software quality, and human resources management within the project. The development team pays more attention to the effort and cost of maintenance and evolution activities. The customers hold the concerns related to the cost, quality, and delivery time of products.

Table 2. Stakeholders of ATD viewpoints and their concerns

Stakeholders	**Concerns**
Architects	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21
Architecture evaluators	C1, C2, C3, C4, C5, C6, C7, C10, C11, C12, C13, C16, C17, C18, C20, C21
Project managers	C2, C6, C8, C9, C15, C16, C17, C18, C20, C21
Development team	C4, C5, C8, C9, C10
Customers	C2, C16

4 ATD viewpoints

We developed a set of ATD viewpoints, each framing part of the concerns listed in Table 1. Each ATD viewpoint frames one or more concerns and a concern can be framed by more than one ATD viewpoints. These ATD viewpoints were constructed in an iterative process driven by the stakeholder concerns on ATD. The construction of these viewpoints was also inspired by our previous work [17], where we provide a set of architecture viewpoints for documenting architecture decisions. We describe the ATD viewpoints following the template for documenting architecture viewpoints provided by ISO/IEC/IEEE 42010 [9]. The template suggests to document an architecture viewpoint in multiple parts. We present the following parts for each ATD viewpoint in each subsection: the name, an overview, the typical stakeholders and their concerns, as well as an example view conforming to the ATD viewpoint. The model kinds and correspondence rules for the ATD viewpoints will be detailed in Appendix B to ensure the readability of the current section. In Appendix B, the definition of each ATD viewpoint is presented in a subsection; these definitions can act as guidelines to create views conforming to the viewpoint.

4.1 ATD Detail viewpoint

ATD Detail viewpoint presents the detailed information of individual ATD items in a software system. The stakeholders and concerns of this viewpoint are shown in Table 3. These concerns center mainly around the properties of ATD items, including the cost, benefit, rationale, related change scenarios, and so forth.

Table 3. Typical stakeholders of the ATD Detail viewpoint and their concerns

Stakeholders	Concerns
Architects	C1, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C19, C20
Architecture evaluators	C1, C3, C4, C5, C6, C7, C10, C11, C12, C13
Project managers	C6, C15
Development team	C4, C5, C9, C10

Table 4. Template for documenting an ATD item (adapted from [7])

ID	A unique identification number of the ATD item that serves as a key in other views.
Name	A short name of this ATD item that indicates the essence of this ATD item.
Version	The current version number of the ATD item (e.g., 5).
Date	The date when this ATD item was identified or updated.
Status	The current status of the ATD item. The types of status are described in detail in Appendix B.5.
Priority	The priority of this ATD item to be repaid if this ATD item is unresolved. The priority is a positive natural number between 1 and 10. A larger number indicates a higher priority.
Intentionality	The ATD item can be incurred intentionally or unintentionally.
Incurred by	The architecture decision that incurs this ATD item. ATD can be incurred by architecture decisions made by architects, or by designers and developers not conforming to those architecture decisions.
Repaid by	The architecture decisions that repays this ATD item.
Responsible	The person or team who is responsible for managing this ATD item.
Compromised QA	The QA(s) that are compromised (modularity, reusability, analyzability, modifiability, testability, or evolvability).
Rationale	The reason why the ATD item was incurred
Benefit	The value gained if the ATD item remains unresolved. The benefit is comprised of two parts: (1) **Measureable benefit** that can be measured in development effort (e.g., person-days). (2) **QA benefit** that cannot be transferred into effort. We can estimate the benefit level of each beneficiary QA.
Cost	The cost suffered by incurring this ATD item, which is the sum of principal and interest described below.
Principal	The cost if this ATD item is resolved at the time when the ATD item is identified.
Interest	The interest that this ATD item accumulates (the interest is calculated based on the predicted change scenarios described below).
Change scenarios	Calculation of the interest of this ATD item (total interest) = $\sum_{k=1}^{n} I_k \times P_k$
Architecture diagram	A diagram or model that illustrates the concerned part in the architecture design
History	Change history of this ATD item

Change scenarios

#	Scenario description	Consequence	Scenario interest	Probability
1	Scenario 1	consequence of scenario 1	I_1	P_1
2	Scenario 2	consequence of scenario 2	I_2	P_2
…	…	…	…	…
n	Scenario n	consequence of scenario n	I_n	P_n

History

Stakeholder	Action	Status	Iteration	Date
Name <Stakeholder role>	Action that the stakeholder performed on the ATD item	Status when the action was completed	Iteration endpoint name	When the action was performed

We codify the details of an ATD item using a template (see Table 4), which is an adaptation based on the ATD item template proposed in [7]. A view conforming to the ATD Detail viewpoint is comprised of multiple ATD items, and each is described using the template. Each element of an ATD item has an associated description as listed in Table 4. Compared with the ATD template used in [7], we add new elements "Priority", "Intentionality" as well as "Repaid by", refine the candidate status set of the "Status" element, and revise the element "History". The status "unresolved" in the "Status" element in [7] is further refined to "identified", "measured", "re-measured", and prioritized. The "History" element of an

ATD item includes five sub-elements: a **Stakeholder** who performs an **Action** on this ATD item, causing it to have a specific **Status**, on a specific **Date** that is in the period of a certain development **Iteration**. The aforementioned ‘action’ can be *identify*, *measure*, *re-measure*, *prioritize*, and *repay*, and accordingly a ‘status’ can be *identified*, *measured*, *re-measured*, *prioritized*, and *resolved*. An example documented ATD item following the ATD Detail viewpoint is shown in Table 5.

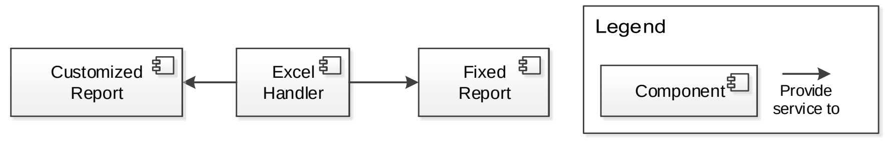
Table 5. Example ATD detail model of an ATD item

ID	ATD1
Name	Poor support for report format and style customization
Version	4
Date	30-09-2013
Status	Resolved
Priority	9 (out of 10)
Intentionality	Intentional
Incurred by	Architecture decision 10 (AD10): using pre-defined Excel templates for product quality reports
Repaid by	Architecture decision 25 (AD25): replacing pre-defined Excel templates with Excel automation
Responsible	Hui
Compromised QA	Evolvability
Rationale	To speed up the implementation of the feature of product quality reports, we decided to use the pre-defined Excel templates instead of Excel automation to set the formats and styles of the report files, since we did not have experience in Excel automation development. We saved 15 person-days.
Benefit	15 person-days
Cost	32.8 person-days
Principal	25 person-days
Interest	7.8 person-days

Change scenarios

#	Scenario description	Consequence	Scenario interest	Prob.
S10	Add a new report type for product line A	Manually add a new type of report template and test it for product line A	3 person-day	0.8
S11	Add new a product model for product line B	Manually update and test all the existing report templates	1 person-day	0.9
S13	Add new a product line	Manually add and test all types of report templates for the new product line	5 person-day	0.9

Architecture diagram



History

Stakeholder	Action	Status	Iteration	Date
Architect1 <<Architect>> Developer5 <<Developer>>	Identify	Identified	Release 16.0	05-08-2014
Architect1 <<Architect>> Developer5 <<Developer>>	Measure	Measured	Release 16.1	22-08-2014
Architect1 <<Architect>>	Prioritize	Prioritized	Release 16.1	22-08-2014
Developer5 <<Developer>>	Repay	Resolved	Release 16.2	16-09-2014

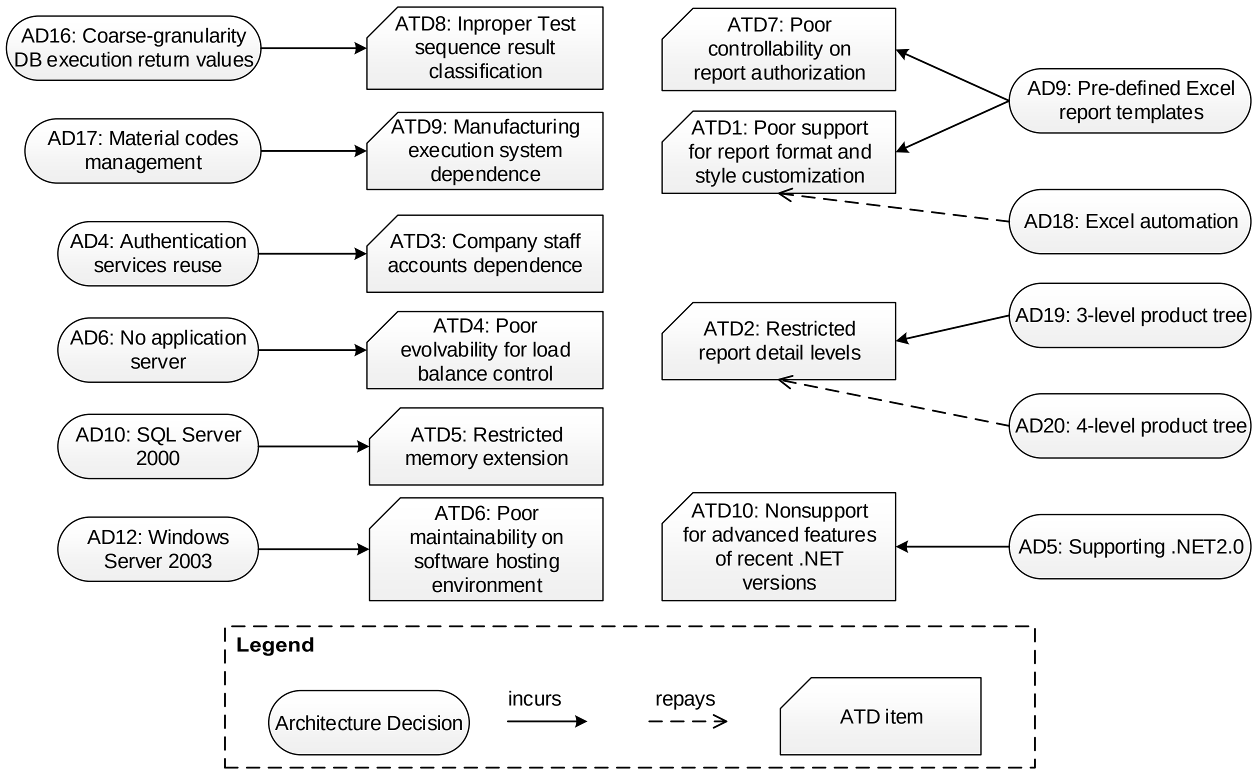
4.2 ATD Decision viewpoint

Architecture decisions are treated as first-class entities of architectures and play an essential role in architecture design [9, 18]. ATD can be incurred by architects, designers, and developers, while all of them can do this intentionally or unintentionally. Architecture decisions made during architecting entail compromises and trade-offs made by architects, potentially together with involved stakeholders during architecture design. Architects usually have to make compromises on technical solutions to meet the business needs such as release deadline or saving short-term cost. ATD is part of the result of such compromises. In addition, new architecture decisions are continuously made to repay existing ATD. Therefore, ATD can be managed based on architecture decisions [7].

