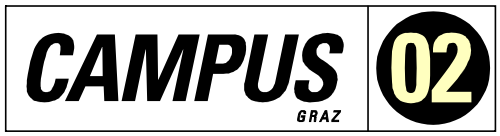
MASTERARBEIT

REQUIREMENTS ENGINEERING IM DOMAIN-DRIVEN DESIGN

ausgeführt an der



am Studiengang

Software Engineering Leadership

Von: Mirka Lehrer

Personenkennzeichen: 1540030009

Stuttgart, am 03.08.2018

Betreuer: Dr. Marcus Winteroll

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

DANKSAGUNG

An erster Stelle möchte ich mich herzlich bei Dr. Marcus Winteroll für die Betreuung meiner Masterarbeit bedanken.

Ein umfassendes Dankeschön geht an alle Teilnehmer der Interviews, die mit ihren Rückmeldungen und Erfahrungen zu dieser Arbeit beigetragen haben.

Ein großer Dank gilt der Pentasys AG, die mir Zeit zur Verfügung gestellt hat, um diese Arbeit zu Ende bringen zu können.

Der größte Dank geht an meine Oma, Dr. Christa Ossmann, weil sie sich größte Mühe gemacht hat, diese Arbeit Korrektur zu lesen.

KURZFASSUNG

Diese theoriegeleitete Fallstudie geht der Frage nach, inwieweit sich das Requirements Engineering (RE) in einem Software-Entwicklungsprozess mit den Konzepten des Domain-Driven Designs (DDD) vereinbaren lässt. Die per Interview und teilnehmender Beobachtung erhobenen Daten wurden gemäß der qualitativen Inhaltsanalyse nach Gläser und Laudel ausgewertet.

Der theoretische Rahmen gibt einen Überblick über die wichtigsten Konzepte des Domain-Driven Designs sowie über das Requirements Engineering, mit Fokus auf der Requirements-Analyse, skizziert Möglichkeiten der Integration des RE in das DDD und stellt Scrum als Vorgehensweise vor.

Die Interviews zeigen, dass dem Projektteam sowohl die Ziele als auch die Produktvision unklar sind und dies zu Verunsicherung und Orientierungslosigkeit im Team führt. Das Requirements Engineering in der Rolle als Mittelsmann zwischen Fachbereich und Entwicklungsteam wird als problematisch eingestuft, da das Domänen-Wissen auf RE-Seite oft nicht ausreicht und der Abstimmungsaufwand durch Dreiecksdiskussionen steigt. Dies erschwert die Formulierung und Nutzung einer Ubiquitous Language und die Modellierung des Domänenmodells. Ist der Domänenexperte gleichzeitig in der Rolle des Product Owners tätig, entsteht zudem ein Rollen- und Interessenkonflikt, der eine DDD-Herangehensweise erschwert.

Ergebnis der Arbeit ist, dass das Requirements Engineering als Vermittler sich nicht mit DDD vereinbaren lässt, aber unterstützende Funktionen einnehmen kann.

Als Handlungsempfehlung kann daraus abgeleitet werden, Situationen zu vermeiden, in denen das Requirements Engineering im Widerspruch zum Domain-Driven Design steht, und Probleme im Entwicklungsprozess mit DDD zu beheben. Methoden und Techniken dafür werden vorgestellt.

Die Arbeit ist für Personen interessant, die Aufgaben des Requirements Engineerings in DDD-basierten Entwicklungsprojekten wahrnehmen, also beispielsweise Projektleiter, Product Owner, Requirements Engineers oder Business Analysten.

GLEICHHEITSGRUNDSATZ

Aus Gründen der Lesbarkeit wurde in dieser Arbeit darauf verzichtet, geschlechtsspezifische Formulierungen zu verwenden. Jedoch möchte ich ausdrücklich festhalten, dass die bei Personen verwendeten maskulinen Formen für beide Geschlechter zu verstehen sind.

INHALTSVERZEICHNIS

1 EINLEITUNG

1.1 Hintergrund

Die Entwicklung von Software ist nicht trivial, denn sie wird von vielen Faktoren beeinflusst. Diese Faktoren können unterschieden werden in solche, die den Entwicklungsprozess beeinflussen, und solche, die sich aus der Aufgabenstellung ergeben, zu deren Lösung die Software entwickelt werden soll. (vgl. Institution of Electrical Engineers, 1980, S. 217).

Zu Ersteren gehören beispielsweise unklare Zielvorstellungen und Zielvorgaben, Sprachbarrieren zwischen Software-Entwicklern und der Auftraggeberseite, die Projektorganisation oder fehlende Ressourcen (vgl. Evans, 2003, S. xvii); sie ergeben die sogenannte extrinsische Komplexität. Letztere beruhen auf den Eigenschaften der zu lösenden Aufgabe innerhalb einer Domäne und bedingen die sogenannte intrinsische oder auch essenzielle Komplexität.

Domain-Driven Design (DDD) ist eine Herangehensweise, welche die essenzielle Komplexität der Gesamtaufgabe handhabbar machen soll, indem die Fachlichkeit der Problemdomäne – sprich: der bestimmte Einsatzbereich der Software – in den Mittelpunkt der Softwareentwicklung gestellt wird.

DDD geht dabei über „harte“ technologische Aspekte hinaus (vgl. Evans, 2003, S. xxii) und bezieht „weiche“ menschliche Faktoren mit ein: Die fundamentalen Prinzipien von DDD sind Diskussion, Zuhören, Verstehen und Entdecken – immer mit dem Ziel, neues Wissen zu erlangen (vgl. Vernon, 2015, S. 2). Um dies zu erreichen, arbeiten im DDD Entwickler mit Experten der entsprechenden Problemdomäne in einem Team. DDD zielt darauf ab, im Verlauf des Entwicklungsprozesses Domänenexperten und Entwickler eine gemeinsame Sprache finden zu lassen (vgl. Evans, 2003, S. 33). Diese *Ubiquitous Language* ermöglicht ein gegenseitiges Verständnis und ist somit eine der Voraussetzungen für eine reibungslose Kommunikation zwischen den Teammitgliedern (vgl. Millett & Tune, 2015, S. 136), da sie die Kluft zwischen beiden Parteien überbrückt (vgl. Vernon, 2015, S. 8).

Als Grundlage für eine fachliche Diskussion dient das Domänen-Modell, das von Entwicklern und Domänenexperten gemeinsam modelliert wird, denn in DDD ist die direkte Zusammenarbeit zwischen Fachexperten und Entwicklern fester Bestandteil der Herangehensweise (vgl. Millett & Tune, 2015, S. 18).

Das Requirements Engineering, bestehend aus Anforderungsanalyse und Requirements Management mit ingenieurmäßigen Vorgehen, wird dagegen auch als „Mittler zwischen den Welten“ beschrieben, weil es zwischen Fachexperten und Entwicklern vermittelt und moderiert (vgl. Rupp, 2014S. 11).

Daher scheint das Requirements Engineering nicht mit den Konzepten und Prinzipien des DDD vereinbar (vgl. Millett & Tune, 2015, S. 17). Dennoch wird dem Requirements Engineering ein Mehrwert zugesprochen, da es unterstützend tätig werden kann. So kann ein Requirements Engineer bei Kommunikationsschwierigkeiten zwischen den Beteiligten in der Rolle des Moderators vermitteln oder den Stakeholdern dabei behilflich sein, die Anforderungen zu konkretisieren und Ergebnisse zu dokumentieren (vgl. Millett & Tune, 2015, S. 17).

Die aufgrund eines möglichen Mehrwerts interessante Kombination beider Ansätze – DDD und Requirements Engineering – wurde bislang nicht ausreichend beschrieben. An diesem Punkt setzt die vorliegende Arbeit an. Untersucht wird, wie ein Requirements Engineering ein Vorgehen mit DDD unterstützen kann. Anhand eines Entwicklungsprojekts werden konkrete Handlungsempfehlungen und Hilfestellungen dazu erarbeitet.

1.2 Problemstellung

Die Aufgaben von Software werden immer vielfältiger und Software wird immer vielschichtiger und umfangreicher. In immer stärkerem Maß sind komplexe und verteilte Organisationen an ihrer Entwicklung beteiligt. Das Requirements Engineering als Schnittstelle zwischen Softwareentwicklung und Anwendungsbereich wird damit immer wichtiger (vgl. Broy, Geisberger & Kazmeier, 2007, S. 127; Weilkiens, 2014, S. 2). In komplexen Entwicklungsprojekten ist es ein Schlüsselfaktor für den Projekterfolg (vgl. Hofmann & Lehner, 2001, S. 102), da es unter anderem die Beachtung des Anforderungskatalogs gewährleistet – ein Scheitern von Projekten lässt sich oft auf mangelhafte Berücksichtigung der Anforderungen zurückführen (vgl. The Standish Group, 2016).

Die Daimler AG hat sich bei der Neuentwicklung eines globalen Vertriebs- und Servicenetzsystems daher dazu entschieden, ein Requirements-Engineering-Team in das bestehende DDD-Team aus Domänenexperten und Entwicklern zu integrieren. Die Pentasys AG wurde von der Daimler AG dazu beauftragt, mit drei Mitarbeitern als Requirements Engineers die Systementwicklung zu unterstützen.

Für die Neuentwicklung wurde eine Microservice-Architektur ausgewählt. Da im Kontext von Microservices DDD in verschiedenen Bereichen hilfreich ist, hat sich das Entwicklungsteam für diese Herangehensweise entschieden. So unterstützt z. B. das Konzept des strategischen Designs bei der Strukturierung von komplexen Microservice-Landschaften. Das Vorgehensmodell im Entwicklungsprozess ist Scrum.

Das mit der Neuentwicklung beauftragte Team besteht aus Entwicklern, den Requirements Engineers und dem Fachbereich. Es hat keine praktische Erfahrung mit DDD und sieht sich sechs Monate nach Projektbeginn mit zwei wesentlichen Problemen konfrontiert:

1. **Anwendung von DDD:**
   Es bestehen Unsicherheiten und Probleme in der Anwendung der Prinzipien und Entwurfsmuster des Domain-Driven Designs. Dadurch werden die positiven Effekte, die durch die Herangehensweise erreicht werden sollten, nicht vollumfänglich erzielt.
2. **Rolle des Requirements Engineerings:**
   Das Requirements-Engineering-Team hat sich in den Entwicklungsprozess integriert und Aufgaben und Verantwortlichkeiten übernommen. Dabei widerspricht das Vorgehen in einigen Funktionen den Prinzipien des DDD. So agiert das Requirements-Engineering-Team als Schnittstellenfunktion zwischen Entwicklern und Domänenexperten. Die positiven Effekte einer direkten Zusammenarbeit kommen deshalb nicht zum Tragen.

1.3 Zentrale Forschungsfrage der Untersuchung

Aus den in Kapitel 1.2 beschriebenen Problemstellungen lässt sich folgende Forschungsfrage ableiten:

Inwieweit lässt sich das Requirements Engineering mit den Konzepten des DDD vereinbaren und wie kann es in einen Softwareentwicklungsprozess, in dem DDD angewendet wird, eingebunden werden, um diesen zu unterstützen?

Zur Beantwortung dieser Forschungsfrage sind die folgenden abgeleiteten Fragen zu klären:

- Welche Aufgaben aus dem Requirements Engineering können in einem Vorgehen mit Domain-Driven Design ausgeführt werden, um dieses zu unterstützen?
- Welche Funktionen aus dem Requirements Engineering widersprechen den Prinzipien des DDD?
- Gibt es im DDD Verantwortlichkeiten, die der Requirements Engineer übernehmen kann, um den Entwicklungsprozess zu unterstützen?
- Welche Verantwortlichkeiten aus dem Requirements Engineering können andere Rollen im Entwicklungsprozess übernehmen, um dieses zu unterstützen?
- Welche Methoden und Techniken können im Requirements Engineering eingesetzt werden, um im Entwicklungsprozess mit DDD zur Wertschöpfung beizutragen?

