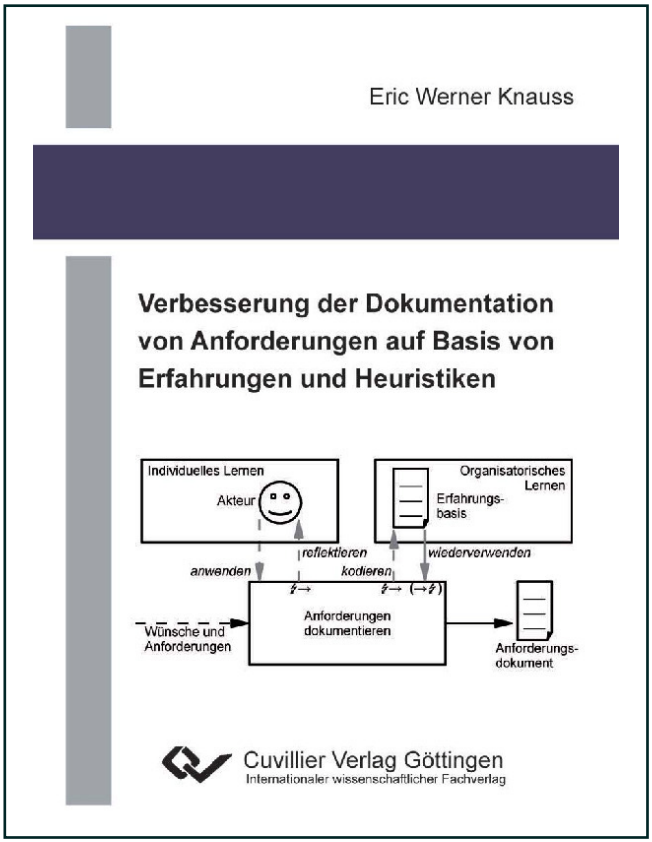
Eric Knauss (Autor)

Verbesserung der Dokumentation von Anforderungen auf Basis von Erfahrungen und Heuristiken



https://cuvillier.de/de/shop/publications/520

1 Einleitung

1.1 Motivation

1.1.1 Problemumfeld

Requirements Engineering umfasst die Analyse und Verwaltung von Anforderungen. Es bildet die Grundlage für planbare Erfolge bei der Erstellung von Software. Entsprechend wird Requirements Engineering seit langer Zeit in der Literatur, auf Konferenzen und in lokalen Arbeitskreisen diskutiert. Trotzdem bleiben in der Praxis viele Fragen offen: Auf welche Aspekte ist, vor allem unter Zeitdruck, besonders zu achten? Wann sind Anforderungen gut genug spezifiziert, um in den Entwurf und die Implementierung überzugehen? Enthält eine Spezifikation noch Missverständnisse?

Die Grundannahme dieser Arbeit ist: Die praktischen Probleme bei der Anwendung von Requirements Engineering liegen nicht darin, dass diese Fragen in der Literatur nicht beantwortet werden. Es liegt auch nicht an schlechter Ausbildung, Faulheit oder Böswilligkeit von Anforderungsingenieuren. Vor dem Hintergrund enger Zeitpläne und hoher Komplexität des zu erstellenden Software-Systems liegen die Ursachen dieser Probleme in der Schwierigkeit der Aufgabe:

- Softwareprojekte werden immer komplexer. Dies führt auch bei der Klärung der Rahmenbedingungen und Anforderungen eines Projekts zu einer kaum überschaubaren Informationsmenge. Die Konsistenz aller anforderungsbezogenen Dokumente und Modelle eines Softwareprojekts ist nur schwer zu wahren.
- Requirements Engineering hängt sehr stark von der persönlichen Erfahrung und den Fähigkeiten einzelner Personen ab. Selbst diesen Experten fällt es aber auf Grund der Komplexität und des Zeitdrucks schwer, ihre Erfahrungen im entscheidenden Augenblick zu aktivieren und anzuwenden.