The ATD Decision viewpoint describes which architecture decisions have incurred ATD items and which architecture decisions are made to repay ATD items. The typical stakeholders of ATD Decision viewpoint and their addressed concerns related to ATD are listed in Table 6. The details of the ATD Decision viewpoint are described in Appendix B.2. Fig. 1 shows a fragment of an example ATD Decision view.

Table 6. Typical stakeholders of the ATD Decision viewpoint and their concerns

Stakeholders	**Concerns**
Architects	C12, C13
Architecture evaluators	C12, C13


AD16: Coarse-granularity DB execution return values
ATD8: Inproper Test sequence result classification
ATD7: Poor controllability on report authorization
AD9: Pre-defined Excel report templates
AD17: Material codes management
ATD9: Manufacturing execution system dependence
ATD1: Poor support for report format and style customization
AD18: Excel automation
AD4: Authentication services reuse
ATD3: Company staff accounts dependence
AD6: No application server
ATD4: Poor evolvability for load balance control
ATD2: Restricted report detail levels
AD19: 3-level product tree
AD20: 4-level product tree
AD10: SQL Server 2000
ATD5: Restricted memory extension
AD12: Windows Server 2003
ATD6: Poor maintainability on software hosting environment
ATD10: Nonsupport for advanced features of recent .NET versions
AD5: Supporting .NET2.0
Legend
Architecture Decision
incurs
repays
ATD item
Fig. 1. Fragment of an example ATD Decision view

4.3 ATD-related Component viewpoint

This viewpoint illustrates bi-directional relations between architecture components and unresolved ATD items. By “ATD item *A* relates to component *Comp*”, we mean that component *Comp* needs to be modified to repay ATD item *A*. Typical stakeholders of the ATD-related Component viewpoint and their concerns are depicted in Table 7. A fragment of an example ATD-related Component view is shown in Table 8, in which an “X” refers that the ATD item in the corresponding row relates to the component in the corresponding column. Note that, due to the limited space, we do not show the names of the ATD items, which practitioners should provide in real cases. The names of the ATD items can be found in the ATD Decision view shown in Fig. 1.

Table 7. Typical stakeholders of the ATD-related Component viewpoint and their concerns

Stakeholders	**Concerns**
Architects	C10
Evaluators	C10
Development team	C10

Table 8. Fragment of an example ATD-related Component view

ATD item	Comp1: DB Handler	Comp2: General Query	Comp3: Product Test Controller	Comp4: User Management	Comp5: Customized Report	Comp6: Fixed Report	Comp7: Excel Handler	Comp8: Quality Analysis Algorithm	Comp9: Test Result File Management	Comp10: FTP Tool	Comp11: Client UI	Comp12: Fixed Report Generator UI	Comp13: SQL Server	Comp14: FTP Server	#(components)
ATD1					X	X	X								3
ATD2	X	X			X	X		X							5
ATD3				X									X		2
ATD4	X	X			X										3
ATD5	X	X			X	X									4
ATD6	X	X													2
ATD7		X			X	X	X								4
ATD8	X	X	X			X	X	X							6
ATD9			X												1
ATD10		X	X		X			X			X	X			6
#(ATD items)	5	7	3	1	7	5	3	3	0	0	1	1	1	0	

4.4 ATD Distribution viewpoint

The ATD Distribution viewpoint shows how the amount of ATD cost and benefit (see ATD Detail viewpoint) distributes over each ATD item and how the amount of total ATD cost and benefit changes in a software system during development. With this viewpoint, we can easily understand the change of the accumulated ATD of a software system and the cost variation of each ATD item during two iterations. The typical stakeholders of this viewpoint and their concerns framed by this viewpoint are shown in Table 9. These concerns are mainly about the benefits, costs, and their changes of the ATD items in a software system. Fig. 2 shows a fragment of an example ATD Distribution view. The ATD items in Fig. 2 are those from Fig. 1. In this example view, we can see that: ATD items ATD1 and ATD2 are completely repaid at *Release V16.1*; ATD item ATD10 is identified at *Release V16.2*; ATD item ATD4 has the highest amount of ATD cost in *Release V16.1* and *Release V16.2*; and the amount of accumulated ATD of this project has decreased since *Release V16.1*. In an ATD Distribution view, only measurable benefit of each ATD item is shown, while the QA benefit is not. The threshold line in Fig. 2 denotes how much ATD can be tolerated in a software system. The threshold is defined by the project manager and the customer, taking into account the project budget, release planning, labor, project size, and other related factors.

Table 9. Typical stakeholders of the ATD Distribution viewpoint and their concerns

Stakeholders	Concerns
Architects	C2, C3, C4, C16, C17, C18, C20, C21
Evaluators	C2, C3, C4, C16, C17, C18, C20, C21
Project managers	C2, C16, C17, C18, C20, C21
Customers	C2, C16

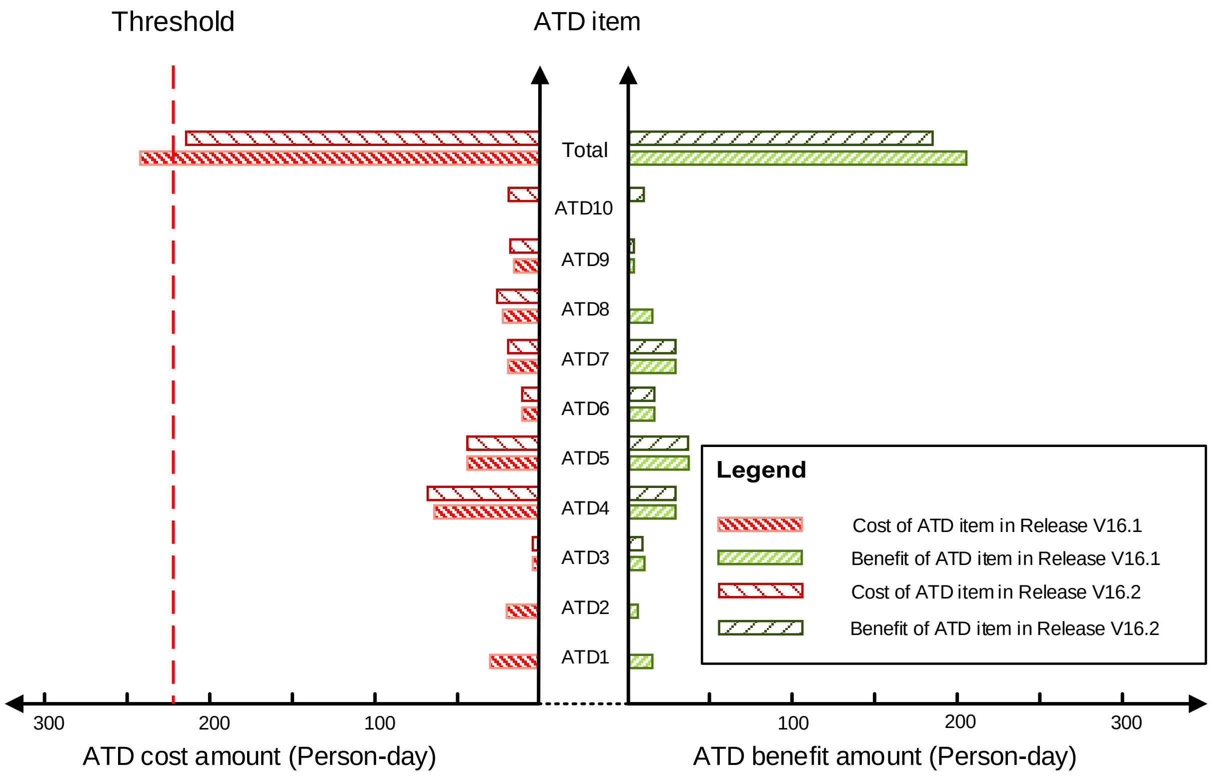


Fig. 2. Fragment of an example ATD Distribution view

4.5 ATD Stakeholder Involvement viewpoint

The ATD Stakeholder Involvement viewpoint describes the responsibilities of the involved stakeholders regarding the managed ATD items. Views governed by this viewpoint show ATD items, actions, and stakeholders involved in the ATDM process within one specific iteration. Table 10 shows the typical stakeholders of the ATD Stakeholder Involvement viewpoint and their concerns framed by it. The stakeholders of this viewpoint include technical ones (e.g., architects) that participate in the management of ATD, and project managers who are concerned with the human resources assigned to ATD items. Fig. 3 depicts an example ATD Stakeholder Involvement view.

Table 10. Typical stakeholders of the ATD Stakeholder Involvement viewpoint and their concerns

Stakeholders	Concerns
Architects	C8, C9
Project manager	C8, C9
Development team	C8, C9