1.4 Hypothese

Um die aufgeworfene Forschungsfrage beantworten zu können, wird für die Fallstudie eine Hypothese aufgestellt, die aus der Auseinandersetzung mit der Literatur abgeleitet werden kann:

> H1: Das Requirements Engineering kann einen auf Domain-Driven Design basierenden Softwareentwicklungsprozess unterstützen, wenn es nicht klassisch in einer Schnittstellenfunktion zwischen Fachbereich und Entwicklungsteam agiert.

1.5 Zielsetzung

Das Ziel dieser Arbeit ist die kritische Auseinandersetzung mit der Rolle und der Funktion des Requirements Engineerings in der Softwareentwicklung mit DDD.

Mit der Beantwortung der in Kapitel 1.3 aufgeführten Forschungsfragen sollen konkrete Handlungsempfehlungen für das Requirements Engineering zur Lösung der in Kapitel 1.2 beschriebenen Problemstellungen erarbeitet werden.

Diese beziehen sich zum einen auf die Einbettung des Requirements Engineerings in einen Entwicklungsprozess, in dem DDD angewendet wird, zum anderen auf die Hilfestellungen, die das Requirements Engineering zur Behebung der Probleme bei der Anwendung der Prinzipien und Entwurfsmustern des DDD leisten kann.

Grundlage zur Beantwortung der Forschungsfragen ist die Ist-Analyse des aktuellen Modells der Zusammenarbeit zwischen Entwicklern, Domänenexperten und Requirements Engineers im Entwicklungsprozess. Bestehende Probleme werden ermittelt und analysiert.

Es wird zwischen Problemen unterschieden, die durch die Einbettung des Requirements Engineerings in den Entwicklungsprozess selbst verursacht werden, und solchen, die unabhängig vom Requirements Engineering durch eine fehlerhafte Anwendung oder Missachtung der Prinzipien und Entwurfsmuster des DDD entstehen.

Mit der Beantwortung der Forschungsfrage sowie der Entwicklung von konkreten Handlungsempfehlungen für das Requirements Engineering ist die vorliegende Arbeit von praktischer Relevanz für zukünftige Entwicklungsprojekte. Zudem liefert sie einen wissenschaftlichen Beitrag, indem sie die Forschungslücke hinsichtlich der Rolle des Requirements Engineerings im DDD schließt.

1.6 Gliederung der Arbeit

Nachdem in diesem ersten Kapitel der Hintergrund der Arbeit, die Problemstellung samt Forschungsfrage und wegleitender Hypothese sowie die Zielsetzung aufgezeigt werden, folgt im zweiten Kapitel ein Überblick über das Forschungsdesign. Eingegangen wird dort auf die Fallstudie als Forschungsmethode, die Art der Datenerhebung und der Analyse sowie auf die grundlegende Literatur und die Kriterien, welche die Qualität der Forschung bewertbar machen.

Thema des dritten Kapitels sind die theoretischen Grundlagen von DDD, RE und Scrum. Angerissen wird dort auch der Konflikt beim Einsatz von DDD in Verbindung mit RE, da diese beiden Konzepte auf den ersten Blick unvereinbar scheinen.

Das vierte Kapitel beinhaltet die Fallstudie. Beleuchtet werden die Hintergründe, die zur Forschungsfrage geführt haben, sowie die aktuelle Situation im Unternehmen, welches das Softwareentwicklungsprojekt aufgegleist hat.

Im fünften Kapitel werden die Ergebnisse der Fallstudie vorgestellt. In ihm werden die Resultate der Akteursbefragungen und Datensammlung beschrieben; die Forschungsfrage wird beantwortet. Ein Fazit aus der Auswertung der empirischen Untersuchung erfolgt im sechsten Kapitel. Das siebte Kapitel beinhaltet die Handlungsempfehlungen für das Requirements Engineering in einem Entwicklungsvorgehen mit DDD, welche auf den Ergebnissen der Fallstudie basieren.

2 FORSCHUNGSDESIGN

2.1 Übersicht

In diesem Kapitel werden die Methoden, die in der vorliegenden Arbeit zur Untersuchung der Forschungsfragen angewendet wurden, beschrieben und deren Einsatz begründet. Damit soll der Forschungsweg verständlich und nachvollziehbar dargelegt werden.

2.2 Die Fallstudie als Forschungsmethode

Die Forschungsmethode, die dieser Arbeit zugrunde liegt, ist die Fallstudie, englisch Case Study. Sie wird nach Yin (2003) wie folgt definiert (S. 13 f.):

> „Eine Fallstudie ist eine empirische Untersuchung, die
>
> - ein zeitgenössisches Phänomen in einem natürlichen Kontext untersucht, insbesondere wenn
> - die Grenze zwischen beobachtetem Phänomen und Kontext nicht klar erkennbar ist.
>
> Fallstudien werden in der Forschung genutzt, um anhand eines Beispiels einen komplexen Zusammenhang theoretisch durchdringen zu können:
>
> - Gegenstand der Fallstudien sind unterschiedlichen Situationen, in denen weit mehr Variablen von Interesse sind als offensichtliche Datenpunkte.
> - Fallstudien beziehen mehrere unterschiedliche Datenquellen ein, wobei die gewonnenen Daten durch Triangulation auf einzelne Punkte zusammengeführt werden.
> - Im Vorfeld definierte theoretische Annahmen leiten dabei den Prozess der Datenerhebung und -analyse“.

Allerdings ist diese Definition der Fallstudie nicht unumstritten. So wird nach Stake (1985) bei der Fallstudie theorielos vorgegangen. Auch der Einbezug von quantitativen Methoden wird dabei nicht explizit in Erwägung gezogen. Nach Yin (1985) wird die Erhebung von quantitativen Daten nicht ausgeschlossen (S. 17). In dieser Arbeit wird der Ansatz nach Ying verfolgt, weil er hinsichtlich der Wahl der Datenerhebungsmethoden offener und flexibler ist.

Fallstudien berufen sich, anders als rein quantitative bzw. statistische Methoden, nicht auf eine große Anzahl von Stichproben Fallstudien bzw. Fällen, sondern streben eine Interpretation eines Phänomens in seinem Kontext an. Ganz bewusst werden die Rahmenbedingungen des beobachteten Sachverhalts in die Untersuchung einbezogen, da gerade in den gegenseitigen Abhängigkeiten gehaltvolle Informationen vermutet werden. Erkenntnisse ergeben sich durch eine Analyse der komplexen Zusammenhänge (vgl. Göthlich, 2003, S. 7); im Gegensatz zu kontrollierten Experimenten wird die Fallstudie daher nicht in künstlich geschaffenen Umgebungen durchgeführt. Insbesondere, wenn die Grenze

zwischen Phänomen und Kontext unklar ist oder es sich um ein noch nicht erforschtes oder seltenes Phänomen handelt, ist die Fallstudie als Forschungsmethode geeignet, da sie tiefe Einblicke erlaubt (vgl. Runeson & Höst, 2008, S. 132).

Die Durchführung einer Fallstudie erfolgt in mehreren Schritten. Runeson und Höst (2008) unterscheiden dabei die Phasen Fallstudiendesign, Vorbereitung der Datenerhebung, Datenerhebung, Analyse der gesammelten Daten und Reporting (S. 137 f.). Göthlich (2003) fasst die Phasen *Fallstudiendesign* und *Vorbereitung der Datenerhebung* dagegen in der Phase *Planung und Design* zusammen. Das Vorgehen in dieser Fallstudie orientiert sich an Göthlich (2003), da sich die Vorbereitung der Datenerhebung organisatorisch gut in die Planungsphase eingliedern lässt.

Die Fallstudie wird als geeignetes Instrument zur Untersuchung der gegebenen Forschungsfrage angesehen, denn:

1. Die Funktion des Requirements Engineerings in Domain-Driven Design ist ein zeitgemäßes, aber nicht ausreichend erforschtes Gebiet. Bislang liegen dazu keine Studien vor.
2. Im Rahmen des aktuellen Projekts GSSN+ (Global Standard Source for Network Master Data) ist es möglich, das Phänomen in seinem natürlichen Kontext zu erforschen.
3. Die Grenze zwischen dem Phänomen und seinem Kontext kann nicht klar gezogen werden. Als Beispiel kann hier das RE-Team angeführt werden. In seiner Funktion agiert es als Schnittstelle zwischen Fachbereich und Entwicklungsteam, um unter anderem bestehende Probleme im Entwicklungsprozess zu analysieren und zu beheben. Dadurch beeinflusst es aber selbst das Vorgehen, sodass bestimmte Probleme durch die Bemühungen des RE-Teams erst geschaffen werden. Aus diesem Grund verzichtet das DDD auf ein Bindeglied zwischen Fachbereich und Entwicklungsabteilung.

2.3 Planung und Design

In der Planungsphase wird der Grundstein für eine erfolgreiche Durchführung der Fallstudie gelegt.(vgl. Runeson & Höst, 2008, S. 138). Nach Robson (2002) sollte die Planung mindestens folgende Elemente und Fragen berücksichtigen (S. xxi):

- **Ziele.** Was soll die Studie bezwecken?
- **Fall**. Was soll untersucht werden?
- **Theorie**. Welcher Bezugsrahmen ist gegeben?
- **Forschungsfragen**. Welche Erkenntnisse sollen gewonnen werden?
- **Methoden**. Wie werden die Daten erhoben?
- **Strategieauswahl**. Wo sollen die Daten erhoben werden?

Göthlich empfiehlt, die Planungsinhalte in einem Forschungsprotokoll zu dokumentieren, um sich über die Problemstellung und die Forschungsziele klar zu werden (vgl. Göthlich, 2003, S. 8). Des Weiteren

kann es als Leitfaden in der Durchführungsphase eingesetzt werden, um anstehende Arbeiten zu terminieren (vgl. Runeson & Höst, 2008, S. 141).

Im Rahmen der vorliegenden Arbeit wurde ein Forschungsprotokoll erstellt, das die oben aufgeführten Elemente und Fragen zur Planungs- und Designphase behandelt. Es diente sowohl der Planung als auch der Durchführung der Studie. Darin wurden z. B. neben der Darstellung der Ziele die Termine für Interviews sowie Gedanken und weitere Fragen zur Studie notiert.

2.4 Datenerhebung

2.4.1 Theoretische Betrachtung

Um die Gefahr der Überbewertung einer einzigen Informationsquelle zu minimieren und eine ausgewogene Interpretation der Daten zu erlauben, sollten in einer Fallstudie möglichst verschiedene Quellen Verwendung finden (vgl. Runeson & Höst, 2008, S. 144). Die Aussagen gewinnen an Substanz und Glaubwürdigkeit, wenn sie durch unterschiedliche Quellen gestützt werden. Quellen für solche Daten sind z. B. Dokumente, Archivdaten, Interviews, Beobachtungen und Artefakte (vgl. Göthlich, 2003, S. 8 f.). Genutzt als Methoden der Datenerhebung wurden in dieser Arbeit das Interview und die teilnehmende Beobachtung. Deren Einsatz wird im Folgenden näher erläutert.

2.4.2 Interviews

Offene und teilstandardisierte Befragungen wurden mit den Teammitgliedern des Projekts GSSN+ durchgeführt. „Offen“ bezieht sich auf die Möglichkeit des Befragten, sich im Verlauf des Gesprächs frei zu äußern und somit auf weitere Themen zu sprechen zu kommen, die ihm wichtig erscheinen. Teilstandardisiert bezieht sich auf die Vorgehensweise der Befragung durch den Interviewer: Die Fragen wurden nicht in einer bestimmten Reihenfolge gestellt und auf die jeweiligen Rollen angepasst.

Befragt wurden diese Projektbeteiligten:

- vier Entwickler
- zwei Product Owner
- zwei Requirements Engineers

Im Interview mit den Entwicklern und den Requirements Engineers lag der Fokus der Fragen auf der Zusammenarbeit im Team, dem Entwicklungsvorgehen im Allgemeinen und auf den spezifischen Konzepten des Domain-Driven Designs. Das Interview, das mit beiden Product Ownern gemeinsam durchgeführt wurde, war darauf ausgerichtet, deren Verständnis und Haltung zum Thema Domain-Driven Design abzufragen.

