Causes of Architecture Changes: An Empirical Study through the Communication in OSS Mailing Lists

Wei Ding[1,4], Peng Liang[1*], Antony Tang[2], Hans van Vliet[3]

[1] State Key Lab of Software Engineering, School of Computer, Wuhan University, China
[2] Faculty of Science, Engineering and Technology, Swinburne University of Technology, Australia
[3] Department of Computer Science, VU University Amsterdam, The Netherlands
[4] Key Laboratory of Earthquake Geodesy, Institute of Seismology, China Earthquake Administration, China
tingwhere@whu.edu.cn, liangp@whu.edu.cn, atang@swin.edu.au, hans@cs.vu.nl

***Abstract*—Understanding the causes of architecture changes allows us to devise means to prevent architecture knowledge vaporization and architecture degeneration. But the causes are not always known, especially in open source software (OSS) development. This makes it very hard to understand the underlying reasons for the architecture changes and design appropriate modifications. Architecture information is communicated in development mailing lists of OSS projects. To explore the possibility of identifying and understanding the causes of architecture changes, we conducted an empirical study to analyze architecture information (i.e., architectural threads) communicated in the development mailing lists of two popular OSS projects: Hibernate and ArgoUML, verified architecture changes with source code, and identified the causes of architecture changes from the communicated architecture information. The main findings of this study are: (1) architecture information communicated in OSS mailing lists does lead to architecture changes in code; (2) the major cause for architecture changes in both Hibernate and ArgoUML is preventative changes. (3) more than 45% of architecture changes in both projects happened before the first stable version was released, which indicates that the architectures of the investigated OSS projects are relatively stable after the first stable release.**

Keywords-architecture change; cause of change; open source software; mailing list; communication

I. INTRODUCTION

Software architecture (SA) represents "*the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution*" [1]. Systems continuously evolve and change to be adapted to new uses, just as buildings change over time [2], which consequently leads to architecture changes. Understanding the causes of architecture changes is important to help practitioners to understand the knowledge of the design decisions that lead to the architecture changes [3], and also allows researchers to devise means to prevent architecture knowledge vaporization and architecture degeneration [4]. The causes of architecture changes are regarded as an essential element of architectural design decision, which is a first-class entity to represent architecture [5], and are used to develop related methods to deal with specific architecture changes, for example, architects analyze due to what cause the property of an architecture is inhibited in order to transform the architecture to satisfy non-functional requirements [6]; architectural styles as analysis tools are used to analyze the causes of architecture changes, and in turn to predict the effect of the architecture changes [7]. Architectural knowledge vaporization (e.g., design decisions and causes of architecture changes) will lead to increased maintenance costs [5]. To prevent this problem, developers (especially architects) need a way to record and communicate the causes of changes in architecture. With an explicit description of architecture as well as their changes [8], software maintainers can better understand the ramification of architecture changes and thereby more accurately analyze the impact and estimate costs of modifications [9]. But the reality is that the rationale of architectural design decisions (e.g., their causes) is often not available in SA documentation [10], especially in OSS development when SA is rarely documented (only 5.4% of 2000 investigated OSS projects have some SA documentation) [11]. We conjecture that causes of architecture changes are communicated between developers through various media, especially in a distributed development context when face-to-face communication is difficult. Mailing list is an important social media for knowledge sharing between knowledge providers and knowledge seekers in OSS projects [12]. Our recent study has shown that communication on architecture does exist in the mailing lists of two popular OSS projects (Hibernate and ArgoUML) [13], and OSS development mailing lists may act as a potential source to extract and identify the cause information of architecture changes in a project.

One of the characteristics of many successful OSS projects is the existence of a SA [14]. Architecture change is also a widespread phenomenon in OSS development, for example, an investigation of the changes in Linux kernel's evolution indicates that most remarkable growth for a "stable" version has been in the addition of new features and support for new architectures rather than fixing defects [15]. To understand the causes of architecture changes [16][17], we conducted an empirical study to extract, identify, and analyze the architecture change information communicated in the OSS mailing lists of two popular OSS projects: Hibernate and ArgoUML based on the data (i.e., architectural threads, which are a set of communication posts on the same topic that contain architecture information in mailing lists) we collected in [13]. The identified architecture changes in

* Corresponding author

This work is sponsored by the NSFC under Grant No. 61170025, 61472286.