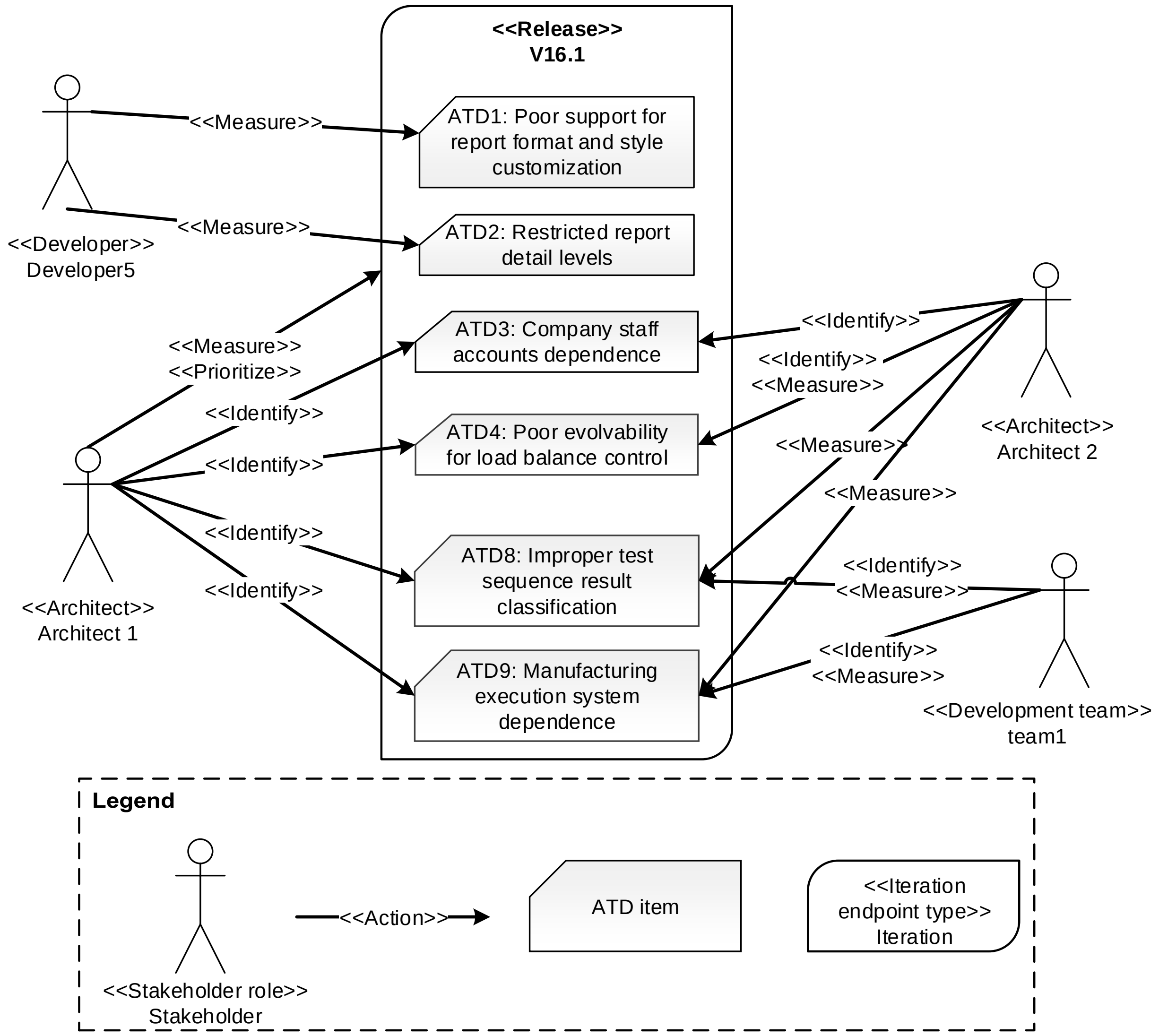


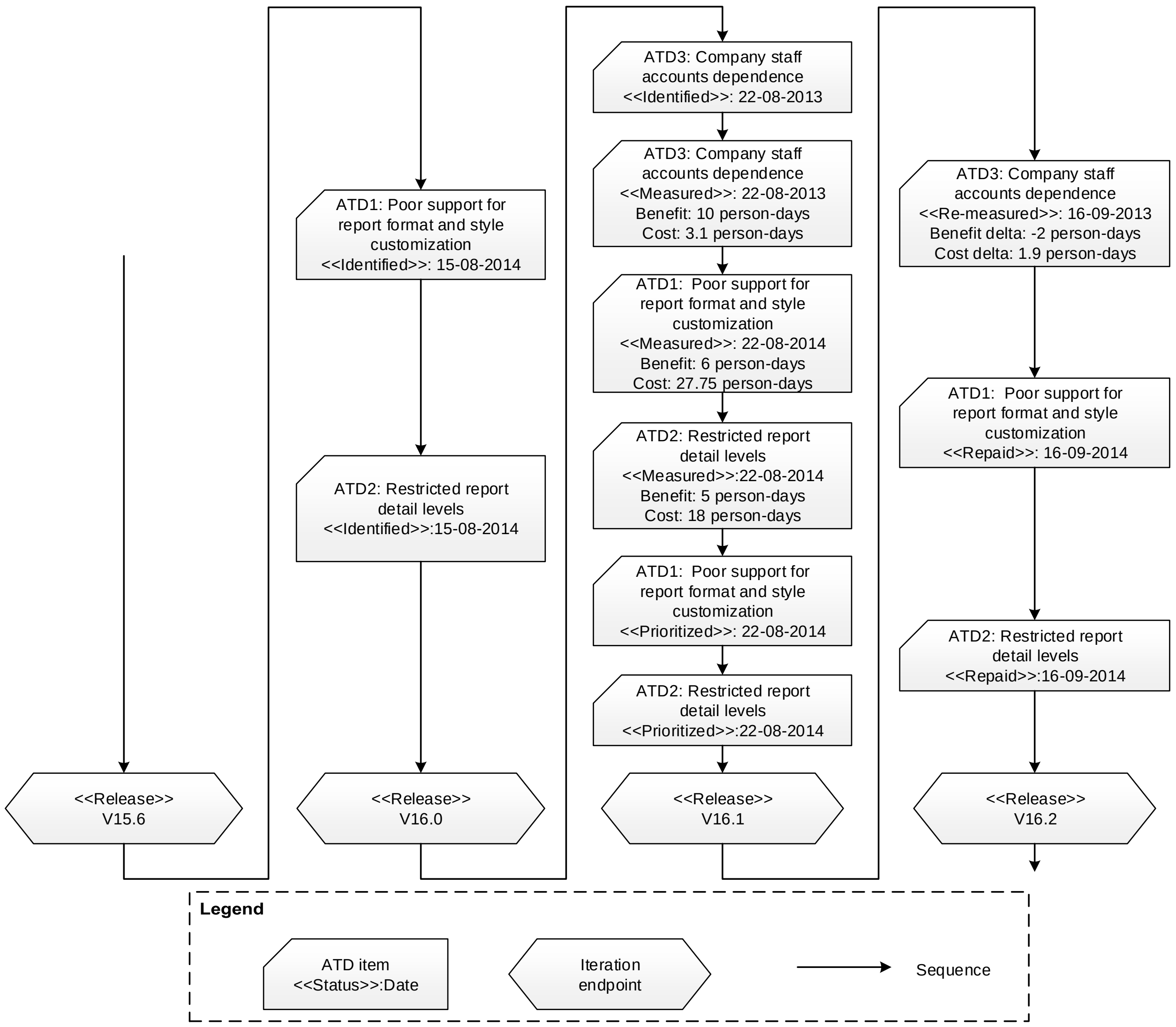
Fig. 3. Fragment of an example ATD Stakeholder Involvement view

4.6 ATD Chronological viewpoint

This viewpoint focuses on the change of the ATD items in a software system over time. From this viewpoint, we can see how ATD is managed along the timeline, i.e., what ATD items are dealt with in each iteration and how each ATD item is handled over time. This viewpoint also shows the benefit and cost of the measured ATD item, and the benefit delta and cost delta of the re-measured ATD item. Typical stakeholders of the ATD Chronological viewpoint and their concerns are shown in Table 11. A fragment of example ATD Chronological view is depicted in Fig. 4.

Table 11. Typical stakeholders of the ATD Chronological viewpoint and their concerns

Stakeholders	Concerns
Architects	C14, C17
Project managers	C17



ATD3: Company staff accounts dependence <<Identified>>: 22-08-2013
ATD3: Company staff accounts dependence <<Measured>>: 22-08-2013 Benefit: 10 person-days Cost: 3.1 person-days
ATD3: Company staff accounts dependence <<Re-measured>>: 16-09-2013 Benefit delta: -2 person-days Cost delta: 1.9 person-days
ATD1: Poor support for report format and style customization <<Identified>>: 15-08-2014
ATD1: Poor support for report format and style customization <<Measured>>: 22-08-2014 Benefit: 6 person-days Cost: 27.75 person-days
ATD1: Poor support for report format and style customization <<Repaid>>: 16-09-2014
ATD2: Restricted report detail levels <<Measured>>:22-08-2014 Benefit: 5 person-days Cost: 18 person-days
ATD2: Restricted report detail levels <<Identified>>:15-08-2014
ATD1: Poor support for report format and style customization <<Prioritized>>: 22-08-2014
ATD2: Restricted report detail levels <<Repaid>>:16-09-2014
ATD2: Restricted report detail levels <<Prioritized>>:22-08-2014
<<Release>> V15.6
<<Release>> V16.0
<<Release>> V16.1
<<Release>> V16.2
Legend
ATD item <<Status>>:Date
Iteration endpoint
Sequence

Fig. 4. Fragment of an example ATD Chronological view

5 Case study

To validate the effectiveness of the proposed ATD viewpoints in a real-life environment, we carried out a case study in which the ATD viewpoints were used to document ATD in an industrial project. We designed and reported the case study following the guidelines proposed by Runeson and Höst [19]. However, we have not included the section on data analysis suggested by the guidelines, since only descriptive statistics were used to analyze the data collected in the case study.

5.1 Study objective and research questions

The goal of this case study, described using the Goal-Question-Metric approach [20], is: *to analyze* ATD viewpoints *for the purpose of* evaluation *with respect to* their effectiveness in documenting ATD, *from the point of view of* ATD stakeholders *in the context of* industrial software development.

We define effectiveness in documenting ATD as being comprised of the following aspects:

- *Understandability of the ATD viewpoints*. The understandability of the ATD viewpoints themselves (e.g., typical stakeholders and framed concerns, model kinds, and correspondence rules) reflects to what extent the stakeholders can generate the corresponding ATD views efficiently and correctly. If the ATD viewpoints cannot be easily understood, they are not likely to be adopted for ATDM.

- *Effort for collecting necessary data and further producing ATD views*. How easy data collection is, affects the feasibility of using the ATD viewpoints in practice. If the data collection is too complicated and time-consuming, stakeholders would be reluctant to use the viewpoints. In addition, the effort it takes to document the ATD views with available information plays a major role in their adoption.
- *Usefulness in helping stakeholders to understand the ATD in software systems*. This aspect is concerned with whether the views conforming to the ATD viewpoints can enhance stakeholders' understanding on the current state of the ATD and is comprised of 3 parts: a) whether stakeholders perceive the actual health level of the SA compared to their pre-conception; b) which ATD views are useful to understand ATD; and c) which ATD views are promising to be adopted by the stakeholders both to produce and consume the views.

Accordingly, we ask three research questions (RQs), each corresponding to one aspect of effectiveness of the ATD viewpoints, respectively:

RQ1: How easy is it to understand the ATD viewpoints?

RQ2: How easy is it to collect the required information for generating ATD views governed by the ATD viewpoints and to document ATD views with the gathered information?

RQ3: Do ATD views effectively support stakeholders to understand the ATD?

5.2 Study execution

This case study was conducted to empirically evaluate how the proposed ATD viewpoints can effectively support stakeholders to document and understand ATD. This case study is evaluatory in nature since the case study aims at evaluating the effectiveness of the ATD viewpoints in an industrial environment.

5.2.1 Case description

The case is an information system in a large telecommunications company in China. The system analyzes the test data in various formats of telecommunications equipment and generates various types of reports about the quality of the tested telecommunications equipment. This system also provides the functionality of managing and controlling whether a piece of telecommunications equipment is allowed to proceed in tests.

The software project team includes a project manager, two architects, and nine development team members. The project manager, two architects, and six development team members participated in this case study; the remaining three developers were not available. The software system has a history of around seven years. Its size is about 760,000 lines of source code, and around 290 person-months (approximately 50,000 person-hours) has been invested in this project.

5.2.2 Data collection

5.2.2.1 Data to be collected

To answer the research questions (RQs) defined in Section 5.1, we collected the data items listed in Table 12, where the target RQ for each data item is listed. We also collected the participants' information on their experience in software industry (see Table 13) and the related information on the selected software project in this case study (see Table 14).