Das Interview als Methode zur Datenerhebung eignete sich in diesem Fall gut, da unterschiedliche Sichtweisen, Meinungen und Stimmungen der Teammitglieder gezielt abgefragt werden konnten. Zusätzlich konnten bestimmte Informationen zu vergangenen Ereignissen erhalten werden, die nicht dokumentiert vorlagen.

2.4.3 Teilnehmende Beobachtung

Als Teil des Requirements-Engineering-Teams war es der Forscherin möglich, Daten während der täglichen Arbeit zu sammeln. Während verschiedener Meetings sowie bei vor- und nachgelagerten Diskussionen konnten das Verhalten der Teammitglieder, die Gespräche sowie die Stimmung im Team beobachtet und wahrgenommen werden. Die Beobachtungen wurden im Forschungsprotokoll festgehalten und bilden zusammen mit den Daten der Interviews die Grundlage für den Analyseteil der Fallstudie.

2.5 Datenauswertung

2.5.1 Theoretische Grundlage und Vorgehen

Die Datenauswertung orientiert sich an der Methode der qualitativen Inhaltsanalyse nach Gläser und Laudel (2010). Die auszuwertenden Texte werden dabei als Materialien behandelt, die Daten beinhalten. Die Schritte der qualitativen Inhaltsanalyse, die in dieser Arbeit umgesetzt wurden, sind die Extraktion (Entnahme von Rohdaten) und die anschließende Auswertung der Daten (vgl. Gläser & Laudel, 2010, S. 230). Die Schritte werden im Folgenden genauer beschrieben.

2.5.2 Extraktion

Zur Extraktion der Informationen aus den durchgeführten Interviews wurde auf Basis der Hypothese H1 (s. Kap. 1.4) ein Suchraster in Form eines Kategoriensystems konstruiert. Dieses konnte während der Extraktion angepasst werden, wenn über die bestehende Systematik hinausgehend für die Beantwortung der Forschungsfrage relevante Informationen im Text entdeckt wurden (vgl. Gläser & Laudel, 2010, S. 201).

Die Extraktion der Informationen aus den acht vorliegenden transkribierten Interviews erfolgte manuell in Tabellenform (s. Anhang- Extraktion). Es wurde ein Kategoriensystem aus Hauptkategorien und Subkategorien erstellt. Tabelle 1 zeigt die Haupt- und Subkategorien, denen in der Auswertung die Textsegmente wurden entsprechenden zugeordnet wurden.

Hauptkategorie	Subkategorie
Ziele	Projektziele; Iterationsziele
Vision	-
DDD	Erfahrung mit DDD; Definition DDD; Vorteile DDD; Probleme mit DDD
Sprache	-
Domänenexperte	-
RE	Aktuelle Rolle des RE; mögliche Rolle des RE
Gemeinsame Sprache	-
Carving	-
Context Map	-
Event Storming	-

Tabelle 1: Kategoriensystem der Datenauswertung

Im nächsten Schritt folgt die Auswertung. Deren Ziel ist die Beantwortung der Forschungsfrage als Ausgangspunkt der Untersuchung (vgl. Gläser & Laudel, 2010, S. 246).

2.5.3 Auswertung

Die Vorgehensweise bei der Auswertung hängt jeweils von der Forschungsfrage und dem vorliegenden empirischen Material ab und soll auf die Untersuchungsstrategie abgestimmt sein, daher kann sie nach Gläser & Laudel (2010) keinen allgemeinen Regeln folgen.

In dieser Arbeit leitet die Hypothese H1 die empirische Forschung an und dient somit der Strukturierung der Auswertung. In den einzelnen sich aus der Hypothese ergebenden Kategorien wurden Sachverhalte und Kausalmechanismen identifiziert.

Dafür wurden die Informationen aus den Textsegmenten den verschiedenen Dimensionen zugeordnet: Ursachendimension, Sachdimension und Wirkungsdimension (s. Anhang- Extraktion). Die Unterteilung diente dem Ziel, Kausalketten und Argumentationsgänge systematisch zu erfassen (vgl. Gläser & Laudel, 2010, S. 190). Sie wurde nur dann vorgenommen, wenn in den geschilderten Sachverhalten Ursachen und Wirkungen erkennbar waren.

Der Schritt der Auswertung ist in Anhang 0 ausführlich dokumentiert und bietet die Grundlage für die Darstellung und Diskussion der Ergebnisse (s. Kap. 5).

2.6 Literatur

Basis der hier vorliegenden (theoriegeleiteten) Fallstudie ist eine Auseinandersetzung mit der relevanten Literatur. Die zentralen Themen der Thesis sind das Requirements Engineering und das Domain-Driven Design.

Die Literaturrecherche zum Thema RE umfasst aktuelle Standardwerke mit hohem Praxisbezug aus dem deutschen sowie dem internationalen Sprachraum. Dazu zählen Rupp (2014), Pohl und Rupp (2015), Hruschka (2014) und Robertson und Robertson (2013). Des Weiteren werden ausgesuchte wissenschaftliche Arbeiten weiterer Autoren zur Ergänzung und Verdeutlichung dargestellter Konzepte

zitiert. Da im Projekt GSSN+ die Anforderungsanalyse den Arbeitsschwerpunkt des RE-Teams darstellt und die Verwaltung von Anforderungen von nebensächlichem Interesse ist, wird der Themenkomplex des Requirements-Managements in der Auseinandersetzung mit der Literatur nicht betrachtet.

Für die Auseinandersetzung mit dem Thema Domain-Driven Design werden hauptsächlich die Standardwerke von Evans (2003), Vernon (2015) und Millet und Tune (2015) zitiert.

Eine vollständige Auseinandersetzung mit der relevanten Literatur erfolgt in Kapitel 3.

2.7 Qualität der Forschung

In der Fallstudienforschung stützt sich der Erkenntnisgewinn nicht auf statistische Signifikanz, sondern auf die Verbindung verschiedener Beweise, Aussagen, Darstellungen und Dokumente, um zu einer aussagekräftigen und relevanten Schlussfolgerung zu kommen (vgl. Runeson & Höst, 2008, S. 137). Um auszuschließen, dass persönliche Eindrücke und Annahmen der durchführenden Person die Ergebnisse beeinträchtigen, sollte die Datenbank aus Artefakten, die ihr zur Beweisführung dienen, vollständig und verständlich sein. Keine Schlussfolgerung darf ohne entsprechenden Beleg erfolgen (vgl. Göthlich, 2003, S. 10 f.). Die Belege dieser Arbeit befinden sich im Anhang (s. Anhnag- Ergebnisse der teilnehmenden Beobachtung) und werden bei Schlussfolgerungen stets zitiert.

Allerdings wurde nicht jedes Gespräch, jede Diskussion oder jedes Ereignis im Rahmen der teilnehmenden Beobachtung festgehalten. Deshalb kann nicht vollständig ausgeschlossen werden, dass eine gewisse Voreingenommenheit auf Durchführungsseite vorhanden ist und somit Interpretationen persönlich gefärbt sein könnten.

In den Interviews wurden die Teammitglieder unter anderem dazu aufgefordert, sich kritisch zur Funktion des RE im aktuellen Entwicklungsprozess zu äußern. Da die Interviewerin zum Zeitpunkt der Befragung aber selbst Teil des RE-Teams war, konnte von einer gewissen Befangenheit aufseiten der interviewten Personen ausgegangen werden. Um eine Scheu vor kritischen Äußerungen zu vermeiden, wurde das Problem in einer dem Interview vorausgehenden E-Mail angesprochen, in der eine kritische Haltung gegenüber dem RE als gewünscht gekennzeichnet wurde (siehe Anhang- E-Mail).

Die Interviews wurden anonym durchgeführt, auf eine Audioaufnahme wurde bewusst verzichtet. Dadurch sollte eine vertrauensvolle Umgebung geschaffen werden, in der sich die zu interviewte Person offen und kritisch äußern konnte.

3 THEORETISCHER RAHMEN

3.1 Bedeutung der theoretischen Einbettung

Zur Einbettung der Fallstudie in einen theoretischen Kontext und zur Generierung von möglichen Hypothesen erfolgt in diesem Kapitel eine ausführliche Auseinandersetzung mit der relevanten Literatur zu den Themen Domain-Driven Design (DDD) und Requirements Engineering (RE). Die wichtigsten Begriffe, Konzepte und Prinzipien werden erklärt und in Bezug auf die Forschungsfrage kritisch gewürdigt.

Anschließend werden die Themen DDD und RE im Zusammenhang betrachtet und analysiert. Da sich das Projektteam in seinem Entwicklungsvorhaben an dem Vorgehensmodell Scrum orientiert, werden die wichtigsten Begriffe dieses agilen Rahmenwerks zur Softwareentwicklung erklärt, um das Verständnis der Projektsituation zu verbessern.

3.2 Domain-Driven Design

3.2.1 Bedeutung des DDD

Domain-Driven Design als Herangehensweise legt den Fokus der Entwicklung von Software auf ein bestehendes, zu lösendes Problem – die Domäne. Diese Problemdomäne wird zum Treiber der Softwareentwicklung und bestimmt deren Design, sprich Auslegung. DDD beruht dabei auf der Annahme, dass nicht nur die technische Natur des Problems zu beachten ist, sondern insbesondere auch die fachlichen und menschlichen Anforderungen der beteiligten Personen zu berücksichtigen sind. Dies ist Ursache der Komplexität vieler Softwareanwendungen (vgl. Evans, 2003, S. xxi).

Millet und Tune (2015) bezeichnen DDD als Philosophie, die darauf abzielt, die Komplexität der Problemdomänen beherrschbar zu machen, um so einen effizienten Entwicklungsprozess sicherzustellen (S. 13). Mit ihrer Definition machen die Autoren deutlich, dass es bei DDD um weit mehr geht als die bloße Anwendung von Methoden und Techniken. Den wahren Wert von DDD sehen sie in der engen Zusammenarbeit zwischen Fachexperten und Entwicklern, um gemeinsam zu einem verbesserten Verständnis der Domäne zu gelangen (vgl. Millett & Tune, 2015, S. 47).

Evans (2003) nennt zwei Prämissen für die Entwicklung von Software nach der DDD-Philosophie (S. xxi):

1. Für die meisten Softwareprojekte sollte der Fokus auf der Domäne und der Domänenlogik liegen.
2. Komplexe Domänen sollten auf einem Modell basieren.

Aus diesen beiden Aussagen leitet er die Entwurfsmuster, Prinzipien und Praktiken des DDD ab. Der Schlüssel dafür, die Komplexität der Domäne beherrschbar zu machen, ist ein Domänenmodell. Es steht im Zentrum des Entwicklungsprozesses (S. xxii) und dient nicht nur einer frühen Analyse (S. 46), sondern ist auch als Abstraktion der Implementierung die Basis aller Designentscheidungen und die

Grundlage der Kommunikation zwischen Entwicklern und Fachexperten (vgl. Evans, 2003, S. 4). Letztere werden dabei Domänenexperten genannt, da sie Experten auf dem Gebiet sind, in dem das zu lösende Problem angesiedelt ist (vgl. Millett & Tune, 2015, S. 18).

Neben dem Verständnis der Domänen ist das Verstehen und Beschreiben der zu lösenden Probleme wichtig. Voraussetzung dafür ist ein klar definierter Kontext, innerhalb dessen das Lösungsmodell gültig und die Terminologie aufgrund einer unmissverständlichen Sprache eindeutig ist (vgl. Millett & Tune, 2015, S. 4). Diese Sprache, *Ubiquitous Language* genannt, entwickelt sich im Laufe des Projekts. Sie wird sowohl als eine gemeinsam gesprochene Sprache zwischen Domänenexperten und Entwicklern verwendet als auch im Software-Modell implementiert (vgl. Vernon, 2015, S. 23).