Um Anforderungsingenieure bei ihrer Aufgabe zu unterstützen, müssen diese zunächst einmal entlastet werden. So bieten sich für die Verarbeitung und Konsistenzprüfung großer Mengen anforderungsbezogener Informationen in unterschiedlichen Dokumenten computerbasierte Lösungen an. Im Rahmen dieser Arbeit wird gezeigt, dass dadurch enorme Zeit (bis zu einem Personenmonat in größeren Projekten) für die formale Konsistenzprüfung eingespart werden kann, die dann für

wichtigere, inhaltliche Aufgaben zur Verfügung steht (siehe Abschnitt 8.3). Zudem wird gezeigt, wie Computer für eine gegebene Situation im Requirements Engineering relevante Erfahrungen ermitteln und dem Nutzer zugänglich machen können. Da dabei hauptsächlich natürlichsprachliche Anforderungsbeschreibungen verarbeitet werden müssen, ist es schwer, Algorithmen zu finden, die eindeutige Ergebnisse liefern. Durch die Mehrdeutigkeit und Kontextabhängigkeit natürlicher Sprache führen die meisten dieser Algorithmen bei ihrer Analyse mit einer gewissen Wahrscheinlichkeit zu Fehlern. Im Rahmen dieser Arbeit werden daher heuristische Regeln verwendet, um Indikationen aufzuspüren, die darauf hinweisen, dass eine Erfahrung anwendbar oder eine Inkonsistenz enthalten ist. Heuristische Regeln liefern häufig das gewünschte Ergebnis, aber nicht immer (vergleiche Abschnitt 5.2), sollen jedoch dennoch nützlich sein. Durch diese im Vergleich zu anderen Algorithmen geringeren Anforderungen lassen sich heuristische Regeln einfacher implementieren und verbessern, in vielen Fällen sogar im Betrieb durch den Nutzer. Abbildung 1.1 zeigt in einem Beispiel, wie heuristische Regeln bei der Erstellung eines Use-Cases eingesetzt werden können. Der Nutzer erhält als computerbasiertes Feedback die Kritik, dass sich passive Formulierungen nicht für Anforderungen eignen.

Abbildung 1.1: Eine heuristische Kritik weist auf ein mögliches Problem hin (hier in der Beispielimplementierung HeRA).

Kritiken auf Basis heuristischer, automatisch prüfbarer Regeln (sogenannte *heuristische Kritiken*) können Inkonsistenzen und Verstösse gegen Best Practices und andere Erfahrungen also direkt aufzeigen (ähnlich der Rechtschreibprüfung bei der Textverarbeitung). Sie können aber auch helfen, Konsistenz zu sichern und Erfahrungen zwischen Personen und Projekten zu transferieren.

1.1.2 Hintergrund

Um die Dokumentation von Anforderungen zu verbessern, sind in Literatur und Praxis verschiedene Ansätze gebräuchlich. Die folgenden beiden Ansätze erfordern zum Teil recht umfangreiche Änderungen im Vorgehen einer Organisation:

- *Prozess:* Durch Verbesserungen am Erstellungsprozess der Dokumente kann eine hohe Steigerung der Qualität von Anforderungsdokumenten erreicht

werden (z.B. [139]). Verbesserungen am Prozess führen dazu, dass Anforderungen verschiedene Bearbeitungs-, Verifikations- und Validierungsschritte durchlaufen müssen, bis sie als ausreichend dokumentiert gelten. Die Einbeziehung unterschiedlicher Rollen erleichtert es, Fehler zu finden und zu beheben.

- *Modelle:* Bessere Modelle (zum Beispiel [175, 82, 21]) führen zu einer formalen Repräsentation von Anforderungen, die mit Hilfe von Model-Checkern oder Simulation überprüft werden kann [22].

Beide Ansätze haben gemeinsam, dass sie die Qualitätskontrolle erleichtern. Sowohl Änderungen am Prozess, als auch Änderungen an der Modellierungsart stellen jedoch große Einschnitte für Software erstellende Organisationen dar. Erfahrungen mit dem alten Prozess oder den bisher verwendeten Modelltypen werden damit größtenteils wertlos.

Etwas weniger einschneidende Verbesserungstypen beziehen sich auf die Verwendung und Verbesserung von Vorlagen (englisch: Templates) oder Qualitätssicherungsmaßnahmen:

- *Vorlagen:* Vorlagen sind ein gutes Mittel, Erfahrungen in Softwareprojekte einzubringen. Sie können als eine Art Checkliste gesehen werden, die verhindert, dass wichtige Aspekte vergessen werden. Zudem geben Vorlagen eine Strukturierung vor, die sich bereits in Projekten bewährt hat. Prominente Vorlagen für die Anforderungsdokumentation sind die Anforderungsschablone von Rupp [142] und die Use-Case-Vorlagen von Cockburn [42].
- *Qualitätssicherungsmaßnahmen:* Für die Qualitätssicherung haben sich verschiedene, überwiegend analytische Verfahren bewährt. Beispiele sind Quality Gates [66] sowie Checklisten und Bewertungskriterien für Anforderungsreviews [48].