(DOI reference number: 10.18293/SEKE2015-193)

mailing lists were further located and verified (confirmed) in the source code of the two projects, and the causes of the architecture changes were classified through the communicated content in architectural threads. The goal of this work is to provide a practical understanding of the causes of architecture changes through communication in OSS mailing lists: Does architecture communication in mailing lists lead to architecture changes in source code? What types of causes that lead to the architecture changes? When do OSS developers communicate the causes of architecture changes?

To answer these questions, we first extracted architecture change information from the architectural threads of OSS mailing lists and further classified the causes of architecture changes with a top-down approach (i.e., using an existing categorization of causes of architecture changes provided in [16]), then checked and verified these changes against source code. We conducted this study based on the architectural threads collected in two popular OSS projects: Hibernate and ArgoUML) [13], in which we identified 131 architectural threads from 20,413 posts in the mailing list of Hibernate from Jan 2002 to Aug 2014; and 200 architectural threads from 26,439 posts in the mailing list of ArgoUML from Jan 2001 to Aug 2014. These architectural threads in Hibernate and ArgoUML are used to extract, identify, and analyze the causes of architecture changes. The results show that the major cause for architecture changes in both Hibernate and ArgoUML is **preventative changes**, which ease future maintenance by restructuring or reengineering the system.

The rest of this paper is organized as follows. A brief review of related work is discussed in Section II. The methodology, including research questions and study process, is described in Section III. The results of this study are presented and discussed in Section IV. Threats to validity are discussed in Section V. We conclude and outline the future directions of this work in Section VI.

II. Related Work

A. *Cause of Architecture Change*

The causes of architecture changes have been explored in software development in various perspectives. The work in [16] uses a systematic literature review to characterize architecture changes from existing literatures. As part of the Software Architecture Change Characterization Scheme (SACCS), a general classification of causes of architecture changes presented in [16] and another work [17] by the same authors can be used as the basic categorization of the causes of architecture changes in the two OSS projects in this study, which is elaborated in Section III. The work in [18] analyzes group interviews in various workshops for different levels of participants, e.g., developers, testers, and architects in five companies. The results validated a taxonomy of the causes for architecture technical debt, a kind of architecture inconsistency, which can be incurred and repaid by architecture changes. The work in [19] uses various versions of an ATM Simulator to observe and analyze what happens when a system evolves and new requirements are added. The results of this work show that changes in requirements may lead to architecture changes and drift, and consequently developers (architects) that do not fully understand the design may take sub-optimal decisions, resulting in design erosion. The authors also identified the causes of design erosion, which can also be the causes for architecture change.

B. *Communication through Mailing Lists in OSS Projects*

Mailing lists in OSS development, as a rich source of communication of development, have been investigated in many studies. The work in [12] discusses the altruistic sharing of knowledge between knowledge providers and knowledge seekers in the developer and user mailing lists of Debian project. The authors developed the Knowledge Sharing Model (KSM) to show how knowledge can be shared (communicated) in OSS mailing lists, and used email exchanges between mailing list participants as quantifiable measures of knowledge sharing activities in OSS development. Some keywords in the subject of posts of mailing lists are used to identify posting and replying posts, e.g., "*Re:*". The study in [20] examined the first posts of newcomers in the mailing lists of four popular OSS projects: MediaWiki, GIMP, PostgreSQL, and Subversion. The authors found that knowledge communication (nearly 80% of newbie posts received replies) was positively correlated with their future participation. Mockus and his colleagues used email archives of source code change history and problem reports to quantify aspects of developer participation, core team size, code ownership, productivity, defect density, and problem resolution intervals, for two large OSS projects, Apache and Mozilla [21]. These works pay attention to all the posts and threads in a mailing list during a certain period, while our study specifically extracts, identifies, and analyzes architecture changes and their causes through the communication in mailing lists.

III. Methodology

To explore the causes of architecture changes through the communication in OSS mailing lists, we select and analyze the mailing lists of two popular OSS projects: Hibernate and ArgoUML, based on the data (i.e., architectural threads) collected in our recent work [13]. In this section, we describe the design of this study with following components: the objective and research questions are presented in Section III.A, the selection criteria of the OSS projects are described in Section III.B, and the study process is elaborated in Section III.C.