DDD lässt sich in der Anwendung in ein strategisches Design und ein taktisches Design unterteilen. In diesen beiden Bereichen sind unterschiedliche Techniken einzusetzen (vgl. Vernon, 2015, S. 7; siehe auch Kap. 3.1.5 und 3.1.6).

Im Folgenden werden die wichtigsten Begriffe des DDD näher erläutert: Domäne, Domänenmodell, Knowledge Crunching, strategisches und taktisches Design, Domain Vision Statement. Am Ende des Kapitels wird auf die häufig vorkommenden Probleme in der Anwendung von DDD eingegangen.

3.2.2 Die Domäne

Das Unternehmen, dessen Arbeitsgebiete und die darin auftretenden Herausforderungen, wird durch Domänen abgebildet. Diese Domänen spiegeln typischerweise die Unternehmensstruktur wider. Eine Domäne (Domain) kann dazu in Subdomänen (Subdomains) aufgegliedert werden, um die Funktionsbereiche und die Beziehungen innerhalb der Domäne darzustellen. Die Unterteilung einer Domäne in Subdomains ist ein erster Schritt, um die Komplexität beherrschbar zu machen (vgl. Millett & Tune, 2015, S. 34-38). Zusätzlich lassen sich auch von der Struktur des Unternehmens unabhängige Kompetenz- oder Funktionsbereiche als Domänen beschreiben (vgl. Vernon, 2015, S. 43).

Sind die Aktivitäten innerhalb einer Domäne eine Schlüsselkompetenz des Unternehmens und tragen sie in bedeutendem Umfang zu dessen Wertschöpfung bei, wird diese Domäne als *Core Domain* bezeichnet. Die *Core Domain* hat im DDD eine besondere Bedeutung. Auf ihr sollte der Fokus liegen, da nicht alle Domänen von gleicher Relevanz für die Entwicklung des Softwaresystems sind. So können z. B. besonders erfahrene Entwickler für die Entwicklung der *Core Domain* eingesetzt werden (vgl. Millett & Tune, 2015, S. 35).

Eine *Subdomain*, die allgemeine Funktionalitäten oder Kompetenzen des Unternehmens darstellt, wird als *Generic Domain* bezeichnet (vgl. Evans, 2003, S. 406). Ein Beispiel dafür ist die Funktion eines E-Mail-Services. Sie ist kein zentraler Bestandteil des Unternehmens, aber zur Erfüllung des Unternehmenszweckes notwendig. Andere *Subdomains* helfen dem Unternehmen als *Supporting Domains* dabei, die *Core Domain* zu unterstützen. Ein Beispiel dafür wäre die Suchfunktion im Produktkatalog eines Unternehmens (vgl. Vernon, 2015, S. 52).

3.2.3 Das Domänenmodell

Evans (2003) definiert das Domänenmodell als organisierte und strukturierte Form des Wissens über ein Problem; repräsentiert werden die UL sowie die Beziehungen aller Entitäten innerhalb des Modells (S. 3 f.). Millet und Tune (2015) beschreiben das Domänenmodell als einen Entwurf, um die betrachteten fachlichen Anwendungsfälle erfüllen zu können (S. 42).

Das Domänenmodell kann sowohl in Form eines Diagramms oder eines Programmcodes dargestellt als auch schriftlich beschrieben werden (vgl. Evans, 2003, S. 37); Abbildung 1 zeigt ein Domänenmodell im Lösungsraum. Das Modell dient als Kommunikationsgrundlage für alle Projektbeteiligten und sollte daher von jedem verstanden werden. Aus diesem Grund ist es wichtig, dass das Modell in Zusammenarbeit zwischen Entwicklern und Domänenexperten im Prozess des Knowledge Crunchings (s. Kap. 2.3) erarbeitet wird (vgl. Millett & Tune, 2015, S. 19).

Im Gegensatz zu anderen Vorgehensweisen in der Softwareentwicklung verbindet DDD das Analyse-Modell mit dem Modell der Implementierung (*Code Model*) (vgl. Evans, 2003, S. 47 f.). Millet und Tune (2015) schließen sich dieser Sichtweise an und betonen, dass die strikte Umsetzung des Modells in die Implementierung wichtig ist. Nur durch diese enge Bindung können Abweichungen zwischen Analyse-Modell und Code Model verhindert werden (S. 43). Abbildung 1 zeigt den Übergang von Problemdomäne zum Domänenmodell im Lösungsraum.

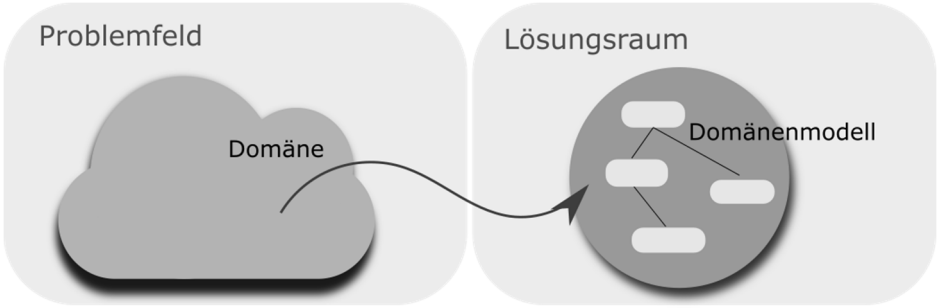


Abbildung 1: Die Domäne und das Domänenmodell [Eigene Darstellung in Anlehnung an (Millet & Tune, 2015, S. 43)]

3.2.4 Knowledge Crunching

Nur mit einem Verständnis für die fachlichen Aspekte der Probleme können die richtigen Design-Entscheidungen getroffen werden (vgl. Millett & Tune, 2015, S. 6 f.). *Knowledge Crunching* (sinnbildliche Übersetzung: Wissen jonglieren) ist der kontinuierliche Prozess, in dem die relevanten Informationen aus der Problemdomäne mit dem Ziel herausgearbeitet werden, ein Modell zu entwerfen, welches den Anforderungen der Stakeholder entspricht. DDD sieht vor, dass dieser Prozess in der Zusammenarbeit zwischen Domänenexperten und Entwicklern stattfindet (vgl. Evans, 2003, S. 14 f.).

Die dazu erforderlichen Informationen können unterschiedlichen Quellen entspringen. Sie können Ergebnisse eines Gesprächs oder einer Diskussion sein sowie aus Dokumentationen oder Erfahrungen mit ähnlichen Projekten stammen (vgl. Evans, 2003, S. 14). Auch das Festhalten der gewonnenen Er-

kenntnisse ist nicht an eine bestimmte Form gebunden. So können auch informelle, von Hand gezeichnete Schaubilder als Ergebnis festgehalten werden. Wichtig ist das Ausprobieren und Experimentieren, denn Modelle werden im Zuge des Knowledge Crunchings entworfen und verworfen (vgl. Millett & Tune, 2015, S. 20).

Durch das kontinuierliche Verfeinern des Modells werden die Entwickler dazu angehalten, ihr Wissen über die relevanten Geschäftsprozesse und Prinzipien des Unternehmens stetig zu erweitern (vgl. Vernon, 2015, S. 28). Dieser Prozess ist aber nicht nur für das Entwicklungsteam relevant. Auch die Domänenexperten profitieren davon, indem sie durch das Formulieren der essenziellen Aspekte ihr Wissen präzisieren und eventuell auf eigene Wissenslücken stoßen (vgl. Evans, 2003, S. 15).

Welche Auswirkungen das Knowledge Crunching auf bestehende Anforderungen haben kann, wurde bislang ebenso wenig betrachtet wie die Rolle des Stakeholders in diesem Prozess. Dabei ist es durchaus möglich, dass eine Diskussion mit den Domänenexperten zu neuen Anforderungen an die Software führt; wer die Verantwortung für die Änderungen übernimmt, bleibt bislang offen. Eine Lösung kann darin bestehen, den Stakeholder in den Prozess des Knowledge Crunchings einzubinden.

3.2.5 Strategisches Design

3.2.5.1 Die Bedeutung des strategischen Designs

Das strategische Design erarbeitet die Struktur des Lösungsraums (s. Abb. 1) und ist damit die Grundlage für weitere Designentscheidungen. Das Entwurfsmuster zur Abgrenzung der verschiedenen Domänenmodelle des Lösungsraums (s. Kap. 3.2.3) voneinander ist der *Bounded Context* (s. Kap. 3.2.5.2). Diese Abgrenzung führt zu einem klaren und gemeinsamen Verständnis davon, welche Domänenkonzepte in sich konsistent sind und wie sie sich von Domänenkonzepten anderer Kontexte unterscheiden (vgl. Evans, 2003, S. 335).

> *„Das Softwaremodell innerhalb der Kontextgrenze spiegelt eine eigene Sprache wider, die vom Team entwickelt wird, das in diesem Bounded Context arbeitet. Diese Sprache wird Ubiquitous Language (UL) genannt, weil sie sowohl von den Teammitgliedern gesprochen als auch im Softwaremodell implementiert wird“ (Vernon et al., 2017, S. 12).*

Um die UL entwickeln zu können, ist die enge Zusammenarbeit mit einem Domänenexperten notwendig. Er hat das nötige Wissen innerhalb einer Domäne.

Die Begriffe „Bounded Context“, „Ubiquitious Language“ und „Domänenexperte“ werden für ein besseres Verständnis im Folgenden erklärt.

3.2.5.2 Bounded Context

Große und komplexe Domänen enthalten meist mehrere Modelle (vgl. Vernon, 2015, S. 66). Im Idealfall kann jedes Modell einem eigenen Bereich der Problemdomäne, also einer Subdomain, zugeordnet werden, doch kann sich ein Modell auch über mehrere Subdomains erstrecken. Unabhängig von der

Anzahl der Modelle müssen diese Subdomains miteinander interagieren, um ein gewünschtes Systemverhalten zu erzeugen (vgl. Vernon, 2015, S. 67).

Zu Problemen kommt es, wenn kontextübergreifende Verflechtungen zwischen verschiedenen Modellen bestehen. Teilen sich die Modelle z. B. ein gemeinsames Domänen-Konzept, das in unterschiedlichen Kontexten eine andere Bedeutung hat, führt dies zu Konflikten in der zugrunde liegenden Logik und beeinflusst das Systemverhalten somit ungewollt (vgl. Vernon, 2015, S. 68). Abbildung 2 zeigt zwei Modelle, die sich das Konzept »Fahrzeug« teilen.

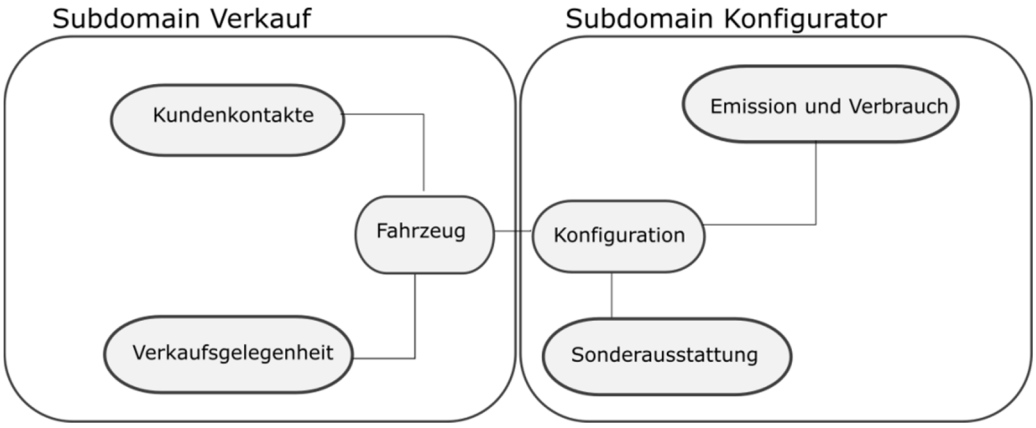


Abbildung 2: Das Domänen- Konzept »Fahrzeug« in der Subdomain „Verkauf" [Eigene Darstellung]

In der *Subdomain* »Verkauf« ist ein Fahrzeug Teil einer Verkaufsgelegenheit. Dabei sind die Attribute »Anzahl Kundenkontakt« oder »Verkaufsregion« von Interesse. In der Subdomain »Konfigurator« sind dagegen Konfigurationseigenschaften wie Lackfarbe, Sonderausstattung oder Emissionen für ein Fahrzeug wesentlich. Das Domänen-Konzept »Fahrzeug« hat folglich in den Subdomains eine andere Bedeutung und sollte in getrennten Modellen dargestellt werden.