Diese beiden Ansätze haben gemeinsam, dass sie eine kontinuierliche Verbesserung erlauben. Nach jeder Anwendung können die Vorlagen und Checklisten angepasst werden, um im gegebenen Kontext einer Organisation noch bessere Resultate zu liefern (siehe [102]).

In der Praxis steht zudem noch die Möglichkeit offen, besonders erfahrene Mitarbeiter mit einzubinden, zum Beispiel als externe Berater. Diese können häufig wertvolles Feedback liefern und helfen, die häufigsten Fallstricke zu vermeiden. Experten sind in der Regel aber knapp und gute Berater teuer. Daher wird diese Option auch dann oft nicht wahrgenommen, wenn der Bedarf an Beratung erkannt wurde.

Der Versuch, fehlende Experten zu substituieren, liegt daher nahe. Die computerbasierte Analyse von Anforderungsdokumentation erlaubt es, mehr oder weniger

komplexe Indikatoren für Fehler zu nutzen [59, 18, 90], um dann geeignete Verbesserungsvoschläge wie durch einen Experten zu unterbreiten. Die Ansätze reichen von der einfachen Suche nach Schlüsselwörtern, über die Analyse von Aspekten der Anforderungsmodelle bis zur Verarbeitung natürlicher Sprache. Je nach Einsatz dieser Ansätze kann so ein Experte zwar nicht vollständig ersetzt werden. Es kann jedoch wertvolles Feedback geben werden, durch das ein vorhandener Experte entlastet wird oder das Fehlen eines Experten abgefedert wird. Die Analyse von Modellen hängt von der spezifischen Repräsentation der Modelle ab und ist damit oft abhängig von bestimmten Modellierungswerkzeugen. Die Analyse natürlicher Sprache gilt als sehr komplex. Dies mögen Gründe dafür sein, dass computerbasierte Ansätze zur Analyse von Anforderungsdokumentation relativ schwach verbreitet sind.

1.2 Zielsetzung und Methodik

Heuristische Kritiken scheinen also eine gute Möglichkeit zu sein, um die Nutzung von Erfahrungen computerbasiert zu unterstützen. Diese Arbeit untersucht, ob sich dieses Konzept eignet, um Erfahrungen im Requirements Engineering zur kontinuierlichen Verbesserung der (natürlichsprachlichen) Dokumentation von Anforderungen zu nutzen. Die Arbeit bedient sich damit bei Ansätzen aus dem Erfahrungsmanagement[1] und wendet sie auf das Requirements Engineering an.

1.2.1 Zielsetzung und Forschungsfragen

Das Ziel der Arbeit ist es zu zeigen, wie Dokumentation von Anforderungen durch erfahrungsbasierte und heuristische Konzepte verbessert werden kann. Aus der Perspektive dieser Arbeit ergeben sich daraus die folgenden Forschungsfragen:

1. Lässt sich durch heuristische Kritiken das systematische Lernen einer Software erstellenden Organisation unterstützen? Wie groß sind mögliche Effekte dieser Unterstützung? Dabei wird a) das Lernen der Organisation und b) das Lernen der Individuen der Organisation unterschieden.
2. Lässt sich durch heuristische Kritiken die Effektivität (d.h. die Qualität) oder die Effizienz (d.h. die Kosten) von Anforderungsdokumenten verbessern? Wie groß sind mögliche Effekte dieser Unterstützung?

Die folgenden Abschnitte zeigen die wissenschaftliche Methodik und das Vorgehen in dieser Arbeit, um dieses Ziel zu erreichen und Antworten auf die Forschungsfragen zu finden.

[1] Erfahrungsmanagement ist ein relativ neuer Begriff und wird in der Regel als Teilgebiet des Wissensmanagement definiert. Vergleiche [23, Seite 14], [151] sowie Def. 10 dieser Arbeit.

Tabelle 1.1: Vergleich des Vorgehens bei konstruktiven und empirischen Arbeiten.

