Master Thesis Kick-Off in Collaboration with Allianz SE

Min Jeong Yu, 030222

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Outline

Motivation

Research Approach

Current State

Roadmap

Motivation

Why is Architectural Debt (AD) important?

Architectural inadequacy is the most encountered instances of TD.

Holvitie et al., 2014

Architectural aspect of TD has leverage within overall development lifecycle.

Kruchten, 2012

AD concerns the cost of long-term maintenance and evolution of a software system instead of the visible short-term business value.

Kruchten, 2012

Risk when AD makes adding new business value so slow -> widespread refactoring or rebuilding needed

Martini et al., 2014

Lack of quantitative measure / tool to continuously manage AD

Brown et al., 2010

Research Approach

Goal & Research Questions

Title	Designing an Enterprise-wide Governance Framework for Management of Architectural Debt
Goal	Provide guidance on measuring, reducing and governing AD, provide transparency of AD management

Research Questions

- **RQ 1:** What is architectural debt?
- **RQ 2:** How should architectural debt be quantified?
- **RQ 3:** How should business build the governance framework on handling architectural debt?

Research Approach

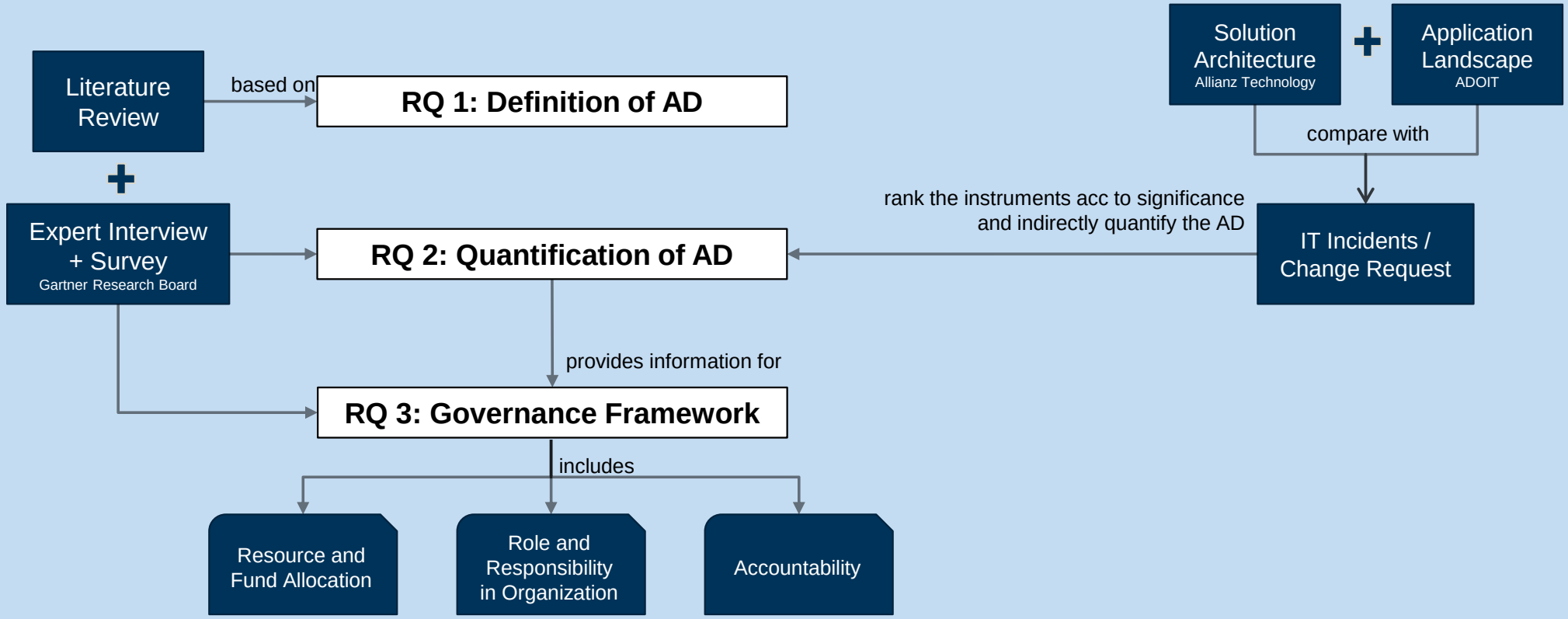
Research Questions

RQ 1: What is architectural debt?

