Eine Untersuchung der Zusammenarbeit zwischen Requirements Engineer und Software Architekt als Seminararbeit 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 hufig eine Kluft zwischen den Software-Architekten auf der einen Seite und den Requirements Engineers auf der anderen Seite. Software Architekten sind zum Beispiel hufig mit dem Problem konfrontiert, dass Anforderungsdokumente nicht ausreichend sind um weitreichende Architekturentscheidungen in Bezug auf den Softwareentwurf zu treffen. Daher ensteht in diesem Fall fr den Software-Architekten hufig der Mehraufwand, dass dieser ber weitere Interviews ein prziseres Bild von der gewnschten Software Architektur erhlt. Dies resultiert hufig in Verzgerungen die sich dann darin wiederspiegeln, dass Termine nicht eingehalten werden knnen. Betreibt der Software Architekt diesen Mehraufwand nicht und trifft eigene Annahmen bezglich der Architektur der Software kann dies wiederum zu einer geringeren Akzeptanz des Kunden und im schlimmsten Fall zum Auftragsverlust fhren. Die Requirements Engineers wissen auf der anderen Seite wiederum nicht, welche Anforderungen konkret wichtig fr den Software Architekten sind, da ihnen die entsprechende Fachkenntnis fehlt. Somit knnen diese nicht zielfhrend die notwendigen Informationen mit dem Kunden erarbeiten. Ohne diese Informationen ist eine ausreichende Grundlage der Anforderungen fr den Architekturentwurf nicht gegeben. Um diese Kluft zu berbrcken ist es notwendig Verfahren zu ermitteln, die es Requirements Engineers ermglicht 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 berprft wo diese Verfahren ihre Strken und Schwchen offenbaren und wie diese gewinnbringend kombiniert werden oder sich gegenseitig ergnzen knnen. Das zentrale Problem der Zusammenarbeit scheint eine mangelnde Verstndigung zwischen den Requirements Engineer und dem Software Architekten zu sein. Software Architekten sehen sich gezwungen, entweder eine Architektur anzunehmen, oder weiterfhrende Gesprche mit dem Kunden zu fhren. 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 ermglichen ist es zunchst notwendig zu identifizieren, warum die Anfordungsdokumente, die der Requirements Engineer generiert, nicht ausreichen um einen vollstndigen Architekturentwurf herzustellen und wie Erkenntnisse des Software-Architekten wieder in die Anforderungs-Spezifikationen einflieen knnen. Auerdem wird recherchiert ob Zusammenarbeit und Kommunikation durch eine passende Tool-Untersttzung verbessert werden kann.

Um eine Optimierung zu ermglichen bieten sich verschiedene Vorgehensweisen an. Im folgenden wird eine Kategorisierung von Anforderungen vorgenommen, die Relevant fr die Architektur einer Software sein knnen. Diese Kategorisierung bietet Requirements Engineers die Mglichkeit speziellere Fragen zu der gewnschten 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 mglich 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 ermglicht eine engere Kommunikation zwischen Software-Architekt und Requirements Engineer. Anschlieend wird dieser Ansatz mit dem Twin Peaks Modell verglichen und beurteilt welches der beiden Verfahren den greren Mehrwert bringt oder ob diese sinnvoll kombiniert werden knnen.

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 Mglichkeit gegeben sein, abzuwgen welche in der individuellen Situation am besten geeignet sind.

I. Einführung

II. Problemstellung

Whrend der Zusammenarbeit zwischen Requirements Engineer und Software Architekt knnen vielfltige Probleme auftreten. Bei der Untersuchung mglicher Probleme lsst sich eine Kategorisierung dieser vornehmen. So sind einige Probleme bedingt durch das Personal, whrend andere Probleme sich aus der Qualitt der Anforderungen ergeben. Im folgenden werden die kategorisierten Probleme genauer ausgefhrt.

A. Personal

Bei der Betrachtung der direkt durch das Personal bedingten Probleme fallen folgende besonders auf:

- Schlechte Kommunikation
- Konkurrierende Interessen
- Fehlendes Know-How

1) Schlechte Kommunikation: Die Probleme im Bezug auf eine schlechte Kommunikation sind als grundstzliches Problem in der Zusammenarbeit zwischen mehreren Personen zu sehen. In diesem Zusammenhang lassen sie sich in zwei Klassen aufteilten. Der ersten Klasse lassen sich Probleme zuordnen, bei denen die Gre des Kommunikationsflusses zwischen Requirements Engineer und Software Architekt nicht ausreichend ist. Der zweiten Klasse werden die Probleme zugeordnet, die aus einem beidseitigen Monolog entstehen. Unter einem beidseitigen Monolog ist hierbei zu verstehen, dass in der Kommunikation zwischen Requirements Engineer und Software Architekt kein richtiger Dialog stattfindet, sondern lediglich Informationen und Handlungsanweisungen ausgetauscht werden. ¡REWORK¿

Bei nicht ausreichendem Kommunikationsfluss zwischen Requirements Engineer und Software Architekt kann das Problem aufkommen, dass der Requirements Engineer, whrend der Anforderungsgewinnnung, keine Rcksprache mit dem Software Architekten hlt. Ohne ausreichende Rcksprache kann beispielsweise eine Beeintrchtigung der Qualitt der architekturrelevanten Anforderungen auftreten. Dies kann von fehlenden bis fehlerhaft Anforderungen reichen. Der Verursacher dieses Problems ist am ehesten der Requirements Engineer. Ein weiteres mgliches Problem ist, dass der Software Architekt dem Requirements Engineer nicht ausreichend vermittelt, welche Informationen er fr einen gltigen Architekturentwurf bentigt. Auch hier ist eine mgliche Folge die negative Beeinflussung der Qualitt der Architekturanforderungen.

In der zweiten Klasse werden Probleme gruppiert, bei denen die Gesprchspartner keinen zielfhrenden Dialog fhren, d.h. aneinander vorbei reden. Dies kann kann der Fall sein, wenn zwei Gesprchspartner sich nicht gegenseitig zuhren oder aber die besprochenen Inhalte anschlieend nicht bercksichtigen. Wenn ein Software Architekt einem Requirements Engineer zum Beispiel nicht zuhrt, kann es passieren, dass Anweisungen missverstanden werden und die konzipierte Software Architektur nicht den Wnschen des Kunden entspricht. Ein weiteres Problem ist, dass der Software Architekt die Vorgaben des Requirements Engineer ignoriert und die besprochenen Inhalte nicht bercksichtigt.

2) Konkurrierende Interessen: TODO

3) Fehlendes Know-How: TODO

B. Qualität der Anforderungen

Mit der Untersuchung der Probleme, die sich auf die Qualitt der Anforderungen beziehen, fallen folgende auf:

- Zu restriktive Anforderungen
- Fehlende architekturrelevante Anforderungen
- Ungenaue / sich wiedersprechende architekturrelevante Anforderungen
- Nicht klar hervorgehobene architekturrelevante Anforderungen

1) Zu restriktive Anforderungen: TODO

2) Fehlende architekturrelevante Anforderungen: TODO

3) Ungenaue / sich wiedersprechende architekturrelevante Anforderungen: TODO

4) Nicht klar hervorgehobene architekturrelevante Anforderungen: TODO

Neben den aufgefhrten Problemen gibt es weitere, die hier nicht nher behandelt werden. Darunter fiele zum Beispiel ein phasenbezogene Requirements Engineering.

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 LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.