Um die Integrität der Modelle zu schützen, müssen klare Grenzen zwischen den von ihnen abgedeckten Verantwortlichkeiten gezogen werden (vgl. Millett & Tune, 2015, S. 101). Dies wird erreicht, indem das Modell an einen bestimmten Kontext gebunden wird, den *Bounded Context* (s. Abb. 3). Er ist vor allem eine sprachliche Begrenzung: In ihm wird die sprachliche Eindeutigkeit der UL sichergestellt (vgl. Evans, 2003, S. 336).

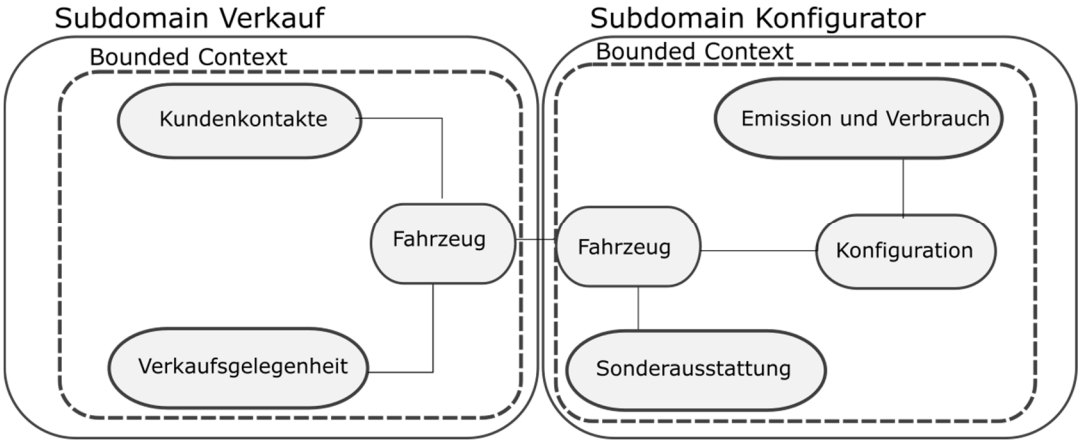


Abbildung 3: Das Domänen-Konzept »Fahrzeug« in verschiedenen Bounded Contexts [Eigene Darstellung]

In einer Subdomain können mehrere Bounded Contexts existieren (vgl. Vernon, 2015, S. 60). Wird z. B. ein 3-D-Visualisierungsdienst in die Applikation eingebunden, um das Fahrzeug nach der Konfiguration visuell darzustellen, kann dieser in einem eigenen Kontext in der *Subdomain* »Konfigurator« modelliert werden.

3.2.5.3 Ubiquitous Language

> *„Among all approaches to software development, Domain-Driven Design is the only one that is focused on language as the key tool for a deep understanding of a given domain's complexity."*
>
> Alberto Brandolini (2017)

Die gemeinschaftliche Sprache im DDD ist ein wesentliches Instrument, um ein gemeinsames Verständnis für die Problemdomäne bei den Teammitgliedern zu schaffen (vgl. Brandolini, 2017 S. 12). Nach Evans (2003) reift diese gemeinsame Sprache zwischen Entwicklern und Domänenexperten als *Ubiquitous Language* (UL) im Verlauf des Entwicklungsprozesses stetig heran (S. 25); Vernon (2015) ergänzt, dass jeder am Entwicklungsprojekt Beteiligte, unabhängig von seiner Rolle, diese UL verwenden sollte (S. 25). Millet und Tune (2015) bezeichnen die Entwicklung der UL als wichtigsten Aspekt im DDD. Ihrer Ansicht nach kann eine Zusammenarbeit ohne gemeinsame Sprache niemals effektiv sein (S. 11)

Die UL beschränkt sich nicht nur auf die gesprochene Ausdrucksform, sondern sollte bei allen im Rahmen des DDD erstellten Artefakten benutzt werden (vgl. Evans, 2003, S. 27).

Folglich werden die Begriffe der UL auch verwendet, um im implementierten Code Klassen, Methoden und Prozeduren zu benennen. Dadurch kann eine enge Bindung zwischen dem Analyse-Modell und der Implementierung erreicht werden – ein fundamentales Prinzip des DDD wird so umgesetzt (vgl. Millett & Tune, 2015, S. 48).

Evans (2011) führt mehrere Gründe auf, weshalb die UL für DDD von zentraler Bedeutung ist (S. 24 ff.):

- Domänenexperten haben oftmals ein eingeschränktes Verständnis von Konzepten und Aspekten der technischen Umsetzung.
- Entwickler kennen zu Anfang des Projekts die Domäne nicht oder nur teilweise.
- Domänenexperten sprechen hauptsächlich in der Sprache ihres Fachbereichs.
- Entwickler sprechen hauptsächlich in einem technischen Jargon.

Dies hat zur Folge, dass in Gesprächen und Diskussionen über entwicklungsrelevante Themen die Aussagen von Domänenexperten und Entwicklern jeweils übersetzt und interpretiert werden müssen. Das kann zu Fehlinterpretationen der Konzepte und zu Missverständnissen führen. Ein effektiver Wissenstransfer ist dabei nur bedingt möglich. Das Resultat ist das Fehlen eines tiefen gemeinsamen Verständnisses der zugrunde liegenden Probleme und Ideen, das für die Entwicklung einer zuverlässigen Software jedoch benötigt wird (vgl. Evans, 2003, S. 24 ff.).

Die UL ist eine Domänen-getriebene Sprache. Das bedeutet, dass sich die Begriffe aus der Fachlichkeit ableiten. Die Entwickler sollen sich in der Sprache der Domänenexperten mit diesen verständigen können (vgl. Millett & Tune, 2015, S. 48).

Die Vokabeln der UL sind Begriffe der Konzepte, Funktionen, Aufgaben und Geschäftsregeln, die das Domänenmodell beschreiben. Auch die obersten Prinzipien der Organisation finden sich in der UL wieder. Darüber hinaus gehören die verwendeten Begriffe des DDD selbst zum Wortschatz der UL (vgl. Vernon, 2015, S. 191). Durch das Experimentieren mit Begriffen und die Bildung von Alternativen bei der Erklärung von Domänen-spezifischen Inhalten entsteht und präzisiert sich die UL im Verlauf des Entwicklungsprozesses (vgl. Evans, 2003, S. 27).

In der DDD-Literatur bleibt allerdings ungeklärt, ob oder inwieweit sich ein Domänenexperte mit den Begriffen der Entwurfsmuster des taktischen Designs (s. Kap.3.2.6) auskennen muss, um mit dem Entwicklungsteam das Domänenmodell zu erarbeiten und zu analysieren. Da im DDD Diskussionen auf einer fachlichen Ebene geführt werden, sollte die vorzugsweise technische Sprache der Entwickler nicht durch eine technische Sprache des DDD ersetzt werden.

Offen bleibt auch, wie mit der Mehrsprachigkeit in Software-Entwicklungsprojekten umgegangen wird. Da die technische Sprache in der Softwareentwicklung meist Englisch ist und häufig sowohl im Code als auch in Dokumentationen und Anforderungsdokumenten benutzt wird, muss geklärt werden, wie die Begriffsfindung erfolgen soll, wenn die Fachsprache nicht Englisch ist.

3.2.5.4 Domänenexperten

Der Domänenexperte ist die Person, die am meisten über einem bestimmten Bereich des Unternehmens weiß – als erfahrener Anwender des Systems, Product Owner oder langjähriger Mitarbeiter des Betriebs. Er kennt die Geschäftsprozesse und Geschäftsregeln der Domäne mit allen ihren Nuancen und Ausnahmen (vgl. Millett & Tune, 2015, S. 18). Seine Zusammenarbeit mit dem Entwicklerteam ermöglicht ein gemeinsames Verständnis über die wichtigsten Konzepte der Domäne und den Entwurf eines Modells, welches dieses Wissen reflektiert (vgl. Millett & Tune, 2015, S. 136). Millet und Tune (2015) schlagen vor, dass das Entwicklungsteam und die Domänenexperten ihren Arbeitsplatz am gleichen Ort haben, damit Fragen sofort geklärt werden können und eine direkte Kommunikation jederzeit möglich ist (S. 136).

3.2.6 Taktisches Design

Das taktische Design umfasst die Umsetzung der Konzepte innerhalb eines Bounded Contexts. Verschiedene Entwurfsmuster wie z. B. *Entities*, *Value Objects* oder *Aggregates* dienen der Modellierung. Aus dieser kann dann der Code abgeleitet werden. Die Modellierung auf Basis der Entwurfsmuster ist damit die Brücke zur Implementierung (vgl. Millett & Tune, 2015, S. 308).

Entities und Value Objects bilden die Struktur des Domänenmodells und repräsentieren ein Konzept in der Problemdomäne (vgl. Millett & Tune, 2015, S. 427). Aggregate sind Entwurfsmuster in DDD, die

durch Abstraktion die technische Komplexität beherrschbar machen (vgl. Millett & Tune, 2015, S. 428). Da das taktische Design für das Requirements Engineering nicht relevant ist, wird es in dieser Arbeit nicht näher betrachtet.

3.2.7 Event-Storming

Event Storming ist eine Workshop-Methode, die das Ziel hat, ein Verständnis der Problemdomäne unter den Entwicklern und Domänenexperten zu schaffen (vgl. Millett & Tune, 2015, S. 405). Mit Post-its, Stiften und viel freier Fläche entwerfen Domänenexperten und Entwickler ein umsetzbares Domänenmodell. Das Ergebnis ist ein Softwareentwurf, der die fachlichen Erfordernisse vollständig berücksichtigt und die Kontextgrenzen des Systems nach DDD definiert (vgl. Brandolini, 2017, S. 2-6).

3.2.8 Domain Vision Statement

Durch die Umsetzung des strategischen und des taktischen Designs wird das Domänenmodell im Laufe des Entwicklungsprozesses geformt und gestaltet. In der Regel ist das Modell zu Beginn der Projektentwicklung noch nicht vorhanden. Gerade deshalb ist es wichtig, dass das Team bei der Ausarbeitung des Modells und des Codes selbst in die gleiche Richtung steuert.

Das Domain Vision Statement ist der Wegweiser, der die Richtung vorgibt. Es ist eine kurze Beschreibung der Fachlichkeit der *Core Domain* und der damit verbundenen Leistungsmerkmale und dient als Bindeglied zwischen Projektvision und den Details der *Core Domain* (vgl. Evans, 2003, S. 415).

3.2.9 Häufige Probleme bei der Umsetzung von Domain-Driven Design

3.2.9.1 Ursache von Problemen

Wird das DDD falsch umgesetzt, kann es einfache Probleme verkomplizieren und zu Frustration innerhalb des Teams führen (vgl. Millett & Tune, 2015, S. 121). Dabei bietet die Literatur keinen vollumfänglichen Überblick über häufig auftretende Probleme. Evans (2003) und Vernon (2015) beschreiben mögliche Probleme stets im Zusammenhang mit Themenblöcken. Eine gezielte Auseinandersetzung mit Problemen in DDD ist so nur unter hohem Aufwand möglich.

Lediglich Millet und Tune (2015) widmen der Thematik ein eigenes Kapitel, wobei sie sich dort auf solche Probleme beschränken, mit denen sich unerfahrene Teams bei einer ersten Nutzung von DDD konfrontiert sehen (S. 121). Generelle Probleme bei der Umsetzung, die unabhängig von der Erfahrung des Teams sind, werden nicht thematisiert.