Table 12. Data items to be collected

#	**Data Item**	**Range**	**RQ**
D1	How easy it is for the participants to understand the ATD viewpoints	Ten-point Likert scale. One for extremely hard, ten for extremely easy.	RQ1
D2	How easy it is for the participants to collect the required information for generating the ATD views	Ten-point Likert scale. One for extremely hard, ten for extremely easy.	RQ2

D3	How much effort it needs to document the ATD views with gathered information	Four-point Likert scale: little, not too much, a little bit too much but acceptable, unacceptably too much	RQ2
D4	How different it is between the actual health level of the architecture and the health level that the participants considered it to be	Five-point Likert scale: much higher than, higher than, roughly equal to, lower than, and much lower than.	RQ3
D5	How useful each ATD viewpoint is in facilitating the understanding of ATD	Five-point Likert scale: not useful, somewhat useful, moderately useful, very useful, not sure.	RQ3
D6	Which ATD views the participants are willing to use to document ATD (produce information), and which views to use to maintain their knowledge about ATD (consume information) and subsequently manage ATD	n.a.	RQ3

Table 13. Information related to the study participants

#	Participant data item	Scale type	Unit	Range
PD1	Time the participants have worked in software industry	Ratio	Years	Positive natural numbers
PD2	Time the participants have worked as developers	Ratio	Years	Positive natural numbers
PD3	Time the participants have worked in the company	Ratio	Years	Positive natural numbers
PD4	Time the participants have worked in the domain that the case belongs to	Ratio	Years	Positive natural numbers
PD5	Time the participants have worked in the current company	Ratio	Years	Positive natural numbers
PD6	Time the participants have been involved in the current project	Ratio	Years	Positive natural numbers
PD7	Received dedicated training in SA	Nominal	n.a.	Yes or No
PD8	Experience level of the participants in SA	Ordinal	n.a.	Five-point Likert scale[1]

Table 14. Information related to the selected case

#	Case data item	Scale type	Unit	Range
CD1	The number of the architecture decisions for analysis	Ratio	Decisions	Positive natural numbers
CD2	The number of ATD items documented in the software project	Ratio	ATD items	Positive natural numbers
CD3	The number of change scenarios used to calculate the cost and benefit of ATD items	Ratio	Change scenarios	Positive natural numbers
CD4	Duration of the selected project in this case study	Ratio	Months	Positive natural numbers
CD5	Project effort	Ratio	Person-months	Positive natural numbers
CD6	Project size in lines of code	Ratio	Lines of code	Positive natural numbers

5.2.2.2 Data collection method

Interviews were the main method to collect data in this case study. As suggested in [19], interviews allow us to get in-depth knowledge about the topics of interest in the case study, by asking a series of

[1] The five-point Likert scale: a) No knowledge on SA, b) Some knowledge on SA but never involved in architecting, c) Experience in architecting small software systems (<=50,000 lines of code), d) Experience in architecting big software systems (>50,000 lines of code), and e) Chief architect of big software systems.

questions about the interview topic to the participants of the case study. We used semi-structured interviews in this case study, which allowed us to adjust the order of the planned questions according to the development of the conversation between the researcher and the participants. In addition, semi-structured interviews allowed us to explore in more depth the interview topics by asking follow-up questions based on the participants' answers. We interviewed all the nine participants with different sets of questions depending on each participant's role in the selected software project.

5.2.2.3 Data collection process

In order to answer the RQs presented in Section 5.1, we divide the case study into three parts (preparation, workshop, and interview) which include seven tasks (Task1-Task7), as described below (also see Fig. 5).

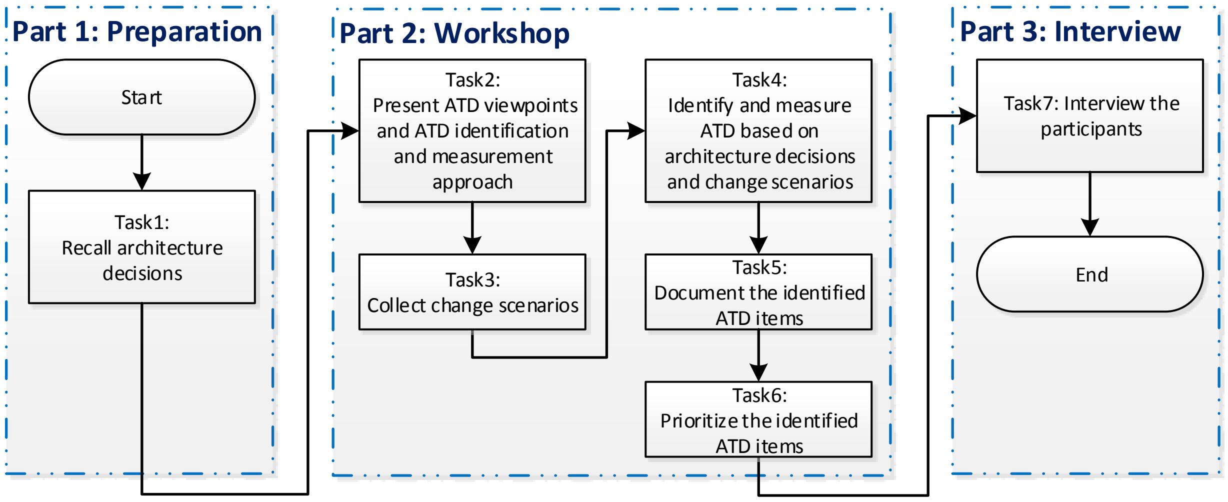

Part 1: Preparation
Start
Task1: Recall architecture decisions
Part 2: Workshop
Task2: Present ATD viewpoints and ATD identification and measurement approach
Task3: Collect change scenarios
Task4: Identify and measure ATD based on architecture decisions and change scenarios
Task5: Document the identified ATD items
Task6: Prioritize the identified ATD items
Part 3: Interview
Task7: Interview the participants
End

Fig. 5. Procedure of the case study

Part 1 - Preparation.

Task1: *Recall architecture decisions*. The architects recalled the architecture decisions of the software system following the guidelines provided by the authors, and documented the architecture decisions using a template provided by the authors.

Part 2 - Workshop.

Task2: *Present ATD viewpoints*. The first author presented the schedule of the workshop, the ATD viewpoints, and the change scenario template to the participants (i.e., the architect, manager, and development team)

Task3: *Collect change scenarios*. The project manager provided a list of change scenarios that may happen in the coming 3 months[2]. A change scenario describes a possible major change in a software system. Typical change scenarios include: (1) the unimplemented features that are planned in the roadmap of the software system, (2) the known but unresolved bugs, and (3) the maintenance tasks that improve certain QAs of the implemented architecture.

Task4: *Identify and measure ATD based on architecture decisions and change scenarios*. We provided guidelines on how to perform the identification and measurement of the ATD. All participants worked together on this task following the guidelines. The chief architect documented the identified ATD items using the ATD item template (i.e., the ATD Detail viewpoint).

[2] There are 3 builds every month, but whether a build will be released depends on the severity of the resolved bugs and the urgency of the new requirements.

Task5: *Document ATD items*. The chief architect documented the identified ATD items using the ATD Decision, ATD-related Component, and ATD Stakeholder Involvement viewpoints. He also improved the ATD Detail view created in Task4.
Task6: *Prioritize the identified ATD items*. The participants read the ATD views generated in tasks 4 and 5, and then prioritized the ATD items based on their understanding and the results of their discussions on the documented ATD views.
Part 3 - Interview.
Task7: *Interview the participants*. We first asked the participants to fill in a questionnaire regarding their experience in software industry (see Table 13). After that, one author interviewed the participants one by one using semi-structured questions.

This workshop in Part 2 took around 4 hours. The schedule of the workshop is described in Table 15. Each interview in Part 3 lasted between 45 and 65 minutes.

Table 15. Schedule of the workshop

#	Step	Participants	Time	%
1	Task2	All	40 minutes	18
2	Task3	All	10 minutes	4
3	Break	All	10 minutes	4
4	Task4	All	80 minutes	36
5	Task5	The chief architect	40 minutes	18
6	Task6	All	45 minutes	20
Total time			225 minutes	100

5.3 Results

We first present the collected information about the participants and the selected case (i.e., the software project) in this case study, then answer each of the research questions defined in Section 5.1, in the following sub-sections.

Table 16. Participants' experience information

Participant	PD1: Years in software industry	PD2: years as a developer	PD3: Years in the company	PD4: years in the domain	PD5: Years in the current role	PD6: Years in the project	PD7: Dedicated SA training	PD8: Experience level in SA
Architect1	8	8	8	6	6	6	Y	Chief architect of big software systems
Architect2	9	6	9	2	5	2	Y	Experience in architecting big software systems
Manager1	10	5	13	7	7	7	Y	Experience in architecting big software systems
Developer1	7	7	7	5	5	5	N	Some SA knowledge but never involved in architecting
Developer2	8	8	5	3	7	3	N	Experience in architecting small software systems
Developer3	9	9	4	2	1	1	Y	Experience in architecting small software systems
Developer4	9	6	6	3	3	3	N	Some SA knowledge but never involved in architecting
Developer5	3.5	3.5	3.5	1.5	3.5	1.5	Y	Experience in architecting big software systems
Tester1	7	7	6	0.1	0.1	0.1	N	Some SA knowledge but never involved in architecting

Table 16 shows the information on the participants' experience in software industry. All the participants have worked in IT industry for seven or more years except Developer5 who has 3.5-year experience in IT industry. Four participants have experience in architecting big software systems (which size is more than 50,000 lines of code); two have experience in architecting small software systems (which size is less than 50,000 lines of code); while the rest three have no experience in architecting, but they have knowledge on SA.

The selected software project in this case study is a relatively big project (see CD4, 5 and 6 below). The information about the case is described below:

- CD1, No. of the architecture decisions for analysis: 20.
- CD2, No. of documented ATD items: 10.
- CD3, No. of change scenarios used to calculate the cost and benefit of ATD items: 26.
- CD4, Duration of the software project: seven years.
- CD5, Project effort: around 290 person-months (about 50,000 person-hours).
- CD6, Project size in lines of code: around 760,000 lines of code.

5.3.1 Understandability of ATD viewpoints (RQ1)

The results of the understandability of the ATD viewpoints are described in Table 17. All ATD viewpoints received an average score above eight, except for the ATD Detail viewpoint. This indicates that the ATD viewpoints are relatively easy to understand. The ATD Detail viewpoint received an average score of 6.8, which indicates some issues in understanding it.

Table 17. Understandability of ATD viewpoints

ATD viewpoint	**Architect1**	**Architect2**	**Manager1**	**Developer1**	**Developer2**	**Developer3**	**Developer4**	**Developer5**	**Tester1**	**Mean score**	**Median score**
ATD Detail viewpoint	8	6	6	5	6	5	7	9	9	6.8	6
ATD Decision viewpoint	7	10	7	8	8	5	10	10	9	8.2	8
ATD-related Component viewpoint	8	10	6	9	10	9	10	9	10	9.0	9
ATD Distribution viewpoint	8	9	9	7	8	8	6	10	8	8.1	8
ATD Stakeholder Involvement viewpoint	8	10	5	9	8	8	9	9	10	8.4	9
ATD Chronological viewpoint	9	10	5	7	10	7	7	9	9	8.1	9

5.3.2 Ease of collecting the required information and documenting ATD views (RQ2)

The collection of required information was performed by all the participants in the workshop, while the documentation of ATD views was only performed by the chief architect. Table 18 shows the ease of collecting the needed information for creating the ATD views. A higher score (in the range between 1 and 10) means that the corresponding piece of information is easier to collect. The benefit, principal, interest, and interest probability received scores lower than 7. The compromised QA and affected components received the highest scores.