Ansatz	**Konstruktiv**	**Empirisch**
Fokus	Entwickeln und Evaluieren	Hypothesen aufstellen und belegen
Vorgehen	1. Problem der Praxis identifizieren 2. Problem auf Theorie zurückführen 3. Lösung konstruieren 4. Lösung an Problem messen	1. Fragestellung durch anekdotische Beobachtung 2. Hypothesen aufstellen 3. Empirische Untersuchung 4. Hypothesen bestätigen oder verwerfen

1.2.2 Wissenschaftliche Methodik

Wissenschaftliche Arbeiten im Software Engineering lassen sich grob in zwei Kategorien einordnen: *konstruktive Arbeiten*, die neue technische Lösungen für ein Problem vorschlagen, und *empirische Arbeiten*, die existierende Fragestellungen beantworten. Tabelle 1.1 gibt einen Vergleich des typischen Vorgehens in den beiden Kategorien. Diese Arbeit lässt sich nicht eindeutig in eine der beiden Kategorien einordnen. Es werden zwar Forschungsfragen empirisch beantwortet, die Evaluation ist jedoch ohne neue technische Lösungen nicht möglich. Die technischen Lösungen hingegen beruhen auf existierenden Techniken und Konzepten aus den Bereichen Requirements Engineerung und Erfahrungsmanagement. Die Leistung dieser Arbeit ist weniger in den technischen Lösungen zu suchen, sondern in den Konzepten, Requirements Engineering und Erfahrungsmanagement zusammenzubringen.

Das Problem, dass eine neue Idee nicht evaluiert werden kann, ohne den Beobachtungsgegenstand zu verändern, ist für wissenschaftliche Arbeiten nicht neu. So kann der Einfluss des Beobachters auf das Beobachtungsergebnis in naturwissenschaftlichen Experimenten häufig auf das Einbringen von (systematischen) Messungenauigkeiten reduziert werden (zum Beispiel bei der *persönlichen Gleichung* in der Astronomie). Bei sozialen Wissenschaften kommt die soziale Interaktion zwischen Beobachter und Beobachteten noch hinzu. Je nach Kontext stellen sich hier noch zwei weitere Probleme: Durch den Aufbau eines klinischen Experiments, das versucht, alle unerwünschten Einflüsse zu eliminieren, entfernt man sich zu weit von dem zu beobachteten sozialen Gefüge. Zudem ist es teilweise moralisch verwerflich oder auf andere Weise unerwünscht, einer Kontrollgruppe den zu erwartenden Vorteil einer zu evaluierenden Idee vorzuenthalten, wenn dieser dadurch ein Schaden entsteht. Dies hat zur Entwicklung des Action Research geführt [112]. Da diese Schwierigkeiten der sozialen Wissenschaften auch für Un-

tersuchungen von Softwareprojekten gelten, hat das Action Research mittlerweile auch in die Informatik Einzug erhalten [55] und auch diese Arbeit beeinflusst:

Diese Arbeit beruht auf den schwer zu evaluierenden Grundannahmen, dass a) Erfahrung ein entscheidender Erfolgsfaktor für das Requirements Engineering ist und b) Requirements Engineering ein entscheidender Erfolgsfaktor für Softwareprojekte ist. Daraus leitet sich die Hoffnung ab, dass erfahrungsbasierte Konzepte im Requirements Engineering einen wichtigen Beitrag zum Gelingen von Softwareprojekten leisten. Für die Evaluation ist daher ein konstruktiver Anteil zur Erstellung dieser Konzepte unumgänglich. Die dabei erstellten neuen technischen Lösungen sind jedoch exemplarisch gemeint - Ziel der Evaluation ist eigentlich die dahinter stehende Idee. Aus Sicht dieser Arbeit ist eine Idee gut, wenn sie einen Vorteil beim Requirements Engineering einer Organisation bringt.

Da der Nutzen der Ergebnisse im Vordergrund steht, lässt sich der Forschungsansatz nach Easterbrook et al. als Pragmatismus bezeichnen [55, S. 292]. Durch den konstruktiven Anteil besitzt die Arbeit im Großen Anteile des Action Research. Methodisch wird hautpsächlich auf Fallstudien zurückgegriffen, ergänzt um Experimente an zentralen Stellen. Nach Easterbrook et al. ist dies charakteristisch für den Pragmatismus.

1.2.3 Vorgehen

Für diese Arbeit leitet sich nach dem vorherigen Abschnitt und Tabelle 1.1 folgendes Vorgehen ab:

1. *(Empirisch) Fragestellung durch anekdotische Beobachtung:* Läßt sich Anforderungsdokumentation mit Erfahrungen und Heuristiken verbessern? Die Forschungsfragen dieser Arbeit werden im folgendem Abschnitt kurz angerissen, aber auf Basis der Theorie erst in Abschnitt 6.2 detailliert.