A. *Objective and Research Questions*

The goal of this study, formulated using the Goal-Question-Metric (GQM) approach [22] is: *to analyze architecture changes through the communication in mailing lists for the purpose of characterizing the causes of architecture changes from the point of view of OSS developers in the context of OSS development*. We formulate the following research questions (RQs) based on the abovementioned goal.

RQ1: What are the causes that lead to architecture changes in OSS development?

Rationale: Mailing lists have been used as a major vehicle for the communication in OSS development [23]. Architecture information is communicated in the mailing lists of OSS projects [13]. Some of them may discuss specific architecture information, e.g., the causes of architecture changes. With the existing categorization of architecture changes provided in [16], we want to understand in a practical perspective the causes of architecture changes in OSS development through the communicated content extracted from architectural threads. Knowledge and understanding about the causes of architecture changes (evolution) as well as their risks can facilitate the development of strategies to mitigate these risks in software evolution [24].

RQ2: What are the trends of causes that lead to architecture changes in a time perspective?

Rationale: We intend to identify when various types of architecture changes happened and their causes were communicated in a time perspective. The answer of this question would allow us to further identify the best timing for performing treatments to various types of causes of architecture changes, and help practitioners to understand distribution of various causes of the changes in the development lifecycle. To investigate the relationship between causes of architecture changes and time point of releases, the studied period of both OSS projects is divided into two stages according to their first stable releases, i.e., ArgoUML v0.10[1] and Hibernate v1.0 final[2].

B. Selection Criteria of OSS Projects

Three criteria are used in this study to select OSS projects that have mailing lists: (1) The duration of the project is more than 10 years. (2) There are more than 1000 posts in the development mailing list of the project, which provides rich data to mine architecture information. (3) There are more than 50 developers who ever used the mailing list, which is meaningful to analyze the behavior of the developers on communicating architecture information using the mailing list.

Based on these selection criteria, we chose Hibernate and ArgoUML as the OSS projects which mailing lists were analyzed. Hibernate provides an Object/Relational mapping (ORM) framework which implements the Java Persistence API, and is popularly used in Java applications. Hibernate has 20,413 posts in its development mailing list between Jan 2002 and Aug 2014, when the release version was evolved from v0.9.1 to v4.3.6. Note that, the mailing list of Hibernate was migrated from Sourceforge to JBoss in Aug 2006. The mailing list of Hibernate in Sourceforge was initially maintained by a core developer, but he did not continue this administration effort after 11-Aug-2006 [3]. Hence, the investigated time period of the mailing list of Hibernate in this study covers the time period of two mailing lists hosted at Sourceforge and JBoss respectively. ArgoUML is a popular open source UML modeling tool developed in Java. ArgoUML accumulates 26,439 posts in its development mailing list between Jan 2001 and Aug 2014, when the release version was updated from v0.8 to v0.34. The information of the Hibernate and ArgoUML development mailing lists are elaborated in TABLE I.

TABLE I. HIBERNATE AND ARGOUML DEVELOPMENT MAILING LISTS

OSS Project	Mailing list URL	Time period	Num. of Posts
Hibernate	http://sourceforge.net/p/hibernate/mailman/hibernate-devel/	Jan 2002 - July 2006	8,913
	http://lists.jboss.org/pipermail/hibernate-dev/	Aug 2006 - Aug 2014	11,500
ArgoUML	http://argouml.tigris.org/ds/viewForumSummary.do?dsForumId=450	Jan 2001 - Aug 2014	26,439

The IEEE 1471-2000 standard suggests ten main architecture elements for architectural description [25], which were employed as the categorization of architecture elements in our prior work to identify various architecture elements documented in an architecture document [11]. In this study, we also use this categorization to identify various architecture information communicated in mailing lists.

C. Study Process

This study is conducted in three phases. The **first phase** is data collection. We first identify the architectural threads in the mailing lists from the two OSS projects selected in Section III.B. We then extract and classify the causes of architecture changes in the following steps:

Step1: Select 30 architectural threads as the data for a pilot study;

Step2: Identify architectural threads that lead to architecture changes by checking and verifying source code;

Step3: Extract architecture changes and further classify the causes of architecture changes with a top-down approach (i.e., following an existing categorization provided in [16]);

Step4: Review the identified and classified types of causes of architecture changes by two researchers to partially mitigate the threat of personal bias;

Step5: Repeat ***Step1*** to ***Step4*** on the rest architectural threads in the mailing lists of Hibernate and ArgoUML.