RQ 2: How should architectural debt be quantified?

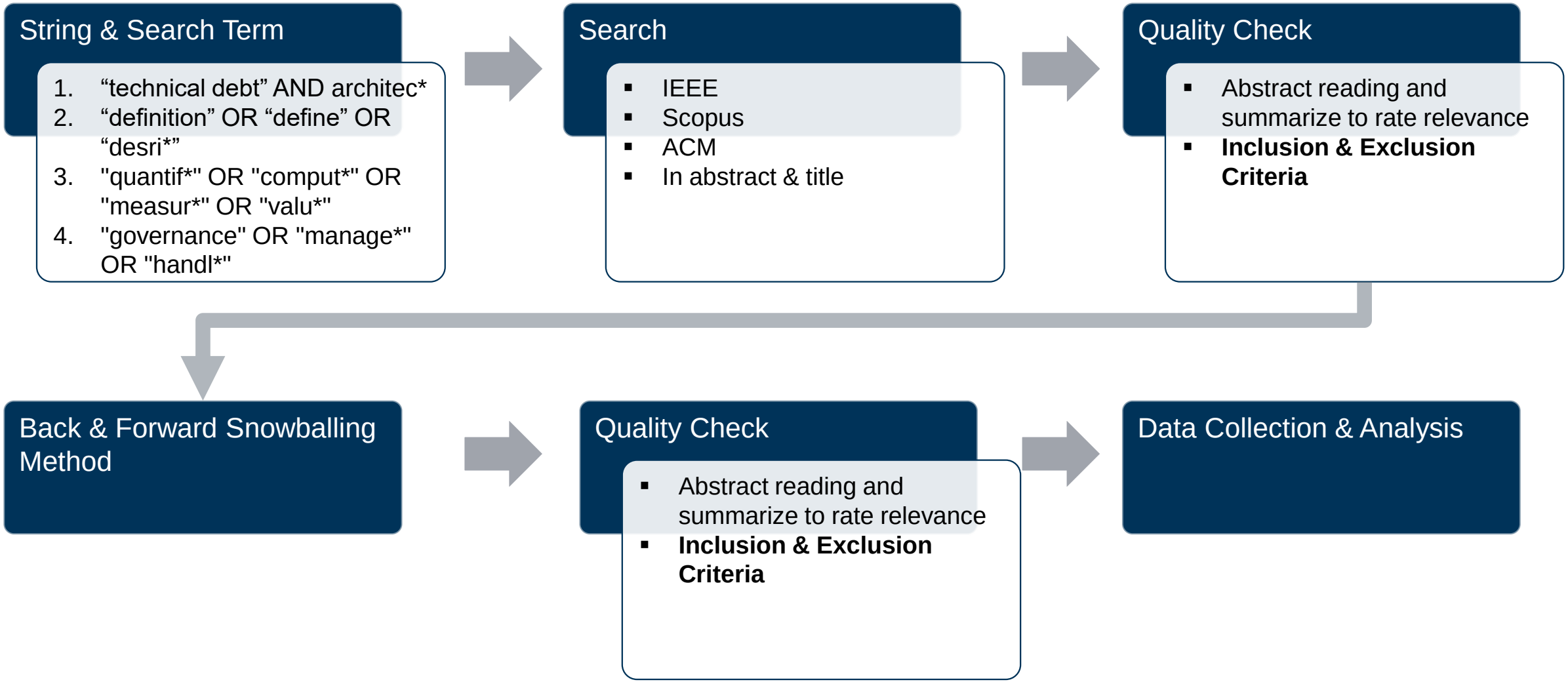
RQ 3: How should business build the governance framework on handling architectural debt?

Methods



Research Approach

Literature Review



Current State

Results of first literature review

RQ 1: Definition of AD

Technical Debt (TD)

The cumulative impact of expedient design and implementation on the continued evolution of a system, which can make future changes more costly or impossible.

Cunningham, 1992

Types of TD
Architectural TD,
Build Debt,
Infrastructure TD,
Requirement TD,
Test Automation TD,
Code TD

Alves et al., 2014

Architectural Debt (AD)

Intentional or unintentional software architectural decisions that differ from best practices or use immature or misapplied software architecture methods.