Table 18. Ease of collecting required information for ATD views

Required information	Architect1	Architect2	Manager1	Developer1	Developer2	Developer3	Developer4	Developer5	Tester1	Mean score	Median score
Architecture decision that incurs an ATD item	5	9	5	5	6	5	9	8	8	6.7	6
Architecture decision that repays an ATD item	7	8	7	7	6	5	9	8	6	7.0	7
Compromised QA	8	9	6	8	6	7	9	9	9	7.9	8
Rationale	7	8	7	7	5	5	9	9	9	7.3	7
Benefit	5	10	3	7	4	2	9	8	6	6.0	6
Principal	5	7	4	3	4	3	8	7	6	5.2	5
Change scenarios	6	9	7	3	8	4	9	8	9	7.0	8
Consequence of a change scenario	5	9	7	3	7	6	9	8	9	7.0	7
Potential interest incurred in a change scenario	6	9	4	4	4	2	3	7	8	5.2	4
Probability of the potential interest incurred in a change scenario	6	5	4	8	4	8	5	9	8	6.3	6
Affected components	6	9	5	8	8	6	10	9	9	7.8	8

The ATD views were documented by the chief architect. The chief architect documented the ATD items using the ATD Detail viewpoint along the ATD identification and measurement (i.e., Task4) which took 80 minutes. In addition, there was 40 minutes (in Task5) dedicated to ATD documentation. Considering that the chief architect only spent around one fourth of the 80 minutes in documenting ten ATD items in Task4, the total time for ATD documentation was around one hour (80*1/4+40=60minutes) in this case study. During the interview with him, he argued that documenting the ATD views needs an acceptable amount of effort, but this amount of effort was a little bit more than expected (i.e., a little bit too much but acceptable). Specifically, documenting the ATD Detail view requires increased effort, while generating other ATD views was comparatively much easier. He suggested that a dedicated tool supporting them to generate ATD views would make ATD documentation much easier, since the information in the ATD Detail view can be used to automatically generate the rest of the views.

5.3.3 Usefulness in understanding ATD (RQ3)

We investigated the usefulness in understanding ATD in the following three aspects: (1) the difference of the architecture health level of the current architecture compared with what they initially thought, (2) how useful the participants thought the ATD viewpoints to be in facilitating their understanding of ATD in the software system, and (3) ATD viewpoints that the participants are willing to use for managing ATD.

- **Architecture health level**. In the interviews, we asked the participants to compare the architecture health level based on the documented ATD views, with their initial assessment of the health level. As shown in Table 19, six participants argued that the architecture health level is lower than that they thought to be; one considered that the former is much lower than the latter; and two believed that the former is roughly equal to the latter.

Table 19. Architecture health level compared with the previously estimated

Participant	Architecture health level
Architect1	lower than
Architect2	much lower than
Manager1	lower than
Developer1	roughly equal to
Developer2	lower than
Developer3	lower than
Developer4	lower than
Developer5	lower than
Tester1	roughly equal to

- **Understanding ATD**. In our interviews to the participants, we asked them about how useful they perceived the ATD views to be in understanding ATD in the system. Table 20 shows the answers to this question. There are five candidate answers: *not useful*, *somewhat useful*, *moderately useful*, *very useful*, and *not sure*. Most of the participants considered that the ATD Detail view, ATD Decision view, ATD-related Component view, and ATD distribution view are *very useful* in understanding ATD in this case study.

Table 20. Usefulness of the ATD views in understanding ATD

Participant	ATD Detail view	ATD Decision View	ATD-related Component View	ATD Distribution View	ATD Stakeholder Involvement View	ATD Chronological View
Architect1	Moderately useful	Moderately useful	Very useful	Moderately useful	Moderately useful	Moderately useful
Architect2	Very useful	Very useful	Very useful	Very useful	Moderately useful	Very useful
Manager1	Moderately useful	Very useful	Moderately useful	Very useful	Somewhat useful	Moderately useful
Developer1	Very useful	Moderately useful	Very useful	Moderately useful	Somewhat useful	Moderately useful
Developer2	Very useful	Somewhat useful	Somewhat useful	Very useful	Somewhat useful	Somewhat useful
Developer3	Very useful	Very useful	Very useful	Very useful	Somewhat useful	Somewhat useful
Developer4	Very useful	Moderately useful	Moderately useful	Very useful	Moderately useful	Somewhat useful
Developer5	Very useful	Very useful	Very useful	Very useful	Somewhat useful	Moderately useful
Tester1	Moderately useful	Very useful	Very useful	Moderately useful	Very useful	Somewhat useful

- **Preferred ATD views.** During the interviews, we asked the architects about which ATD views they are willing to use to document ATD in their future projects (produce ATD views). As shown in the "Willing to use" columns of Table 21, both architects are willing to use the ATD Detail view, ATD Decision view, ATD-related Component view, and ATD Distribution view to document ATD. We asked the other seven participants (i.e., the manager, developers, and tester) about which ATD views they are willing to get informed regarding the ATD in their projects (consume ATD views). As shown in the "Willing to get informed by" columns of Table 21, most of the seven participants preferred the ATD Detail view, ATD-related Component view, and ATD Distribution view to keep up to date with ATD in the system and further manage it. In addition, three out of the seven participants preferred the ATD Decision view. The ATD Stakeholder Involvement view and ATD Chronological view were considered as the least useful ATD views.

Table 21. ATD views that the participant are willing to use or get informed by

	Willing to use		Willing to get informed by						
ATD viewpoint	**Architect1**	**Architect2**	**Manager1**	**Developer1**	**Developer2**	**Developer3**	**Developer4**	**Developer5**	**Tester1**
ATD Detail view	X	X	X	X	X	X	X	X	X
ATD Decision view	X	X				X		X	X
ATD-related Component view	X	X		X		X	X	X	X
ATD Distribution view	X	X	X		X		X	X	X
ATD Stakeholder Involvement view									
ATD Chronological view								X	

5.4 Interpretation

We discuss our interpretation of the case study results for the research questions as follows.

5.4.1 Interpretation of the results regarding RQ1

RQ1 is about the understandability of the ATD viewpoints. As shown in Table 17, the ATD Detail viewpoint received an average score of 6.8, while each of the other viewpoints received an average score above eight. These scores indicate good understandability of the ATD viewpoints, considering that the case study participants spent only 40 minutes (as described in Table 15) on learning the viewpoints. Among the six ATD viewpoints, the ATD-related Component viewpoint received the highest score, since (1) this viewpoint does not introduce new concepts, and (2) they are more interested in this viewpoint as components are more related to the daily work of most of the case study participants. The ATD Detail viewpoint received the lowest score, because some of the participants suggested that (1) this viewpoint introduces several new concepts, such as principal and interest; and (2) an ATD Detail view contains too much information and it takes time to understand and remember every element of the view (even though participants considered it to be rather comprehensive).

5.4.2 Interpretation of the results regarding RQ2

RQ2 is concerned with the ease of collecting the required information and subsequently creating the ATD views. As shown in Table 18, the case study participants gave relatively low scores to the elements that needed to be estimated, including benefit, principal, potential interest incurred in a change scenario, and probability of the potential interest incurred in a change scenario. When collecting these elements, participants were faced with the difficulties of measuring them for each ATD item. In practice, there lacks an effective approach to measure the elements aforementioned. We need such an ATD measurement approach that is efficient, easy to operate, and with acceptable accuracy. The architecture decisions that incur or repay ATD items also received relatively low scores. The ease of collecting these architecture decisions reflects the ease of ATD identification, which requires significant effort. In addition, collecting architecture decisions that incur and repay ATD items received similar scores (i.e., 6.7 and 7.0, respectively), which indicates that collecting these two types of architecture decisions needs similar amount of effort. This is mostly because one can identify a specific architecture decision that incurs an ATD item only when he or she already comes up with a better solution to repay the ATD item.

Creating the ATD views costs more effort than the chief architect expected. This was for two main reasons. First, creating the ATD Detail view manually is time-consuming since there are many elements that need to be filled in for each ATD item. Second, there was no dedicated supporting tool for generating ATD views during the case study. Instead, we provided Excel templates to help with the ATD views generation. When generating the ATD views, the chief architect needed to read the required information from different Excel files or sheets of the same file, and checked the information in one ATD view with the other ATD views to maintain the consistency between all ATD views.

Considering that the total time spent for ATD documentation in this case study was around one hour, we argue that the cost of ATD documentation was rather minimal. In practice, the effort needed in ATD documentation for a project, largely depends on the number of ATD items to be documented. Furthermore, the effort needed also depends on the number of ATD viewpoints chosen to document ATD. Practitioners do not necessarily have to choose all the viewpoints to document ATD in their projects. Instead, they can choose the ATD viewpoints that are most interesting and useful for their projects. In addition, practitioners may select part of the elements in the ATD Detail view that are most useful for their projects and that are required to create views conforming to other selected ATD viewpoints. In practice, the architect would be mainly responsible for ATD documentation. Developers may also be involved in ATD documentation, since their work may influence the ATD. For instance, when developers have resolved a specific ATD item, they can update the status and history of this ATD item in the ATD Detail view.

5.4.3 Interpretation of the results regarding RQ3

RQ3 focuses on the usefulness in facilitating stakeholders' understanding on the ATD in the selected software project in the case study. As shown in Table 19, all participants considered that the health level of the SA is lower or roughly equal to what they thought before this case study. This indicates that the documented ATD views can help the participants to reach a consensus on the understanding of the architecture's health. Especially, seven out of nine participants (including the two architects and the project manager) considered the architecture to be less healthy than what they expected before the case study. In the interviews with the architects and project manager, they suggested that they had never systematically collected and documented the data on the negative consequences caused by the compromises on the system's maintainability and evolvability.

Although documenting the ATD Detail view is time-consuming, all the participants were willing to use this view in managing ATD in the future. This is mainly because this view contains rich information about ATD items and this information provides the basis to generate other ATD views. The ATD Decision view, ATD-related Component view, and ATD Distribution view were considered more useful than the ATD Stakeholder Involvement view and ATD Chronological view, and most participants were willing to use these three ATD views to manage ATD. This is mostly because these ATD views provide holistic views on all the documented ATD items. Stakeholders can find interesting and valuable information in these views without examining the detailed individual ATD items. The ATD Stakeholder Involvement view was regarded as the lease useful view, since this view is not relevant to the key properties (e.g., cost, benefit, related architecture decisions) of ATD items.

5.5 Implications for research and practice

The results of this case study have implications for both research and practice, as follows:

Implications for research

- Industry welcomes the introduction of the concept of ATD and considers that ATDM is important to keep the long-term health of the architecture. Thus, there is momentum to perform ATD research involving the participation of industry.
- ATD documentation approaches should consider reusing existing artifacts (e.g., documented architecture decisions and change scenarios), so that the effort needed to apply ATD documentation approaches can be reduced. Thus, researchers are encouraged to devise approaches that make as much reuse as possible; this would increase their adoption rate in industry.
- Tool support for ATD documentation approaches is essential for practical use of the approaches in industry. Researchers are encouraged to develop prototype tools that provide such support, and further improve the tools with industrial evaluation.

Implications for practice

- Critical ATD analysis and systematic ATD documentation can help the project team to get an in-depth understanding of the health level of the current architecture.
- Practitioners can choose to document ATD using those ATD viewpoints that are most interesting and useful for their projects and can be afforded in terms of required effort. They do not necessarily have to use all the ATD viewpoint in their projects.

5.6 Threats to validity

We discuss the threats to validity according to different types of validity suggested in the guidelines of reporting case study research [19]. Internal validity is not discussed since we do not investigate causal relationships but only evaluate the ATD viewpoints that we proposed.