The **second phase** is verification of architecture changes in source code. The *Classes*, *Packages*, or architecture design discussed in mailing lists are checked and located in source code. A semi-automatic static code analysis tool Understand[4] is used to identify the changes of source code. Understand can identify the existence of a specific *Class* or *Package* in an Understand project compiled with OSS source code by searching with the name of the *Class* or *Package*. The difference between continuous releases in source code can be used to locate and verify the changes of architecture. For example, if a new *Class* discussed in an architectural thread appears in a certain release, e.g., v1.0, but does not exist in a previous version, e.g., v0.8, we can confirm that the

[1] http://argouml.tigris.org/servlets/NewsItemView?newsItemID=128
[2] http://sourceforge.net/p/hibernate/mailman/message/5497028/
[3] http://sourceforge.net/p/hibernate/mailman/message/13328372/

[4] https://scitools.com/

communication in this architectural thread caused an architecture change (i.e., adding the new *Class*).

The **third phase** is data analysis. Qualitative and quantitative data of architecture changes are extracted from architectural threads, and used to answer the research questions. We manually checked architecture changes discussed in each architectural thread in the mailing lists and recorded the causes of architecture changes identified in the threads in an Excel spreadsheet for further analysis.

IV. RESULTS AND DISCUSSION

A. Cause of Achitecture Change

Using a top-down approach by analyzing the extracted architecture changes, we identified the categories of causes of architecture changes. Architecture changes can be classified in different perspectives. Cause of architecture change is one of them. A recent literature review specifies four categories of causes for architecture changes [16]: (1) **Perfective changes** result from new or changed requirements. These changes improve the system to better meet user needs. (2) **Corrective changes** occur in response to defects in software products. (3) **Adaptive changes** occur when moving to a new environment, platform, or accommodating new standards. (4) **Preventative changes** ease maintenance by restructuring/reengineering the system.

We intend to identify whether architecture communication in mailings lists lead to architecture changes by checking source code in continuous releases. For example, when a new *Class* is suggested by a developer in a mailing list, we will check the name of this *Class* in the following releases and verify whether this *Class* is added or not. If the answer is "Yes", we can further extract the cause of this architecture change from the discussion in the architectural thread of the mailing list. The extracted causes of architecture changes can be directly mapped to the abovementioned four categories of causes for architecture changes with the top-down approach (i.e., following an existing categorization provided in [16]). The percentages of the four types of causes of architecture changes in Hibernate and ArgoUML are showed in TABLE II.

TABLE II. PROPORTIONS OF FOUR TYPES OF CAUSES OF ARCHITECTURE CHANGES IN HIBERNATE AND ARGOUML

OSS Project	Perfective changes	Corrective changes	Adaptive changes	Preventative changes
Hibernate	25.7%	5.7%	20.0%	48.6%
ArgoUML	43.3%	10.8%	2.7%	45.9%

The proportions of the four types of causes of architecture changes before and after the first stable releases of Hibernate and ArgoUML are showed in TABLE III. The abbreviations BFR and AFR represent two stages as described in the rationale of RQ2, i.e., before and after the first stable version was released. Note that, the sum of the percentages of ArgoUML shown in TABLE II and TABLE III exceeds 100%, because one architecture change may be caused by several types of reasons (e.g., adaptive change and perfective change).

TABLE III. PROPORTIONS OF FOUR TYPES OF CAUSES OF ARCHITECTURE CHANGES BEFORE AND AFTER THE FIRST STABLE RELEASE IN HIBERNATE AND ARGOUML

OSS Project	Perfective changes	Corrective changes	Adaptive changes	Preventative changes
BFR Hibernate v1.0 (45.7%)	17.1%	5.7%	0.0%	22.9%
AFR Hibernate v1.0 (54.3%)	8.6%	0.0%	20.0%	25.7%
BFR ArgoUML v0.10 (89.1%)	35.1%	8.1%	2.7%	43.2%
AFR ArgoUML v0.10 (13.5%)	8.1%	2.7%	0.0%	2.7%

Answer to RQ1: There are four types of causes of architecture changes in OSS development: perfective changes, corrective changes, adaptive changes, and preventative changes, which cover all the types of causes of architecture changes in [16]. As shown in TABLE II, the major cause for architecture changes in both Hibernate and ArgoUML is preventative changes.