Im Folgenden wird ein Überblick über häufig auftretende und im aktuellen Kontext relevante Probleme bei der Umsetzung von DDD gegeben.

3.2.9.2 Einbindung der Domänenexperten

Für das Entwicklungsteam ist es ohne die Zusammenarbeit mit Domänenexperten nicht möglich, das für die Systementwicklung benötigte Domänenwissen zu erlangen. Doch die Domänenexperten sind nicht immer vor Ort oder haben keine Zeit, ihr Wissen einzubringen, z. B., wenn sie viele Termine außerhalb des Unternehmens wahrnehmen müssen (vgl. Vernon, 2015, S. 28). Millet und Tune (2015) empfehlen, den Domänenexperten ernsthaftes Interesse entgegenzubringen, damit diese einen Wert darin erkennen, ihr Wissen zu teilen (S. 19).

Es ist allerdings fraglich, ob der Ausdruck von Interesse ausreicht, um einen Domänenexperten davon zu überzeugen, seine Zeit und sein Wissen in ein Projekt zu investieren, mit dem er nicht offiziell beauftragt wurde.

3.2.9.3 Domänenexperte und Business-Analyst

Oft wird anstelle eines Domänenexperten ein Business-Analyst in die Entwicklung integriert. Da sich Business-Analysten oftmals in verschiedenen Domänen auskennen müssen, ist ihr Wissen in der Problemdomäne häufig nicht tief genug. Die Folge ist, dass wichtige Aspekte der Domäne im Verborgenen bleiben (vgl. Millett & Tune, 2015, S. 125). Die Eignung eines Business-Analysten hängt dabei sicherlich von dessen Erfahrung ab. Hat ein Business-Analyst sich viele Jahre mit einer bestimmten Domäne beschäftigt, ist es durchaus möglich, dass er die Person mit der größten Erfahrung in dieser Domäne im Unternehmen ist und sich als Domänenexperte eignet (vgl. Vernon, 2015, S. 1).

3.2.9.4 Unzureichende Umsetzung von Domain-Driven Design

Die taktischen Entwurfsmuster des DDD (s. Kap.3.2.6) unterstützen die Erstellung des Domänenmodells. Fälschlicherweise wird häufig angenommen, dass das taktische Design der wichtigste Bestandteil des DDD ist. Dabei wird das strategische Design (s. Kap. 3.2.5) vernachlässigt (vgl. Millett & Tune, 2015, S. 122). Das Team nimmt sich häufig zu wenig Zeit dafür, das benötigte Fachwissen der Domäne zu erlernen, um so die verschiedenen kontextbildenden Sprachräume voneinander abzugrenzen, und setzt den Fokus stattdessen voreilig auf die technische Umsetzung des Modells. Die Folge ist die fehlende Abstraktion der Domänen-Konzepte und die Vernachlässigung der Entwicklung einer gemeinsamen Sprache. Diese unvollständige Umsetzung von DDD, bei der sich der Wert des DDD, der auf der engen Zusammenarbeit, der Entwicklung der *UL* (s. Kap. 3.2.5.3) und dem Bounded Context (s. Kap. 3.2.5.2) beruht, nicht entfalten kann, wird als DDD Lite bezeichnet (vgl. Millett & Tune, 2015, S. 123).

3.2.9.5 Trennung von Analyse-Modell und Implementierung

Die enge Bindung zwischen dem Modell der Analyse und dem Modell des implementierten Codes ist ein Prinzip von DDD. Diese Bindung kann nur durch die enge Zusammenarbeit zwischen Domänenexperten und dem Entwicklungsteam erzielt werden.

In konventionellen Vorgehensmodellen werden die Anforderungen jedoch häufig lange vor der Implementierung aufgenommen und dokumentiert. Die Requirements Engineers sprechen mit den Domänenexperten und Architekten und erstellen mit den Informationen ein Analyse-Modell. Dieses Modell wird anschließend dem Entwicklungsteam zur Umsetzung übergeben. Die Folge ist, dass die Entwickler in der Umsetzung oft vom Analyse-Modell abweichen und wichtige Begriffe und Konzepte übersehen. Mit der Weiterentwicklung des Systems weicht die Umsetzung immer mehr vom Analyse-Modell ab. Sind in einem späteren Stadium der Entwicklung Änderungen oder Erweiterungen erforderlich, führt die Divergenz der Artefakte zu Problemen (vgl. Millett & Tune, 2015, S. 44 f.).

Wenn die Entwickler sich nicht als für das Analyse-Modell zuständig begreifen oder nicht wissen, wie das Modell erstellt wurde, ergeben sich am Ende der Entwicklung Divergenzen zwischen dem Modell und der entwickelten Software. Wenn andererseits der Ersteller des Modells isoliert von Fragestellungen der Implementierung vorgeht, wird er mögliche Einschränkungen derselben nicht antizipieren und berücksichtigen können. Dies hat zur Folge, dass das Domänenwissen und die Erfahrung der Experten nicht in die Entwicklung einfließen. Daher sollte sich jede Person, die an der Erstellung des Domänen-Modells beteiligt ist, eine gewisse Zeit mit dem Code auseinandersetzen, und zwar unabhängig von ihrer Rolle im Projekt. Doch auch jeder Entwickler sollte sich Kenntnisse darüber aneignen, wie das Modell durch Code ausgedrückt wird (Evans. 2003, S. 61 f.).

3.3 Requirements Engineering

3.3.1 Theoretische Verortung

> *„There are lots of people who believe their job is collecting and communicating requirements. But it´s not. The truth is, your job is to change the world.“*
>
> Jeff Patton (2014)

Nach der Definition von Pohl (1986) hat das Requirements Engineering (RE) die Aufgabe, die Anforderungen der Stakeholder zu ermitteln, zweckmäßig zu dokumentieren, zu überprüfen und abzustimmen sowie die dokumentierten Anforderungen über den gesamten Lebenszyklus des Systems hinweg zu verwalten. Er bezeichnet das RE als „systematischen Ansatz“ (S. 10).

Für Robertson und Robertson (2013) hingegen ist das RE ein iterativer Prozess, bei dem die Anforderungen entdeckt, verifiziert und dokumentiert werden. Ziel ist, die Geschäftsprobleme zu verstehen und Lösungsmöglichkeiten zu entwickeln (S. 10).

Vergleicht man diese Definition mit der von Pohl (1986) wird deutlich, dass sich mit der Etablierung von agilen Ansätzen in der Softwareentwicklung auch die Rolle des RE verändert hat. Robertson und Robertson (2013) betonen, dass sich dabei nicht die Aufgaben des RE geändert haben, sondern lediglich die Vorgehensweise in deren Durchführung (S. xxi).

Auch Hruschka (2014) beschreibt das RE als iterativen und inkrementellen Prozess. Bei diesem Vorgehen wird die gesamte Systementwicklung in wiederholt zu durchlaufende Arbeitsschritte (Iterationen) unterteilt und das System in Teilstücken (Inkrementen) realisiert. Jedes Inkrement bildet eine Erweiterung an Funktionalität (S. 14).

Wie durch das einleitende Zitat von Patton (2014) plakativ hervorgehoben, misst dieser dem RE eine hochfliegende Bedeutung zu (S. xliii). Dies erklärt er im Weiteren damit, dass der Erfolg der Softwareentwicklung nicht am Entwicklungsergebnis gemessen werden sollte, sondern daran, ob das Ergebnis einen Mehrwert für den Kunden darstellt (S. xxxix). Hier wird die starke Kundenorientierung deutlich, die vor allem bei agilen Vorgehensweisen im Fokus steht.

Das RE kann in zwei Bereiche unterteilt werden, die Requirements-Analyse und das Requirements-Management (vgl. Hruschka, 2014, S. 7). Gegenstand der Requirements-Analyse ist, durch ein systematisches Vorgehen Anforderungen zu ermitteln, zweckmäßig zu dokumentieren, zu überprüfen und abzustimmen. Die Verwaltung der Anforderungen über den gesamten Lebenszyklus hinweg fällt unter den Begriff des Requirements-Management und ist kein Gegenstand dieser Arbeit (vgl. Hruschka, 2014, S. 14 f.).

Im Folgenden wird zunächst spezifiziert, was der Begriff „Anforderungen“ enthält und wie diese grundsätzlich erhoben werden. Danach wird die Bedeutung der Produktvision aufgezeigt. Erläutert wird anschließend die Rolle der Roadmap als „Ablaufplan“. Näher Beleuchtet werden dann die Ermittlung, die Dokumentation und die Prüfung der Anforderungen sowie deren Abstimmung mit den Kundenwünschen. Auch das Konfliktmanagement wird beschrieben.

3.3.2 Anforderungen

3.3.2.1 Definition

Nach Robertson und Robertson (2013) sind Anforderungen (Requirements) das, was eine Software, ein Hardware-Produkt oder ein Service leisten oder ermöglichen soll; sie existieren unabhängig davon, ob sie bereits erhoben oder dokumentiert wurden (S. 1). Patton (2014) definiert Anforderungen dagegen sehr allgemein als Ideen, die anderen Menschen helfen (S. 4). Dagegen spricht Hruschka (2014) von Anforderungen erst, sobald diese formalisiert und als solche identifizierbar sind (S. 11).

Die vorliegende Arbeit schließt sich der Definition von Hruschka an, da diese dem internationalen ISO/IEC/IEEE-Standard 29148 entspricht (vgl. Pohl & Rupp, 2015, S. 54).

3.3.2.2 Zweck einer Anforderung

Anforderungen wirken unmittelbar auf den Systementwicklungsprozess ein und bilden die Basis für viele weitere Schritte wie Ausschreibung und Vertragsverhandlung, Systemarchitektur oder Systemintegration, Test und Abnahme sowie Wartung. Sie dienen allen am Systemprozess Beteiligten als Kom-

munikations-, Diskussions- und Argumentationsgrundlage. Ihr Zweck ist es, das gemeinsame Verständnis und Wissen der Teammitglieder widerzuspiegeln und zu explizieren (vgl. Rupp, 2014, S. 16). Ziel des Anforderungsprozesses ist das Verstehen von Geschäftsproblemen und die Entwicklung von Lösungsmöglichkeiten (Robertson & Robertson, 2013, S. 1).

3.3.2.3 Arten von Anforderungen

Es existieren verschiedene Arten von Anforderungen und unterschiedliche Ansätze zu ihrer Klassifizierung. Ein verbreiteter Ansatz ist die Unterteilung in funktionale und nichtfunktionale Anforderungen.

Nach Rupp (2014) ist eine funktionale Anforderung „eine Anforderung bezüglich des Ergebnisses eines Verhaltens, das von einer Funktion des Systems (oder einer Komponente eines Services) bereitgestellt werden soll“ (S. 17). Nach Robertson und Robertson (2013) beschreiben funktionale Anforderungen, was das Produkt tun muss, um die Arbeit des Kunden zu ermöglichen oder zu unterstützen. In diesen Definitionen spiegeln sich zwei verschiedene Paradigmen der Softwareentwicklung wider: die Produkt- und die Kundenorientierung. Bei Robertson und Robertson (2013) steht der Kunde im Fokus der Anforderungen, während dieser in der Definition nach Rupp (2014) nicht erwähnt wird.

Nichtfunktionale Anforderungen beschreiben die qualitativen Eigenschaften dieser Funktionen sowie die technischen und organisatorischen Vorgaben, die es zu berücksichtigen gilt (vgl. Hruschka, 2014, S. 203). Das International Requirements Engineering Board (IREB) unterteilt für eine spezifischere Unterscheidung nichtfunktionale Anforderungen in Qualitätsanforderungen und Randbedingungen (vgl. Hruschka, 2014, S. 13).

3.3.3 Grundlagen

3.3.3.1 Bedeutung der Grundlagen

Bevor die Anforderungen an die Software erhoben werden können, müssen zu Beginn des Projekts die verschiedenen Anforderungsquellen identifiziert werden. Dies bildet die Grundlage für alle weiteren Schritte des RE-Prozesses.