2. *(Konstruktiv) Problem auf Theorie zurückführen:* Wie hängen Requirements Engineering, Erfahrungsmanagement und Heuristiken zusammen? Der Rahmen der Arbeit wird gesteckt, indem das Ziel (Verbesserungen im Requirements Engineering, Abschnitt 4.1), das Problemfeld (Anforderungsdokumentation, Abschnitt 4) und der Ansatz (erfahrungsbasiertes Requirements Engineering, Abschnitt 5) theoretisch erarbeitet werden. Die relevanten Aspekte werden definiert und sind Grundlage für den ersten wichtigen Beitrag dieser Arbeit: Der konzeptionellen Definition des Zusammenhangs computerbasierter Analyse von Anforderungsdokumentation und erfahrungsbasierten Ansätzen im Software Engineering.

3. *(Empirisch) Hypothesen aufstellen:* Wie sind die Wirkzusammenhänge von Heuristiken, Dokumentation von Anforderungen und Erfahrungen? Auf wel-

che Faktoren kommt es dabei an? Auf Basis des Konzepts wird die Problemstellung für die Arbeit in Abschnitt 6.2 konkretisiert und mögliche Lösungen charakterisiert. Dies ist ein wichtiger Schritt, denn er leitet über von der theoretischen Beschreibung von Ziel, Problemfeld und Ansatz hin zu konkret mess- und evaluierbaren Lösungen.

4. *(Konstruktiv) Lösung konstruieren:* Mit Hilfe des Konzepts und der Charakterisierung möglicher Lösungen werden systematisch Beispielimplementierungen erarbeitet. Die Auswahl der Beispielimplementierungen erlaubt es, verschiedene Aspekte heuristischer und erfahrungsbasierter Werkzeuge zu evaluieren.

5. *(Empirisch) Hypothesen bestätigen oder verwerfen:* Um tatsächliche von vermeintlichen Wirkzusammenhängen zu trennen, werden die Beispielimplementierungen in unterschiedlichen Evaluationsszenarien angewendet. Die Auswahl der Evaluationsszenarien erlaubt es, empirisch zu untersuchen, welche Arten von Anforderungsdokumentation und welcher Kontext durch erfahrungsbasierte und heuristische Ansätze wirkungsvoll unterstützt werden können.

Abbildung 1.2 zeigt ein Beispiel aus der Evaluation: Die Beispielimplementierung wendet heuristische Kritiken bei der Erstellung von Use-Cases an. Im Evaluationsszenario *studentisches Softwareprojekt* zeigt sich, dass die beiden Gruppen, die auf diese Weise mit Erfahrungen unterstützt werden, bessere Use-Cases schreiben, als die drei Vergleichsgruppen (vergleiche Abschnitt 8.2.3).

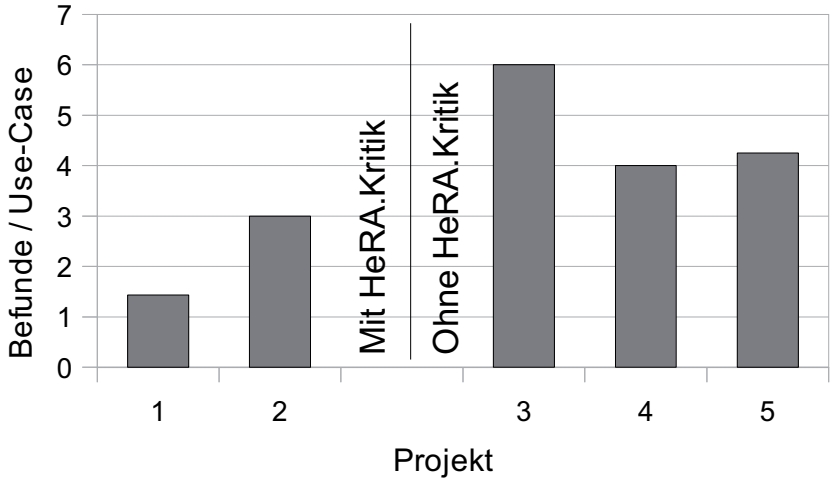


Abbildung 1.2: Evaluatation der Anforderunsqualität: Use-Cases in Projekten mit heuristischer Unterstützung (links) beinhalten weniger Befunde nach Abschluss der Anforderungsanalyse.