Answer to RQ2: Perfective changes and preventative changes are the main causes of architecture changes before the first stable releases in both Hibernate and ArgoUML. As shown in TABLE III, after the first stable version was released, the causes of architecture changes of Hibernate are mixed, e.g., adapted to JDK5 (adaptive change) and redesigning Hibernate to be more event-oriented (preventative change); the causes of architecture changes of ArgoUML are mostly perfective changes, e.g., adding a data interface for a new component.

B. Discussion of Study Results

Categorization of causes of architecture changes: All the causes of architecture changes in Hibernate and ArgoUML can be mapped to the four categories of causes of architecture changes provided in [16] and no new category was identified, which empirically validates that this existing categorization does work with OSS projects.

Stable and maintainable architecture: As shown in TABLE III, 45.7% architecture changes in Hibernate and 89.1% architecture changes in ArgoUML happened before the first stable version. It implies that the major part of the architectures was formed and became stable in the initial stage of the two projects. As illustrated in TABLE II, preventative changes are the major cause for architecture changes in both projects. It is not a surprising result. Preventative changes are made to easy future maintenance and evolution. OSS developers tend to make preventative changes (anticipation) in order to achieve a maintainable and evolvable architecture (e.g., refactoring architecture design to be prepared for new or changed requirements). Corrective and adaptive changes are in a small proportion in all architecture changes. The reasons are diverse, potential defects and environmental changes can be prevented and mitigated through preventative changes, or corrective changes are communicated in other sources (e.g., JIRA).

Role of core developers: Core developers refer to those that are actively involved in high levels of communication and knowledge sharing in development [26] (i.e., architecture information communication in this work), e.g., GK[5] in Hibernate and AC in ArgoUML. In this study, we find that 68.5% architecture changes are made by the top two core developers in Hibernate, and 75.7% architecture changes are conducted by the top three core developers in ArgoUML, according to the core developers identified in [13]. These results show that core developers dominate the changes of architecture. These core developers also act as the role of architect in the two OSS projects.

C. Implications for Researchers and Practitioners

For researchers: The results of this study empirically show that architecture communication in OSS mailing lists is correlated with architecture changes in source code. One of the merits of the architecture information communicated in mailing lists is that it contains rich design rationale information about architecture (e.g., cause information about architecture changes), which is particularly useful to enrich architectural design decisions [27] and architecture documentation [28]. Another promising benefit of architecture changes classification in a cause perspective is that it allows researchers to develop a common approach to deal with the changes with similar causes, instead of addressing each change individually (e.g., the purpose of requirements changes classification [29]). To support the approach, a tool for addressing various types of architecture changes can be developed, e.g., certain architecture changes are better addressed by resolving their conflicts with related design decisions [19].

For practitioners: Participants of OSS projects may use the study results to guide them to trace architecture changes from mailing lists. As we have discussed in Section IV.B, architecture changes frequently happened before the first stable version was released. For example, if a new developer of an OSS project wants to get the basic knowledge about the architecture design in order to have a preliminary understanding of the system, s/he can check the architectural threads that suggest, negotiate design, and interpret design implementation through the mailing list during the early stage of development before the first stable release.

V. Threats to Validity

The threats to the validity of this study are discussed according to the guidelines in [30].

Construct validity in this study focuses on whether we extracted architecture information from the mailing lists, identified architecture changes, and interpreted the results of this study correctly. To mitigate the bias on architecture information definition, we chose the architectural description model in IEEE Standard 1471-2000 [25] as a benchmark model to identify the threads that contain architecture information in mailing lists. To identify architecture changes, we compare the changed *Classes*/*Packages* between continuous releases in source code using Understand tool, which mitigates the bias on confirming architecture changes. There is a risk that the results of this study might be affected by the interpretation of the criteria for extracting and identifying architecture information and architecture changes by different researchers. A pilot data extraction was conducted by two researchers to mitigate the bias on understanding and identifying architecture changes. We admit that some causes of architecture changes at the system level are too abstract to be verified in source code by the identification method used in this study. This threat can be mitigated with an understanding of the code structure through the communication with core developers (architects).

[5] Only the abbreviations of the developers' names are provided due to the privacy concern.