Besker et al., 2018

The gap between the existing state of a software and some hypothesized ideal state in which the system is optimally successful.

Brown et al., 2010

Enterprise Architecture Debt

“The deviation of the currently present state of an **enterprise** from a hypothetical ideal state.”

Hacks et al., 2019

Interest

“The additional effort needed to spend on maintaining the software, because of its decayed design-time quality”

Ampatzoglou et al., 2015

Principal

“The effort that is required to address the difference between the current and optimal level of design-time quality, in an immature software artifact or the complete software system”

Ampatzoglou et al., 2015

Current State

Results of first literature review

Categories of AD

- Architectural dependency violations
- Inadequacies in the use of patterns or naming conventions
- Code complexity issue
- Integration issue with resources and subsystems
- Lack of mechanism to deal with implementation and test of non-functional requirements

Besker et al., 2018

Cause of AD

- Adding functionality into an overly large module
- Incomplete standards compliance
- Incurred over time as the system is updated and software ages
- Throughout the development cycle, global architecture undermined -> decreased intellectual control and fragmented changes

Besker et al., 2018, Rosser et al., 2021

Effects of AD

Negative Effect

- QA's **maintainability and evolvability**
- **Decay** instances that impact the lifecycle properties like understandability, testability, extensibility, reusability and reliability.

Positive Effect

- Strategic benefits (shorter time to market)

Li et al., 2014, Besker et al., 2018

Current State

Results of first literature review

Management of AD (ADM)

- Identification – Measurement – Prioritization – Repayment – Monitoring
- Main goal of continuous and iterative system monitoring is to capture and track the presence of AD within a system, to provide early warnings to detect costs and risks and to map architectural dependencies or pattern drift to decay.
- Key factor: If & When to refactor architecture.

Li et al. 2014, Besker et al., 2018

Challenges of AD Mangement

- Translating architectural debt into economic consequences and estimating principal cost & interest.
- Inconsistency between different levels of abstraction in the architectural design is difficult to detect, but an important source of AD.
- Through communication across functions & networks, loss of essential information.
- Interest hidden from stakeholders, difficult to decide if refactoring should be done.
- Benefits of refactoring is hard to quantify or justify.

Besker et al., 2018

Current State

Literature Review Sources

Alves, N.S.R., Ribeiro, L.F., Caires, V., Mendes, T.S., Spinola, R.O., 2014. Towards an ontology of terms on technical debt. In: Managing Technical Debt (MTD), 2014 Sixth International Workshop on, pp. 1–7.

Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P., 2015. The financial aspect of managing technical debt: a systematic literature review. Inf. Software Technol. 64, 52.

Besker, T., Martini, A. and Bosch, J., 2018. Managing architectural technical debt: A unified model and systematic literature review. Journal of Systems and Software, 135, pp. 1-16.

Cunningham, W., 1992. The WyCash portfolio management system. In: 7th Inter- national Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA ’92), pp. 29–30.

Hacks, S., Hofert, H., Salentin, J., Yeong, Y.C. and Lichter, H., 2019. Towards the definition of enterprise architecture debts, Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW 2019, pp. 9-16.

Holvitie, J., Leppanen, V., Hyrynsalmi, S., 2014. Technical debt and the effect of agile software development practices on it - An industry practitioner survey, Proceedings - 2014 6th IEEE International Workshop on Managing Technical Debt, MTD 2014 2014, pp. 35-42.

Kruchten, P., 2012. Strategic management of technical debt: tutorial synopsis, Proceedings - International Conference on Quality Software, pp. 282–284.

Li, Z., Liang, P., Avgeriou, P. , 2014a. Chapter 9 - architectural debt management in value-oriented architecting. In: Economics-Driven Software Architecture. Morgan Kaufmann, Boston, pp. 183–204.

Martini, A. , Bosch, J. , Chaudron, M. , 2014. Architecture technical debt: understanding causes and a qualitative model. In: Software Engineering and Advanced Appli- cations (SEAA), 2014 40th EUROMICRO Conference on, pp. 85–92.

Roadmap

Next Steps

- Finish Literature Review
- Expert Interviews
- Model Draft
- Survey Preparation

Thesis Roadmap