In der initialen Phase werden die Ziele des Projekts festgelegt, es wird Wissen über den Systemkontext und den Scope (Umfang) erarbeitet sowie Kontakt zu allen relevanten Stakeholder hergestellt (vgl. Rupp, 2014, S. 47).

In der Praxis wird ein Requirements Engineer allerdings nicht immer unmittelbar zu Projektbeginn beauftragt, sondern oft erst zu einem späteren Zeitpunkt tätig. Hier muss geklärt werden, wie er mit den Grundlagen umzugehen hat, wenn diese in mangelnder Qualität vorliegen oder überhaupt nicht vorhanden sind.

Im Folgenden werden die relevanten Begriffe der Grundlagenermittlung vorgestellt. Die Begriffe „Ziele“, „Stakeholder“ sowie „Scope und Systemkontext“ werden näher beschrieben.

3.3.3.2 Ziele

Ziele sind die höchste und abstrakteste Form von Anforderungen. Alle weiteren, detaillierteren Anforderungen sollten sich aus ihnen ableiten (vgl. Hruschka, 2014, S. 30).

Langfristige Ziele sollten sich während der Projektentwicklung nicht ändern. Bei kurzfristigen Zielen handelt es sich oft um Release-Ziele oder, sofern mit Scrum gearbeitet wird, um Sprint-Ziele (siehe Kap. 3.4 für eine Erklärung des Scrum-Frameworks). Diese werden zu Anfang jedes Sprints oder Releases formuliert (vgl. Hruschka, 2014, S. 31)

Ziele müssen SMART sein. S steht dabei für spezifisch, M für messbar, A für angemessen, für R für realistisch und T für terminiert (vgl. Hruschka, 2014, S. 33).

- Spezifisch
  Ein Ziel soll spezifisch, also konkret, eindeutig und präzise formuliert werden. Denn ein Ziel ist kein vager Wunsch.
- Messbar
  Um zu überprüfen ob das Ziel erreicht wurde, muss ein Ziel messbar sein. Bei quantitativen Ziele ist das relativ einfach. Schwerer fällt es bei qualitativen Zielen.
- Angemessen
  Ein Ziel sollte herausfordernd sein. Es ist attraktiv und angemessen zu formulieren, damit es vom Team akzeptiert wird. Ein Ziel, dass den Teammitgliedern widerstrebt, ist mit großer Wahrscheinlichkeit gar nicht oder nur mit großer Mühe zu erreichen.
- Realistisch
  Das Ziel muss prinzipiell erreichbar sein. Scheint ein Ziel unerreichbar, wirkt dies demotivierend auf die Beteiligten.
- Terminiert
  Zu jedem Ziel gehört ein klarer Termin, bis wann es erreicht sein soll. Wenn ein Ziel nicht terminiert ist, besteht die Gefahr, das Ziel hinauszuschieben.

Ziele sind die Ausgangsbasis für die anschließende Anforderungsanalyse. Sie müssen sorgfältig formuliert werden, damit alle am Projekt Beteiligten auf ein gemeinsames Ziel hinarbeiten können. Nicht dokumentierte Ziele sind außerdem nicht verfolgbar und werden daher meist nicht erreicht (vgl. Rupp, 2014, S. 74 f.). Falls die Ziele zu Projektbeginn noch nicht vorliegen, ist es die Aufgabe des Requirements Engineers, die Ziele zu formulieren (vgl. Hruschka, 2014, S. 30). Die Schwierigkeiten, die mit einer Zieldefinition in einem fortgeschrittenen Stadium der Projektentwicklung einhergehen, sind im Folgenden aber nicht Gegenstand der Betrachtung.

Wenn verschiedene Projektbeteiligte bei einer uneindeutigen Festlegung ihre eigene Definition der Ziele verinnerlicht haben, muss das Team zu einer konsistenten Ansicht kommen. Als hilfreich erwiesen hat sich in diesem Fall die Methode PAM: Die Abkürzung ist ein Akronym und setzt sich aus den Begriffen *Purpose*, *Advantage* und *Measure* zusammen (vgl. Robertson & Robertson, 2013, S. 55).

Mit *Purpose* wird ein Satz bezeichnet, der beschreibt, was das System tun soll, also z. B.: „*Das System soll dem Benutzer die Möglichkeit geben, Kundendaten in seinen Sprachdialekt übersetzen zu könne.*“ Unter *Advantage* werden alle Vorteile und die Vorteilsempfänger aufgelistet. Vorteile könnten z. B. in einer Kostenreduktion, einer Serviceverbesserung oder in der Erfüllung von gesetzlichen Erfordernissen bestehen. *Measure* definiert die Metriken, mit denen am Ende des Projekts, des Releases oder des Sprints festgestellt werden kann, ob das Ziel erreicht wurde. Ein Projekt sollte drei bis fünf Ziele enthalten, die gemäß PAM definiert werden (vgl. Hruschka, 2014, S. 33). Im Rahmen der agilen Softwareentwicklung wird im Zielfindungsprozess häufig die Produktvision (s. Kap. 3.3.3.5) definiert (vgl. Rupp, 2014, S. 78).

PAM eignet sich gut für Workshop-Formate. Die Beteiligten schreiben ihre Definition der Ziele auf eine Karte und befestigen diese an einer Pinnwand. In der Runde werden die Ziele gemeinsam vorgestellt und besprochen. So lässt sich prüfen, ob die Beteiligten die gleiche Zielvorstellung haben (vgl. Robertson & Robertson, 2013, S. 56 f.).

3.3.3.3 Stakeholder

Stakeholder sind alle Personen und Organisationen, die ein Interesse an oder einen Effekt auf Systementwicklung oder Einsatz und Betrieb des Systems haben (vgl. Robertson & Robertson, 2013, S. 44; Rupp, 2014, S. 79). Ein Stakeholder hat direkten oder indirekten Einfluss auf die Anforderungen des betrachteten Systems (vgl. Pohl & Rupp, 2015, S. 21).

Dabei existieren projektnähere und projektfernere Stakeholder (vgl. Hruschka, 2014, S. 35). Beispiele für projektnähere Stakeholder sind Entwickler, Software-Architekten, Sicherheitsexperten, der Auftraggeber, Domänenexperten, Tester oder die Nutzer des Systems. Projektfernere Stakeholder sind z. B. Regierungen, Mitbewerber, Auditoren oder Umweltaktivisten (vgl. Hruschka, 2014, S. 35).

Die Stakeholder können zu Beginn des Projekts durch ein Brainstorming identifiziert werden. Folgende vier Fragen sollten dabei beantwortet werden (vgl. Hruschka, 2014, S. 26):

1. Wer hat Interesse (Name und Rolle)?
2. Was können Sie beitragen? Was wissen sie?
3. Welchen Einfluss haben sie? (Wichtigkeit für das Projekt)
4. Was sind ihre Interessen?

Die gewonnenen Informationen sollten systematisch dokumentiert werden (vgl. Rupp, 2014, S. 80). Ein Stakeholder-Liste kann z. B. einfach während des Projekts angepasst und erweitert werden (vgl. Hruschka, 2014, S. 38).

Der Stakeholder ist die wichtigste Quelle zur Identifikation von Anforderungen (vgl. Pohl & Rupp, 2015, S. 22). Er unterscheidet sich vom Domänenexperten, denn er kommuniziert sowohl die Geschäftsziele als auch seine Erwartungen an die Funktionalität des zu entwickelnden Systems und fokussiert dabei auf die Ein- und Ausgabedaten. Millet und Tune schlagen vor, diese Anforderungen in Zusammenarbeit

zwischen den Entwicklern und dem Stakeholder in Form von Business Use Cases (s. Kap. 3.3.4.4) zu erfassen. Diese Business Use Cases sind Grundlage dafür, mit den Domänenexperten anschließend ein Domänenmodell zu entwickeln, das die Anforderungen der Business Use Cases erfüllt (vgl. Millett & Tune, 2015, S. 18).

3.3.3.4 Scope und Systemkontext

Der Scope (Systemumfang) eines Systems ist der Bereich, der bei der Entwicklung eines Systems bewusst gestaltet und beeinflusst wird (vgl. Hruschka, 2014, S. 42). Rund um diesen Scope befindet sich der Systemkontext, die relevante Umgebung des zu analysierenden Systems (vgl. Rupp, 2014, S. 85). Relevant ist der Kontext deshalb, weil er Auswirkungen auf das zu erstellende System haben kann (vgl. Hruschka, 2014, S. 42).

Um den Systemkontext zu definieren, ist es notwendig, ihn gegen das zu entwickelnde System und die irrelevante Umgebung – also den Teil der Umgebung, der für das neue System keine Rolle spielt – abzugrenzen. In der Kontextabgrenzung wird analysiert, welche Aspekte der Umgebung Beziehungen zum geplanten System haben (vgl. Rupp, 2014, S. 86 f.).

Die wichtigste Abgrenzung ist die des geplanten Systems gegen den Kontext. Die Systemgrenze trennt den Scope vom Systemkontext und legt fest, welche Aspekte dem geplanten System zu eigen sind und welche Aspekte der Umgebung des Systems angehören (vgl. Hruschka, 2014, S. 43).

3.3.3.5 Produktvision

Die Produktvision ist die Grundlage für die Motivation des Teams, auf ein Ziel hinzuarbeiten, und sollte deshalb zu Beginn des Projekts entwickelt und kommuniziert werden. Eine gut kommunizierte Vision löst in Softwareentwicklungsprojekten das Problem der Unsicherheit darüber, ob man noch in die richtige Richtung entwickelt (vgl. Gloger, 2016, S. 78).

Bei agilen Herangehensweisen werden die Grundlagen in der initialen Phase nicht vollständig spezifiziert. Die Kommunikation der Vision hilft dem Projektteam dabei, ein gemeinsames Verständnis des Systems zu entwickeln und die Zielsetzung des Projekts festzulegen (vgl. Rupp, 2014, S. 508). Dabei geht die Bedeutung der Vision über die initiale Phase der Produktentwicklung hinaus. Sie ist für das Entwicklungsteam entscheidend, denn wenn – wie in der agilen Softwareentwicklung üblich – keine detaillierten Anforderungsdokumente vorhanden sind, muss das Team wissen, was entwickelt werden soll (vgl. Leffingwell, 2011, S. 252).

Die Vision kommuniziert die strategische Absicht des zu entwickelnden Produkts und beantwortet folgende Fragen (vgl. Leffingwell, 2011, S. 252):

- Warum entwickeln wir das Produkt, das System oder die Applikation?
- Welches Problem löst es?
- Welche *Features* mit welchem Nutzen stellt es zur Verfügung?

- Wem nützt das System?
- Welche Performance, Zuverlässigkeit und Skalierbarkeit muss es erfüllen?
- Welche Pattformen, Standards und Applikationen unterstützt es?

Bei agilen Herangehensweisen kann sich die Vision im Laufe der Projektentwicklung verändern, sollte aber mindestens bis zum Ende eines strategischen Zeitrahmens gelten. Jede Änderung der Vision hat Einfluss auf die Projektentwicklung und muss kommuniziert werden, um die Motivation der Projektbeteiligten zu erhalten. Aus diesem Grund ist es wichtig, die Produktvision regelmäßig zu kommunizieren. (vgl. Kittlaus & Fricker, 2017, S. 52).

Die Kommunikation der Vision kann auf unterschiedliche Arten erfolgen und hängt von der Präferenz der Organisation ab. In einem Visionsdokument können z. B. auf fünf bis zehn Seiten die wichtigsten Fragen zur Vision dokumentiert werden (Leffingwell, 2011, S. 253). Zusätzlich sollte die Vision eine Reihe von priorisierten *Features* enthalten.

Die Produktvision ist zeitunabhängig formuliert, denn die Frage nach dem „Was" steht dabei im Vordergrund. Soll zur Planung und Priorisierung der zeitliche Aspekt berücksichtigt werden, bietet sich die Erstellung einer Roadmap an.