Internal validity focuses on the avoidance of confounding factors that may influence the interpretation of the results of a study. There is a risk that the scope of architecture changes might be affected by the identification method used in this work. To mitigate this potential issue, we used the changed *Classes/Packages* to identify the architecture changes in source code by Understand tool. Some architecture changes at the system level discussed in architectural threads were excluded from analysis, because they are too abstract and we could not verify them in source code. We employed a descriptive statistics method to present the results of this study, and the threats to internal validity are minimized. We did not intend to establish any causal relationship between architecture changes and other aspects (e.g., change time) of OSS development in this study.

External validity refers to the degree to which our findings from this study can be generalized. In order to improve the generalizability of the study results and findings, we chose two popular and representative OSS projects that have mailing lists based on the selection criteria in Section III.B. Studying the mailing lists of more OSS projects based on the selection criteria can also alleviate this threat.

Reliability focuses on whether the study yields the same results if other researchers replicate it, which in this work is related to the collection and analysis of architecture changes as well as their causes. By making explicit the process and criteria of data collection and data analysis of this study in Section III, and using Understand (a third-party tool) for verifying architecture changes, this threat is mitigated.

VI. Conclusion and Future Work

In this empirical study, we analyzed the architecture information communicated in architectural threads of the mailing lists of two popular OSS projects: Hibernate and ArgoUML, and located and verified architecture changes by comparing the differences between the source code of continuous OSS releases. Four types of architecture changes in a cause perspective are classified. The main findings of this work are: (1) architectural information communicated in OSS mailing lists does lead to architecture changes in code; (2) the major cause for architecture changes in both Hibernate and ArgoUML is preventative changes. (3) more than 45% of architecture changes in both projects happened before the first stable version was released, which indicates that the architectures of the investigated OSS projects are relatively stable after the first stable release.

The results of this study provide several promising research directions: (1) The results and findings of this work can be further validated through a survey with the core developers of the two OSS projects; (2) As we mentioned in Section IV.C, a tool for a certain type of causes of architecture changes can be developed to deal with similar architecture changes based on the results of this work; (3) Other sources in OSS development, e.g., forums, commit data [31], and blogs [32], may also contain information about architecture changes and their causes. We may explore the possibility to identify architecture changes and their causes from these sources.

REFERENCES

[1] ISO/IEC/IEEE, ISO/IEC/IEEE Std 42010-2011 International Standard, Systems and software engineering - architecture description, 2011.

[2] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture", Software Engineering Notes, ACM, vol. 17, no. 4, pp. 40-52, 1992.

[3] J. Bosch, "Software architecture: The next step", in: Proceedings of the 1st European Workshop of Software Architecture (EWSA), St Andrews, UK, Springer, pp. 194-199, 2004.

[4] A. Jansen, J. van der Ven, P. Avgeriou, and D. K. Hammer, "Tool support for architectural decision", in: Proceedings of the 7th working IEEE/IFIP Conference on Software Architecture (WICSA), Mumbai, India, IEEE, pp. 44-53, 2007.

[5] A. Jansen and J. Bosch, "Software architecture as a set of architectural design decisions", in: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA), Pittsburgh, USA, IEEE, 2005.

[6] J. Bosch and P. Molin, "Software architecture design: evaluation and transformation", in: Proceedings of the 6th IEEE Conference and Workshop on Engineering of Computer-Based Systems (ECBS), Nashville, USA, IEEE, pp. 4-10, 1999.

[7] M. H. Klein, R. Kazman, L. Bass, J. Carriere, M. barbacci, and H. Lipson, "Attribute-based architecture styles", in: Proceedings of the 1st Working IFIP Conference on Software Architecture (WICSA), San Antonio, USA, Springer, pp. 225-243, 1999.

[8] W. Ding, P. Liang, A. Tang, and H. van Vliet, "Knowledge-based approaches in software documentation: A systematic literature review", Information and Software Technology, Elsevier, vol. 56, no. 6, pp. 545-567, 2014.

[9] D. Garlan, "Software architecture: A roadmap", in: Proceedings of the the 22th Conference on Software Engineering, Future of Software Engineering Track (ICSE), Limerick, Ireland, ACM, pp. 91-101, 2000.

[10] R. Weinreich, I. Groher, and C. Miesbauer, "An expert survey on kinds, influence factors and documentation of design decisions in practice", Future Generation Computer Systems, Elsevier, vol. 47, no. 6, pp. 145-160, 2015.