Construct validity reflects "to what extent the studied operational measures really represent what the researcher have in mind and what is investigated according to the research questions [19]". A potential threat in case studies is that operational measures are not clearly defined so that the collected data cannot be used to effectively answer research questions. To mitigate this threat, before this case study was performed, we clearly defined the research questions, and the data items that need to be collected for answering each research question. All these data items were collected during the interviews with the participants. Another potential threat is that the participants may have different understandings on the interview questions from the researchers, so that the collected data are not what the researchers expect. In order to alleviate this threat, before the case study, we invited an architect from another company to do a pilot case study. We revised and improved the interview questions according to the feedback from the invited architect. We believe that the threats to construct validity were significantly reduced by the two measures taken above.

External validity is concerned with the generalizability of the case study results [19]. In case studies, there is always a threat to external validity, since only one or several cases are studied, which makes statistical generalization impossible. In [21], Seddon and Scheepers suggest to generalize the results of a single study using *analytic generalization:* "*arguing, based on similarities between relevant attributes of things in a sample and things in other settings, that knowledge claims based on the sample are also likely to hold true in those other settings* [21]." According to the theory of analytic generalization, we believe that the study results are valid for those software projects with similar project and team sizes as well as application domains. In addition, although the case study only took place in a company in China and the cultural context may have played a role in the results, we believe that the study results hold true in similar culture backgrounds. To confirm the aforementioned generalization claims, replication of the study with different project and team sizes in other countries would be desirable.

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers [19]. To make the case study replicable, before we performed the case study, we defined a protocol for this case study in which we clearly defined the research questions, data items to be collected for each research question, interview questions to collect the needed data items, concrete operation steps, and required resources for each step. However, different people may have different understandings on the protocol. To validate the protocol, we invited an architect from another company to carry out a pilot study following the protocol, as already mentioned in 'construct validity'. We revised the protocol according to the feedback received as follows: (1) we improved the Excel templates for producing ATD views; (2) we fine-tuned the timeline of the workshop; (3) we reordered a few interview questions; (4) we provided candidate answers for those interview questions that the participants felt difficult to answer; (5) we reformulated several interview questions that partially overlapped with other questions; and (6) we also reformulated those interview questions containing new concepts that were not introduced in our tutorial. This pilot study effectively improved the data collection procedure and the understandability of the interview questions. Note that we did not include the data collected in the pilot study in the data analysis.

6 Conclusions and future work

ATD has important influence on the long-term health of software architectures, especially on maintainability and evolvability. When left unmanaged, ATD may accumulate significantly, making maintenance and evolution tasks hard to complete. To facilitate ATDM, ATD needs to be recorded in a systematic manner to make it visible to stakeholders and thus facilitate ATD communication and understanding.

To systematically document ATD, in this chapter, we proposed six architecture viewpoints for documenting ATD in software systems. Each ATD viewpoint addresses one or more stakeholders' concerns on ATD, which were collected from literature on TD and derived from ATDM activities. The viewpoints are as follows: (1) The ATD Detail viewpoint is concerned with the detailed information of ATD items in a software system. (2) The ATD Decision viewpoint is concerned with the relationship between architecture decisions and ATD items. (3) The ATD Stakeholder Involvement viewpoint is concerned with the responsibilities of stakeholders in the process of ATDM. (4) The ATD Distribution viewpoint is concerned with the distribution of the amount of the ATD over ATD items and the change of the ATD amount between milestones. (5) The ATD-related Component viewpoint is concerned with the relationship between system components and ATD items. (6) The ATD Chronological viewpoint is concerned with the evolution of ATD items.

To evaluate the effectiveness of the proposed ATD viewpoints in documenting ATD, we conducted a case study in an industrial project in a large telecommunications company. The results of the case study show that: (1) the ATD viewpoints are relatively easy to understand; (2) some of the data (including benefit, principal, interest, and interest probability) that need to be estimated require more effort to collect, compared with other data, such as the compromised QA and affected components; creating an ATD

Detail view also requires relatively more effort while generating the other ATD views are much easier; acceptable effort is needed to generate views using the proposed ATD viewpoints; and (3) the ATD viewpoints are useful in understanding ATD. To summarize, this empirical evaluation shows that the ATD viewpoints can effectively help the documentation of ATD.

The impact of this work is twofold: it contributes (1) to the domain of software architecture with a set of ATD viewpoints for architecture description, and (2) to empirical software engineering and the body of evidence regarding ATD management.

As future work, first, we plan to replicate the case study in more industrial cases with different project and company sizes as well as culture contexts, and continuously revise the ATD viewpoints according to the feedback collected during the case studies. Second, since we received positive feedback from the empirical evaluation on the proposed ATD viewpoints, the next step is to design and develop a dedicated tool to assist with the generation of architecture views conforming to the ATD viewpoints. The tool support can reduce the needed effort by reusing ATD description elements, keep the consistency between ATD views, and improve the traceability between different ATD views.

Acknowledgement

This work is partially supported by AFR-Luxembourg under the contract No. 895528 and the NSFC under the grant No. 61472286. We are thankful to all the participants in the case study and the architect in the pilot study for their help.

Appendix A. ATD concerns

We came up with the concerns on ATD according to two sources: (1) concerns adapted or derived from the concerns on TD in general (TD concerns) collected during our mapping study on TD [1]; (2) concerns derived from ATD management (ATDM) activities in the ATDM process proposed in our previous work (ATDM activities) [7]. From the first source (mapping study), we extracted TD concerns from the primary studies through: (1) the problems addressed by the primary studies; and (2) the problems expected to be solved in future work of the primary studies. We subsequently derived ATD concerns from the identified TD concerns, based on the following criteria: (1) if a TD concern is directly related to the architecture (i.e., not the system details), then the concern is considered as an ATD concern; OR (2) if a TD concern is not about architecture but makes sense to ATD stakeholders, then this concern is regarded as an ATD concern.

From the second source (ATDM activities presented in [7]), we derived ATD concerns based on the concrete tasks performed in each ATDM activity and the intents of the tasks. For instance, in the ATDM activity *ATD measurement*, the involved tasks are to estimate the benefit, interest, and cost of each ATD item, thus, we got the ATD concerns on the quantities of these properties of ATD items. As a result, we derived the ATD concerns C2, C3, C4, and C5. All the resulting ATD concerns and their detailed sources are shown in Table 22.

Table 22. ATD Concerns and their sources

Description of source	**Derived concerns**	**Concern source**
How can I efficiently measure how much debt I already have? [22] How large is my technical debt? [23]	C2	TD concern
How much interest am I paying on the debt? [23]	C5	TD concern
What is the consequence of holding onto a debt for future maintenance? [23]	C19, C20	TD concern
Is the debt growing, and how fast? [23]	C16, C17	TD concern
Technical debt can be considered as a particular type of risk in software maintenance and the problem of managing technical debt boils down to managing risk and making informed decisions on what tasks can be delayed and when they need to be paid back. [14]	C6, C15	TD concern
The analysis and measurement of TD-Principal can guide critical management decisions	C4, C6	TD concern

about how to allocate resources for reducing business risk and IT cost. [24]		
A technical debt "SWAT" team, led by one of the company's most senior architects, tasked with learning how to reduce the technical debt and then rolling that knowledge out to the rest of the development staff, should be established. [25]	C13	TD concern
Which delayed (maintenance) tasks [a type of TD] need to be accomplished, and when. [15]	C6, C15	TD concern
The proposed approach to technical debt management centers around a "technical debt list." The list contains technical debt "items," each of which represents a task that was left undone, but that runs a risk of causing future problems if not completed. [15]	C1	TD concern
Overall, it is important for a project team to understand (1) where TD exists in a system so that it can be tagged for eventual removal, (2) the cost of removing TD (i.e., Principal) and (3) the consequences of not removing TD (i.e., Interest). [26]	C4, C5, C10	TD concern
The person who takes on technical debt is not necessarily the one who has to pay it off. [6]	C8	
Is technical debt increasing or decreasing for a system or for a component? [15]	C18	TD concern
How much debt is "too much" (i.e. high interest) versus manageable (i.e., low interest)? [22]	C21	TD concern
Developers tend to vote for investments into internal quality but managers often tend to question these investments' values and, therefore, tend to decline to approve them. [27]	C7	TD concern
Our questions focus on how technical debt is propagated along those dependencies and how technical debt accumulates at various points in the chain. [28]	C20	TD concern
It enables taking into account not only the sunk cost of development but also the cost yet to be paid to reduce the amount of technical debt. [29]	C4	TD concern
Practices related to identification provide the developer ways to identify Technical Debt in the code whereas classification helps to categorize them in order to understand the reason. [30]	C11	TD concern
After acquiring the source implementation components for technical debt, the *DebtFlag* mechanism completes the projection by propagating technical debt through dependencies while following a possible rule set. [13]	C10, C20	TD concern
ATD identification detects ATD items during or after the architecting process. An ATD item is incurred by an architecture decision; thus, one can investigate an architecture decision and its rationale to identify an ATD item by considering whether the maintainability or evolvability of the software architecture is compromised.[7]	C1, C7, C11, C12	ATDM activity
ATD measurement analyzes the cost and benefit associated with an ATD item and estimates them, including the prediction of change scenarios influencing this ATD item for interest measurement. [7]	C2, C3, C4, C5	ATDM activity
ATD prioritization sorts all the identified ATD items in a software system using a number of criteria. The aim of this activity is to identify which ATD items should be resolved first and which can be resolved later depending on the system's business goals and preferences. [7]	C6	ATDM activity
ATD monitoring watches the changes of the costs and benefits of unresolved ATD items over time.[7]	C9, C14, C16, C17, C18, C19, 21, C21	ATDM activity
ATD repayment concerns making new or changing existing architecture decisions in order to eliminate or mitigate the negative influences of an ATD item.[7]	C13, C15,	ATDM activity

Appendix B. Viewpoint definitions and correspondence rules

In this section, we first propose a shared metamodel of the six ATD viewpoints, then give each ATD viewpoint a detailed definition that can act as guidelines to generate ATD views governed by the ATD viewpoint, and finally define the correspondence rules for the ATD viewpoints.

