

# Eine Untersuchung der Zusammenarbeit zwischen Requirements Engineer und Software Architekt for Seminar Wintersemester 2016/2017

Franz-Dominik Dahmann  
Master Informatik

Hochschule Bonn-Rhein-Sieg  
<https://www.h-brs.de/de>

Grantham-Allee 20, 53757 Sankt Augustin  
Email: franz.dahmann@smail.inf.h-brs.de

Jan Eric Müller  
Master Informatik

Hochschule Bonn-Rhein-Sieg  
<https://www.h-brs.de/de>

Grantham-Allee 20, 53757 Sankt Augustin  
Email: jan-eric.mueller@smail.inf.h-brs.de

**Abstract—** Bei der Realisierung eines Softwareentwicklungsprojektes besteht häufig eine Kluft zwischen den Software-Architekten auf der einen Seite und den Requirements Engineers auf der anderen Seite. Software Architekten sind zum Beispiel häufig mit dem Problem konfrontiert, dass Anforderungsdokumente nicht ausreichend sind um weitreichende Architekturentscheidungen in Bezug auf den Softwareentwurf zu treffen. Daher entsteht in diesem Fall für den Software-Architekten häufig der Mehraufwand, dass dieser bei weiteren Interviews ein präziseres Bild von der gewünschten Software Architektur erhält. Dies resultiert häufig in Verzerrungen die sich dann darin widerspiegeln, dass Termine nicht eingehalten werden können. Betreibt der Software Architekt diesen Mehraufwand nicht und trifft eigene Annahmen bezüglich der Architektur der Software kann dies wiederum zu einer geringeren Akzeptanz des Kunden und im schlimmsten Fall zum Auftragsverlust führen. Die Requirements Engineers wissen auf der anderen Seite wiederum nicht, welche Anforderungen konkret wichtig für den Software Architekten sind, da ihnen die entsprechende Fachkenntnis fehlt. Somit können diese nicht zielführend die notwendigen Informationen mit dem Kunden erarbeiten. Ohne diese Informationen ist eine ausreichende Grundlage der Anforderungen für den Architekturentwurf nicht gegeben. Um diese Kluft zu überbrücken ist es notwendig Verfahren zu ermitteln, die es Requirements Engineers ermöglicht die richtigen Informationen einzuholen und die Zusammenarbeit mit den Software Architekten zu optimieren.

Im folgenden werden Verfahren untersucht die das Potenzial haben die Zusammenarbeit zwischen Requirements-Engineer und Software-Architekt zu verbessern. Ferner wird überprüft wo diese Verfahren ihre Stärken und Schwächen offenbaren und wie diese gewinnbringend kombiniert werden oder sich gegenseitig ergänzen können. Das zentrale Problem der Zusammenarbeit scheint eine mangelnde Verständigung zwischen den Requirements Engineer und dem Software Architekten zu sein. Software Architekten sehen sich gezwungen, entweder eine Architektur anzunehmen, oder weiterführende Gespräche mit dem Kunden zu führen. Requirements Engineers sehen sich mit dem Problem konfrontiert, dass sie nicht die Fachkenntnis haben, die architekturrelevanten Informationen mit dem Kunden zu erarbeiten. Um jedoch eine Optimierung zu ermöglichen ist es zunächst notwendig zu identifizieren, warum die Anforderungsdokumente, die der Requirements Engineer generiert, nicht ausreichen um einen vollständigen

Architekturentwurf herzustellen und wie Erkenntnisse des Software-Architekten wieder in die Anforderungs-Spezifikationen einfließen können. Außerdem wird recherchiert ob Zusammenarbeit und Kommunikation durch eine passende Tool-Unterstützung verbessert werden kann.

Um eine Optimierung zu ermöglichen bieten sich verschiedene Vorgehensweisen an. Im folgenden wird eine Kategorisierung von Anforderungen vorgenommen, die relevant für die Architektur einer Software sein können. Diese Kategorisierung bietet Requirements Engineers die Möglichkeit speziellere Fragen zu der gewünschten Architektur der Software zu stellen. Neben einer Kategorisierung wird ferner ein iteratives Vorgehen zum Design der Software Architektur vorgestellt. Mithilfe des Attribute driven Designs (ADD) soll es möglich sein eine feste Struktur zu haben, anhand derer eine korrekte Architektur entworfen werden kann. Desweiteren wird ein Ansatz vorgestellt und bewertet, welcher die Prozesse der Architektur-Entscheidung und des Requirements-Engineering erfahrungsgetrieben in einen gemeinsamen Prozess integriert. Dieser ermöglicht eine engere Kommunikation zwischen Software-Architekt und Requirements Engineer. Anschließend wird dieser Ansatz mit dem Twin Peaks Modell verglichen und beurteilt welches der beiden Verfahren den größeren Mehrwert bringt oder ob diese sinnvoll kombiniert werden können.

Mit der Untersuchung der Verfahren soll erreicht werden, dass sowohl dem Requirements Engineer als auch dem Software Architekten, Potenziale aufgezeigt werden auf deren Basis die Zusammenarbeit verbessert werden kann. Durch die Beschreibung mehrerer Verfahren soll zudem die Möglichkeit gegeben sein, abzuwägen welche in der individuellen Situation am besten geeignet sind.

## I. EINFÜHRUNG

## II. PROBLEMSTELLUNG

## III. UNTERSUCHUNG GEGEBENER METHODEN

A. *ADD 3.0*

B. *Probing*

C. *Twin Peaks*

D. *Goal-Oriented*

## IV. AUSWERTUNG DER METHODEN

## V. FAZIT

## VI. AUSBLICK

## REFERENCES

- [1] H. Kopka and P. W. Daly, *A Guide to L<sup>A</sup>T<sub>E</sub>X*, 3rd ed. Harlow, England: Addison-Wesley, 1999.