[11] W. Ding, P. Liang, A. Tang, H. van Vliet, and M. Shahin, "How do open source communities document software architecture: An exploratory survey", in: Proceedings of the 19th International Conference on Engineering of Complex Computer Systems (ICECCS), Tianjin, China, IEEE, pp. 136-145, 2014.

[12] S. K. Sowe, I. Stamelos, and L. Angelis, "Understanding knowledge sharing activities in free/open source software projects: An empirical study", Journal of Systems and Software, Elsevier, vol. 81, no. 3, pp. 431-446, 2008.

[13] W. Ding, P. Liang, A. Tang, and H. van Vliet, "Communicating architecture information in open source software development using mailing lists", in: Proceedings of the 9th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Beijing, China, IEEE, 2015. (under review)

[14] A. Brown and G. Wilson, "The Architecture of Open Source Applications", Creative Commons, 2012.

[15] M. W. Godfrey and Q. Tu, "Evolution in open source software: A case study", in: Proceedings of the 16th International Conference on Software Maintenance (ICSM), San Jose, CA, IEEE, pp. 131-142, 2000.

[16] B. J. Williams and J. C. Carver, "Characterizing software architecture changes: A systematic review", Information and Software Technology, Elsevier, vol. 52, no. 1, pp. 31-51, 2010.

[17] B. J. Williams and J. C. Carver, "Examination of the software architecture change characterization scheme using three empirical studies", Empirical Software Engineering, Springer, vol. 19, no. 3, pp. 419-464, 2014.

[18] A. Martini, J. Bosch, and M. Chaudron, "Architecture technical debt: Understanding causes and a qualitative model", in: Proceedings of the 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Verona, Italy, IEEE, pp. 85-92, 2014.

[19] J. van Gurp and J. Bosch, "Design erosion: Problems and causes", Journal of Systems and Software, Elsevier, vol. 61, no. 2, pp. 105-119, 2002.

[20] C. Jensen, S. King, and V. Kuechler, "Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists", in: Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS), Kauai, USA, IEEE, pp. 1-10, 2011.

[21] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla", ACM Transactions on Software Engineering and Methodology, ACM, vol. 11, no. 3, pp. 309-346, 2002.

[22] V. R. Basili, "Software modeling and measurement: The Goal/Question/Metric paradigm", University of Maryland at College Park, 1992.

[23] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen, "Communication in open source software development mailing lists", in: Proceedings of the 10th International Working Conference on Mining Software Repositories (MSR), San Francisco, USA, IEEE, pp. 277-286, 2013.

[24] O. P. N. Slyngstad, J. Y. Li, R. Conradi, and M. Ali Babar, "Identifying and understanding architectural risks in software evolution: An empirical study", in: Proceedings of the 9th International Conference of Product Focused Software Development and Process Improvement (PROFES), Monte Porzio Catone, Italy, Springer, pp. 400-414, 2008.

[25] IEEE, IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software Intensive Systems, 2000.

[26] S. A. Licorish and S. G. MacDonell, "Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study", Information and Software Technology, Elsevier, vol. 56, no. 12, pp. 1578-1596, 2014.

[27] M. Shahin, P. Liang, and M. R. Khayyambashi, "Architectural design decision: Existing models and tools", in: Proceedings of the Joint 8th Working IEEE/IFIP Conference on Software Architecture & 3rd European Conference on Software Architecture (WICSA/ECSA), Cambridge, UK, IEEE, pp. 293-296, 2009.

[28] A. Jansen, P. Avgeriou, and J. S. van der Ven, "Enriching software architecture documentation", Journal of Systems and Software, Elsevier, vol. 82, no. 8, pp. 1232-1248, 2009.

[29] N. Nurmuliani, D. Zowghi, and S. P. Williams, "Using card sorting technique to classify requirements change", in: Proceedings of the 12th IEEE International Requirements Engineering Conference (RE), Kyoto, Japan, IEEE, pp. 240-248, 2004.

[30] M. Höst, P. Runeson, M. C. Ohlsson, B. Regnell, and A. Wesslén, "Experimentation in Software Engineering", Springer, 2012.

[31] J.S. van der Ven and J. Bosch, "Making the right decision: supporting architects with design decision data", in: Proceedings of the 7th European Conference on Software Architecture (ECSA), Montpellier, France, Springer, pp. 176-183, 2013.

[32] D. Pagano and W. Maalej, "How do open source communities blog?", Empirical Software Engineering, Springer, vol. 18, no. 6, pp. 1090-1124, 2013.