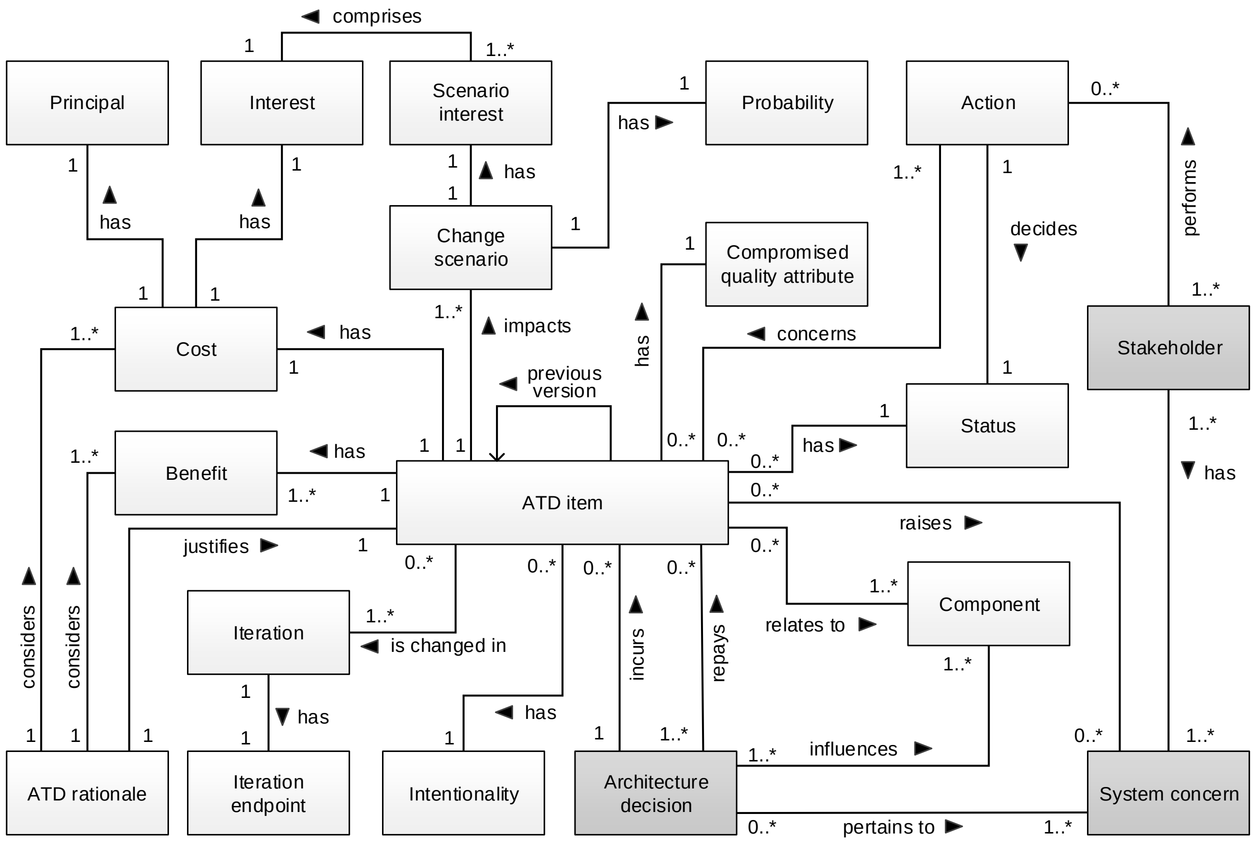


Fig. 6. Metamodel of the ATD viewpoints

B.1 Metamodel of ATD viewpoints

To facilitate the generation of ATD views that are governed by the proposed ATD viewpoints, we constructed a common metamodel that integrates all the elements of the ATD viewpoints. The metamodel also serves to maintain traceability and consistency between different ATD views. Fig. 6 shows the metamodel of the ATD viewpoints. The elements in the dark part of Fig. 6 are concepts adopted from ISO/IEC/IEEE 42010 [9]. An **architecture decision** can incur **ATD item**(s), which is adopted from our previous work [7] and shown in details in Table 4. An **ATD item** relates to one or more **components**, which are influenced by one or more architecture decisions. One or more architecture decisions can be made to repay ATD item(s). An **ATD item** has a specific **status**. An **ATD item** has some **cost** to the future maintenance and evolution of a software system, which is the reason why the ATD item should be managed. The cost of an ATD item has a **principal** and **interest**. The interest of an ATD item is comprised of one or more **scenario interests**, each corresponding to a **change scenario** impacted by the ATD item. A change scenario has an associated **probability**, indicating the possibility that the change scenario will happen. An **ATD item** has some **benefit**(s) which is the reason why the ATD item is incurred. An **ATD item** has a **compromised quality attribute**, i.e., one of the six QAs mentioned in Table 4. An ATD item can raise new **system concern**(s) when the ATD item has significant impact on the system under consideration. For instance, if the ATD item is possible to negatively influence over certain functionality of the system, a new system concern is raised to eliminate or mitigate the negative influence. An **ATD rationale**, which considers the **benefit** and **cost** of the corresponding ATD item, tells why the ATD item is incurred. A **stakeholder** performs an **action** on an **ATD item**, for which the **status** of the ATD item is changed. An ATD item corresponds to an **intentionality**, indicating that it was incurred intentionally or unintentionally. An ATD item may be changed in an **iteration** that has one **iteration endpoint**.

B.2 ATD Decision viewpoint

The ATD Decision viewpoint shows the relationships between ATD items and architecture decisions of a software system. A view conforming to the ATD Decision viewpoint shows all ATD items, which were incurred from the beginning of the ATDM process till the current iteration in a software system, and their relationships with related architecture decisions.

B.2.1 Model kind

The metamodel of the ATD Decision viewpoint is shown in Fig. 7. This metamodel documents the model kind, which describes the conceptual elements for architectural models that conform to the ATD Decision viewpoint. The notation of UML class diagrams is used to describe this metamodel.

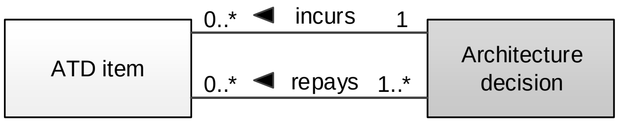


Fig. 7. Metamodel of ATD Decision viewpoint

The constraints listed below apply to the elements within this model kind:

- Every ATD item has a unique ID and name.
- Every architecture decision has a unique ID and name.
- An ATD item is incurred by one architecture decision.
- An ATD item is repaid by one or more architecture decisions.
- An architecture decision can incur or repay zero or more ATD items.

B.3 ATD-related Component viewpoint

The ATD-related Component viewpoint shows the components that are related to ATD items. The number of the related components to a specific ATD item may vary in different versions over time, but, in a view conforming to the ATD-related Component viewpoint, it only shows the ATD items and their related components in the latest versions.

B.3.1 Model kind

The metamodel of the ATD-related Component viewpoint is shown in Fig. 8. This metamodel documents the model kind, which describes the conceptual elements for architectural models that conform to the ATD-related Component viewpoint. The notation of UML class diagrams is used to describe this metamodel.

The constraints listed below apply to the elements within this model kind:

- Every ATD item has a unique ID and name.
- Every component has a unique ID and name.
- An ATD item relates to one or more components.
- A component is related to zero or more ATD items.

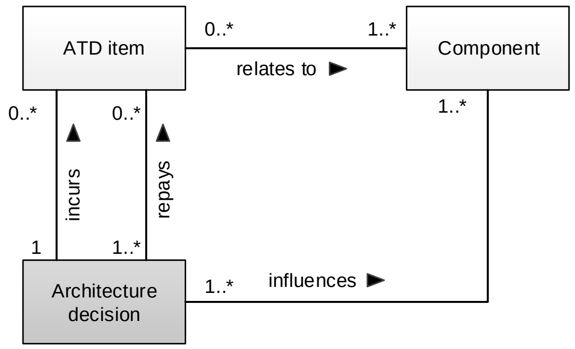


Fig. 8. Metamodel of ATD-related Component viewpoint

B.4 ATD Distribution viewpoint

The ATD Distribution viewpoint shows the costs and benefits of all ATD items in two neighboring iterations.

B.4.1 Model kind

The metamodel of the ATD Distribution viewpoint is shown in Fig. 9. This metamodel documents the model kind, which presents the conceptual elements for architectural models that conform to the ATD Distribution viewpoint. The notation of UML class diagrams is used to describe this metamodel. An iteration endpoint has a date and a type that can be chosen from the following:

- **Milestone**: "A version of the architecture that has reached a stable state (or an intermediate stable state) [17]."
- **Release**: "A version of the architecture that is delivered to a customer of made available to the public for use [17]."

The constraints listed below apply to the elements within this model kind:

- Every ATD item has a unique ID and name.
- Every iteration has exactly one endpoint with a unique name.
- An ATD item has one or more benefits. A benefit can be technical benefit (e.g., QA benefit) or non-technical benefit (e.g., business benefit). Only the measurable benefit is shown in the ATD Distribution viewpoint.
- An ATD item has one cost. The cost is the sum of principal and interest of the ATD item.
- An ATD item (its benefit and cost) can change in one or more iterations.
- In each iteration, zero or more ATD items (their costs and benefits) change.

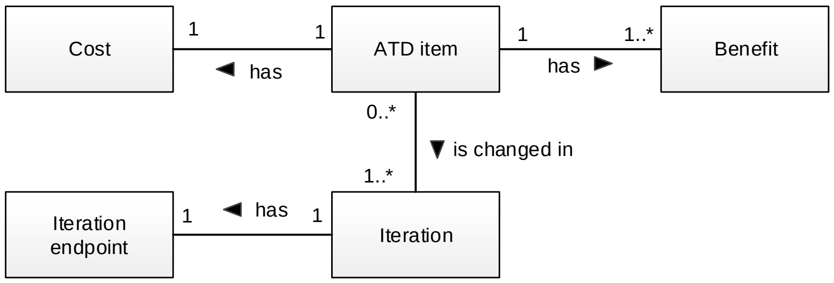


Fig. 9. Metamodel of ATD Distribution viewpoint

B.5 ATD Stakeholder Involvement viewpoint

The ATD Stakeholder Involvement viewpoint shows the responsibilities of relevant stakeholders in the ATDM process. A view conforming to the ATD Stakeholder Involvement viewpoint presents the activities performed by the involved stakeholders on ATD items in the current iteration and their statuses.

B.5.1 Model kind

The metamodel of the ATD Stakeholder Involvement viewpoint is shown in Fig. 10. This metamodel documents the model kind, which describes the conceptual elements for architectural models that conform to the ATD Stakeholder Involvement viewpoint. The notation of UML class diagrams is used to describe this metamodel.

A **Stakeholder** conducts an **Action** on an **ATD item** in a specific development **iteration**, the **Status** of this ATD item changes accordingly. A stakeholder can be any of the defined stakeholders in Section 3.1. We defined the following types of actions in the ATDM process according to the key ATDM activities [7]:

- Identify: stakeholders find out the location of the ATD item.
- Measure: stakeholders estimate the benefit and cost of the ATD item.
- Re-measure: stakeholders estimate the benefit and cost of an ATD item that was measured in previous iterations.

- Prioritize: stakeholders assign a priority to be resolved to the ATD item based on available information related to this ATD item, such as interest.
- Repay: stakeholders resolve the ATD item by making new or modifying existing architecture decisions.

Accordingly, the status of an ATD item can be *Identified*, *Measured*, *Re-measured*, *Prioritized*, and *Resolved*.

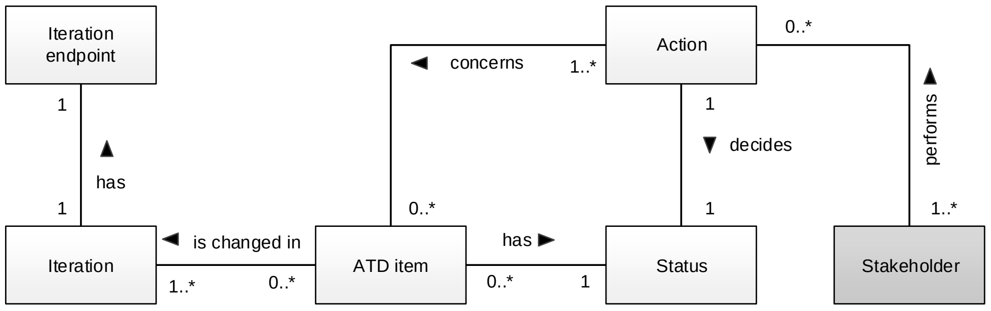


Fig. 10. Metamodel of ATD Stakeholder Involvement viewpoint

The constraints listed below apply to the elements within this model kind:

- Every ATD item has a unique ID and name.
- Every iteration has an iteration endpoint with a unique name.
- All ATD items that changed in one iteration are shown.
- Every stakeholder shown performed at least one action.
- Every stakeholder has a unique name and at least one role.
- Every action points to an ATD item or an iteration endpoint. If the target is an iteration endpoint, the corresponding action is performed for all ATD items changed in that iteration.

B.6 ATD Chronological viewpoint

The ATD Chronological viewpoint shows how the ATD items in a software system evolved over time and how they were managed in the ATDM process.

B.6.1 Model kind

The metamodel of the ATD Chronological viewpoint is shown in Fig. 11. This metamodel documents the model kind, which describes the conceptual elements for architectural models that conform to the ATD Chronological viewpoint. Again, the notation of UML class diagrams is used to describe this metamodel.

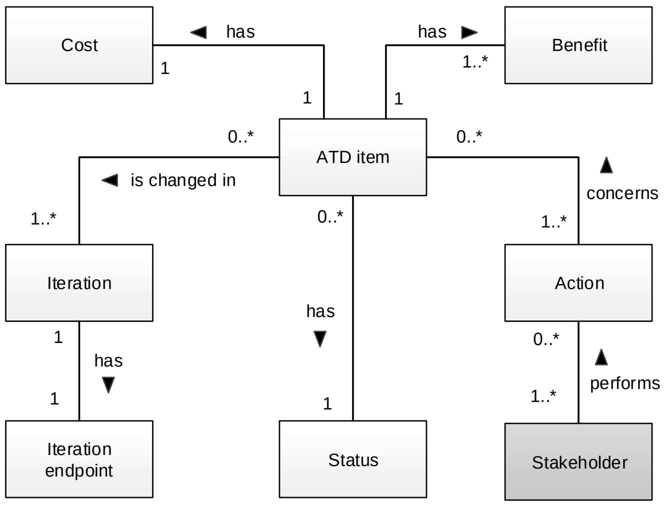


Fig. 11. Metamodel of ATD Chronological viewpoint

The constraints listed below apply to the elements within this mode kind:

- Every ATD item has a unique ID and name.
- Every ATD item has exactly one status at a time.
- Every iteration has exactly one endpoint with a unique name.
- Every ATD item shown is changed in one or more iterations.
- Only an ATD item with the status ‘measured’ shows its benefit and cost.
- Only an ATD item with the status ‘re-measured’ shows its benefit delta and cost delta compared with the previously measured benefit and cost, respectively.

B.7 ATD Detail viewpoint

The ATD Detail viewpoint provides a comprehensive textual description of each ATD item documented in a software project. A view conforming to the ATD Detail viewpoint is comprised of multiple models, each used to describe a single ATD item.

B.7.1 Model kind

The metamodel for the ATD Detail viewpoint is identical to the common metamodel for all the ATD viewpoint as shown in Fig. 6.

B.8 Correspondences between viewpoints

We have proposed six ATD viewpoints to document ATD. We use multiple views governed by these ATD viewpoints to document the ATD of a software system. Each ATD view is comprised of one or more models. Because the same subject is represented in multiple models, there is a risk of inconsistency between different models. Therefore, there is a need to establish rules to express and maintain the consistency of cross-model relationships between ATD description elements. Cross-model relations can be expressed by correspondences, which are introduced in ISO/IEC/IEEE 42010 [9] to express relations between architecture description elements. This international standard further introduces correspondence rules to govern correspondences.

We define a set of correspondence rules in the following to keep the consistency between ATD views:

- An ATD Decision model must contain all ATD items that have ever appeared in the ATD Detail views.
- An ATD-related Component model must exist for every iteration shown in the ATD Chronological model. Every ATD-related Component model contain the ATD items which latest versions are in the status of ‘identified’, ‘measured’, ‘re-measured’, and ‘prioritized’.

- An ATD Distribution model must exist for every iteration shown in the ATD Chronological model. Every ATD Distribution model must contain the existing ATD items that are not in the status of ‘resolved’ in the earlier iteration, and the newly identified ATD items in the later iteration.
- An ATD Stakeholder Involvement model must exist for every iteration shown in the ATD Chronological model. Every stakeholder involvement model must contain the involved stakeholders and their actions in the versions of ATD items belonging to the respective iteration.
- An ATD Chronological model must contains all ATD items that have ever appeared in the ATD Detail views.
- The status of an ATD item in the ATD Detail model must correspond to the status of the latest occurrence of the ATD item in the ATD Chronological model.
- The history of an ATD item represented in the ATD Detail model must contain all actions that are performed by the related stakeholders on that ATD item shown in all ATD Stakeholder Involvement models.

References

[1] Z. Li, P. Avgeriou, P. Liang, A systematic mapping study on technical debt and its management, Journal of Systems and Software, 101 (2015) 193-220.
[2] W. Cunningham, The WyCash portfolio management system, in: Proceedings of the 7th Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'92), ACM, Vancouver, British Columbia, Canada, 1992, pp. 29-30.
[3] F. Buschmann, To Pay or Not to Pay Technical Debt, IEEE Software, 28 (6) (2011) 29-31.
[4] I. Ozkaya, P. Kruchten, R.L. Nord, N. Brown, Managing technical debt in software development: report on the 2nd international workshop on managing technical debt, held at ICSE 2011, SIGSOFT Software Engineering Notes, 36 (5) (2011) 33-35.
[5] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya, R. Sangwan, C. Seaman, K. Sullivan, N. Zazworka, Managing technical debt in software-reliant systems, in: Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER'10), ACM, Santa Fe, New Mexico, USA, 2010, pp. 47-52.
[6] E. Allman, Managing technical debt - Shortcuts that save money and time today can cost you down the road, Communications of the ACM, 55 (5) (2012) 50-55.
[7] Z. Li, P. Liang, P. Avgeriou, Architectural debt management in value-oriented architecting, in: I. Mistrik, R. Bahsoon, R. Kazman, Y. Zhang (Eds.) Economics-Driven Software Architecture, Elsevier, 2014, pp. 183-204.
[8] Z. Li, P. Liang, P. Avgeriou, N. Guelfi, A. Ampatzoglou, An Empirical Investigation of Modularity Metrics for Indicating Architectural Technical Debt, in: Proceedings of the 10th International Conference on the Quality of Software Architectures (QoSA'14), ACM, Marcq-en-Bareul, France, 2014, pp. 119-128.
[9] ISO/IEC/IEEE, Systems and software engineering — Architecture description, ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000), (2011) 1-46.
[10] P. Kruchten, R.L. Nord, I. Ozkaya, Technical Debt: From Metaphor to Theory and Practice, IEEE Software, 29 (6) (2012) 18-21.
[11] ISO/IEC, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, in: ISO/IEC 25010:2011, 2011, pp. 1-34.
[12] S. McConnell, Managing Technical Debt, in, Construx, 2008, pp. 1-14. URL: http://www.construx.com/uploadedFiles/Construx/Construx_Content/Resources/Documents/Managing%20Technical%20Debt.pdf [Accessed on Feb. 1st, 2015]
[13] J. Holvitie, V. Leppänen, DebtFlag: Technical debt management with a development environment integrated tool, in: Proceedings of the 4th International Workshop on Managing Technical Debt (MTD'13), IEEE, San Francisco, CA, USA, 2013, pp. 20-27.
[14] Y. Guo, C. Seaman, A portfolio approach to technical debt management, in: Proceedings of the 2nd International Workshop on Managing Technical Debt (MTD'11), ACM, Waikiki, Honolulu, HI, USA, 2011, pp. 31-34.
[15] C. Seaman, Y. Guo, Measuring and Monitoring Technical Debt, in: M. Zelkowitz (Ed.) Advances in Computers, Elsevier, 2011, pp. 25-45.

[16] N. Zazworka, R.O. Spinola, A. Vetro', F. Shull, C. Seaman, A case study on effectively identifying technical debt, in: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE'13), ACM, Porto de Galinhas, Brazil, 2013, pp. 42-47.
[17] U. van Heesch, P. Avgeriou, R. Hilliard, A documentation framework for architecture decisions, Journal of Systems and Software, 85 (4) (2012) 795-820.
[18] A. Jansen, J. Bosch, Software Architecture as a Set of Architectural Design Decisions, in: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), IEEE, Pittsburgh, Pennsylvania, USA, 2005, pp. 109-120.
[19] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering, 14 (2) (2009) 131-164.
[20] V.R. Basili, Software modeling and measurement: the Goal/Question/Metric paradigm, in, University of Maryland at College Park, 1992, pp. 24.
[21] P.B. Seddon, R. Scheepers, Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples, Eur J Inf Syst, 21 (1) (2012) 6-21.
[22] R.J. Eisenberg, A threshold based approach to technical debt, SIGSOFT Software Engineering Notes, 37 (2) (2012) 1-6.
[23] A. Nugroho, J. Visser, T. Kuipers, An empirical model of technical debt and interest, in: Proceedings of the 2nd International Workshop on Managing Technical Debt (MTD'11), ACM, Waikiki, Honolulu, HI, USA, 2011, pp. 1-8.
[24] B. Curtis, J. Sappidi, A. Szynkarski, Estimating the size, cost, and types of Technical Debt, in: Proceedings of the 3rd International Workshop on Managing Technical Debt (MTD'12), IEEE, Zurich, Switzerland, 2012, pp. 49-53.
[25] I. Gat, J.D. Heintz, From assessment to reduction: how Cutter Consortium helps rein in millions of dollars in technical debt, in: Proceedings of the 2nd International Workshop on Managing Technical Debt (MTD'11), ACM, Waikiki, Honolulu, HI, USA, 2011, pp. 24-26.
[26] D. Falessi, M.A. Shaw, F. Shull, K. Mullen, M.S. Keymind, Practical considerations, challenges, and requirements of tool-support for managing technical debt, in: Proceedings of the 4th International Workshop on Managing Technical Debt (MTD'13), IEEE, San Francisco, CA, USA, 2013, pp. 16-19.
[27] J. Bohnet, J. Döllner, Monitoring code quality and development activity by software maps, in: Proceedings of the 2nd International Workshop on Managing Technical Debt (MTD'11), ACM, Waikiki, Honolulu, HI, USA, 2011, pp. 9-16.
[28] J.D. McGregor, J.Y. Monteith, Z. Jie, Technical debt aggregation in ecosystems, in: Proceedings of the 3rd International Workshop on Managing Technical Debt (MTD'12), IEEE, Zurich, Switzerland, 2012, pp. 27-30.
[29] I. Gat, Technical debt as a meaningful metaphor for code quality, IEEE Software, 29 (6) (2012) 52-54.
[30] V. Krishna, A. Basu, Minimizing Technical Debt: Developer's viewpoint, in: Proceedings of the International Conference on Software Engineering and Mobile Application Modelling and Development (ICSEMA'12), IET, Chennai, India, 2012, pp. 1-5.