3.3.4 Ermittlung der Anforderungen

3.3.4.1 Bedeutung der Anforderungsermittlung

Nachdem die Grundlagen er- und die Produktvision vermittelt wurden, können die Anforderungen an das System erhoben werden. Dies ist oft der schwierigste Schritt der Softwareentwicklung, da hier die Entscheidung darüber fällt, was entwickelt werden soll. Die iterative Erarbeitung und Verfeinerung der Anforderungen ist dabei die wichtigste aller konzeptionellen Tätigkeiten im Entwicklungsprozess (Brooks, 1986, S. 13).

Eine fehlerhafte Anforderungsanalyse führt zu Problemen, die nachträglich nur schwer zu beheben sind. Sie ist daher eine der Hauptursachen für das Scheitern des Systems sowie für hohe Entwicklungskosten (vgl. Brooks, 1986, S. 13; Laudon, Laudon & Schoder, 2010, S. 259).

Das Ziel der Anforderungsermittlung ist es, die Ziele und Anforderungen mit möglichst geringem Aufwand und angepasst an die Rahmenbedingungen des Projekts zu erfassen (vgl. Rupp, 2014, S. 80). Dazu existieren verschiedene Techniken; einige gebräuchliche Methoden werden im Folgenden näher beschrieben.

3.3.4.2 Modellbasierte Techniken

Bei modellbasierten Techniken dient ein Modell als Grundlage zur Ermittlung der Anforderungen (vgl. Nuseibeh & Easterbrook, 2000, S. 40). So wird z. B. bei zielorientierten Methoden des RE in der frühen

Phase der Anforderungsanalyse ein Zielmodell erstellt, aus dem sich die funktionalen und nichtfunktionalen Anforderungen ableiten lassen. Die Ziele der Stakeholder begründen und rechtfertigen die Funktionalitäten und Features des Systems (vgl. Pohl et al., 2012, S. 55). Neben den Zielen werden die Beziehungen zwischen dem zu entwickelnden System und seiner Umwelt dargestellt. Auf diese Weise kann der Systemkontext erfasst werden. Die Vorteile aus einer angemessenen Formulierung von Zielen sind deren Lösungsneutralität und das frühzeitige Erkennen von Abweichungen zwischen den Zielvorstellungen unterschiedlicher Stakeholder (vgl. Pohl et al., 2012, S. 52).

Anstelle formaler Modelle kann auch die schnelle und einfache Modellierung von aktuellen und zukünftigen Aktivitäten und Prozessen zur Anforderungsermittlung genutzt werden. Diese Art der Modellierung simuliert die Realität und kann mithilfe von Klebezetteln, Karteikarten oder einfachen Zeichnungen an Whiteboards, Flipcharts oder auf anderen Flächen durchgeführt werden. Bei dieser Methode werden die Stakeholder aktiv in die Modellierung eingebunden. Durch das Verschieben von Klebezetteln lassen sich Änderungen schnell und einfach vornehmen (vgl. Robertson & Robertson, 2013, S. 107 f.).

3.3.4.3 Workshop-Methoden

Bei konventionellen Projekten können Stakeholder, die über das notwendige Fachwissen und über Entscheidungskompetenzen verfügen, in einem Workshop gemeinsam Anforderungen erarbeiten.

Abbildung 4: Story Map (Patton, 2014, S. xxi)

Für agile Entwicklungsprojekte eignet sich das *User-Story Mapping* (vgl. Patton, 2014, S. 1). Eine Story-Map arrangiert User Stories zu einem Modell, das beschreibt, was der Kunde mit dem zu entwickelnden Produkt unternimmt. Dazu gehören alle Aktivitäten und Aufgaben, die er dabei durchführt oder erledigen muss. Das Ziel dieser Workshop-Methode ist, ein gemeinsames Verständnis unter den Projektbeteiligten sowie einen Gesamtüberblick über das gewünschte Ergebnis zu schaffen (vgl. Patton, 2014, S. 3).

3.3.4.4 Business Use Case

Ein *Business Use Case* (BUC) beschreibt die Geschäftsprozesse und Interaktionen eines Unternehmens mit externen Akteuren, wie dessen Kunden oder Partnern (vgl. Hruschka, 2014, S. 78). Er ist eine funktionale Einheit und die Basis für funktionale und nichtfunktionale Anforderungen (vgl. Robertson & Robertson, 2013, S. 80). In BUC werden die Prozesse nicht in der Tiefe betrachtet, sondern zunächst nur mit den Akteuren in Verbindung gebracht. In einem anschließenden Schritt können dann die Prozesse durch eine BUC-Beschreibung oder Diagramme abgebildet werden (vgl. Rupp, 2014, S. 170).

Die Anwendung von BUC kann sich anbieten, um das Geschäftsproblem, welches das zu entwickelnde System lösen oder vereinfachen soll, verstehen zu lernen (Robertson & Robertson, 2013, S. 69), denn der BUC ist die beabsichtigte Reaktion auf einen *Business Event* (Geschäftsereignis) (Robertson & Robertson, 2013, S. 67). Ein Business Event ist ein Ereignis innerhalb eines Unternehmens, das Auswirkungen auf dieses Unternehmen hat und eine Reaktion erfordert (vgl. Robertson & Robertson, 2013, S. 78). Elemente des Business Use Case sind unter anderem die beteiligten Business-Akteure und das BUC-Diagramm, das die bestehenden Abhängigkeiten aufzeigt.

3.3.4.5 Essenzbildung

Die Anforderungsermittlung wird oft dadurch erschwert, dass die Stakeholder Lösungen formulieren, ohne das zugrundeliegende Problem zu benennen. Das Resultat ist ein Software-Produkt, das zwar den Anforderungen des Kunden entspricht, das eigentliche Geschäftsproblem aber nicht löst. Um diese Fehlentwicklung abzuwenden, muss der Requirements Engineer die Wünsche und Forderungen des Kunden interpretieren (vgl. Robertson & Robertson, 2013, S. 150 f.). Die Essenzbildung ist eine Technik, welche die Anforderungsermittlung durch Abstraktion unterstützt (Rupp, 2014). Die Abstraktion löst die Aspekte der Implementierung vom Kern des Problems und macht somit die zugrunde liegende Essenz sichtbar (vgl. Robertson & Robertson, 2013, S. 153). Das Ziel der Abstraktion sind technologie- und lösungsneutrale Anforderungen (vgl. Rupp, 2005, S. 117).

Bei der Erhebung von Anforderungen werden oft die aktuellen Arbeitsabläufe einschließlich aller ehemals getroffenen technologischen Ablaufentscheidungen dokumentiert. Bevor darauf basierend eine Neuentwicklung eines Systems beginnen kann, sind die Abläufe auf ihre fachliche Grundlage, ihre Essenz, zurückzuführen (vgl. Rupp, 2005, S. 117). Durch die Beschränkung auf das Wesentliche wird die Komplexität der Systembeschreibung reduziert (vgl. Robertson & Robertson, 2013, S. 153).

Vielen Stakeholdern fällt es schwer, die Essenz zu ermitteln, da sie sich gedanklich meist im Konkreten bewegen. Ein externer Analytiker ist häufig hilfreich, da er den notwendigen Abstand zur Materie besitzt (Rupp, 2015, S. 118).

3.3.5 Dokumentation von Anforderungen

Es gibt verschiedene Gründe, warum die Dokumentation von Anforderungen nützlich sein kann. So kann z. B. eine Dokumentation Wissen vor dem Vergessen bewahren oder dabei helfen, Missverständnisse zu vermeiden, indem dieselbe Informationsquelle als Grundlage zur Kommunikation zwischen den Stakeholdern herangezogen wird (vgl. Rupp, 2014, S. 19-23).

Robertson und Robertson (2013) verwenden den Begriff der Dokumentation nicht, sondern behandeln das Thema unter dem Titel »Kommunikation von Requirements« (S. 353). Damit unterstreichen die Autoren ihre Auffassung, dass der Hauptzweck einer Dokumentation die Kommunikation der Anforderungen in Richtung Entwicklungsteam darstellt (vgl. Robertson & Robertson, 2013, S. 353).

Für die Dokumentation von Anforderungen werden verschiedene Techniken eingesetzt (vgl. Rupp, 2014, S. 168).

> *„Dabei gilt jegliche Art der mehr oder weniger formalen Darstellung, die die Verständigung zwischen den einzelnen Stakeholdern erleichtert und die Qualität der dokumentierten Anforderung erhöht“*(Pohl & Rupp, 2015S. 35).

Die Dokumentationstechniken bedienen sich dabei der natürlichen Sprache oder modellbasierter Varianten. Die natürliche Sprache hat den Vorteil, dass sie von jedem Stakeholder gelesen und verstanden werden kann (vgl. Rupp, 2014, S. 187) – sofern alle diese Sprache beherrschen. Ein Nachteil der natürlichen Sprache besteht zudem darin, dass sie unterschiedlich interpretierbar ist. Dies birgt die Gefahr von Missverständnissen (vgl. Pohl & Rupp, 2015, S. 137).

3.3.6 Abstimmung von Anforderungen

Das Ziel der Abstimmung von Anforderungen ist es, unter den beteiligten Stakeholdern ein gemeinsames und übereinstimmendes Verständnis bezüglich der Anforderungen an das zu entwickelnde System zu schaffen (vgl. Pohl & Rupp, 2015, S. 69). Zur Abstimmung dieser Anforderungen ist es notwendig, Konflikte zu identifizieren und diese zu lösen. Werden sie nicht gelöst, drohen Akzeptanzprobleme bzw. können nicht alle Anforderungen umgesetzt werden (vgl. Pohl & Rupp, 2015, S. 112).

Das Konfliktmanagement im RE umfasst folgende Aufgaben:

Konfliktidentifikation

Konflikte können zu jeder Zeit im RE-Prozess auftreten. Diese reichen von einfachen sachlichen Auseinandersetzungen bis hin zu schweren persönlichen Konflikten (vgl. Rupp, 2014, S. 348).

Konfliktanalyse

In der Konfliktanalyse werden die identifizierten Konflikte auf ihre Ursachen hin untersucht. Dies ist notwendig, um im nächsten Schritt die Konflikte auflösen zu können.

Es existieren verschiedene Konflikttypen (vgl. Rupp, 2014, S. 356):

- Sachkonflikte
  Sie sind durch einen Mangel an Informationen, durch Fehlinformationen oder durch unterschiedliche Interpretationen einer Information gekennzeichnet (vgl. Pohl & Rupp, 2015, S. 113).
- Interessenkonflikt
  Die Interessen zweier Stakeholder sind nicht miteinander vereinbar.
- Wertekonflikt
  Es kommt zu Reibungen aufgrund unterschiedlicher Wertesysteme der Stakeholder.
- Beziehungskonflikte
  Sie entstehen hauptsächlich durch Uneinigkeit über soziale und hierarchische Beziehungen oder durch negatives zwischenmenschliches Verhalten der Stakeholder untereinander.
- Strukturkonflikt
  Sind durch ungleiche Macht- und Autoritätsverhältnisse gekennzeichnet.
- Rollenkonflikte
  Sie entstehen aufgrund von Rollen, die einzelne Personen im Unternehmen und im Team spielen, und resultieren auch aus deren Funktion innerhalb des Projekts. Zu einem Rollenkonflikt kommt es, wenn z. B. der Product Owner gleichzeitig in der Rolle des Scrum Masters tätig ist.

Konfliktauflösung

Die Konfliktauflösung hat einen großen Einfluss auf die weitere Kooperationsbereitschaft der Konfliktparteien. Wird eine Konfliktlösung von einer Partei als unfair empfunden, kann das dazu führen, dass das Engagement für das geplante System sinkt oder die Entwicklung vom Stakeholder nicht mehr unterstützt wird (vgl. Pohl & Rupp, 2015, S. 114).

3.4 Scrum

3.4.1 Allgemeine Grundlagen

Scrum (englisch für „Gedränge“) ist die Bezeichnung für ein Vorgehensmodell im Projekt- und Produktmanagement und entspricht der Umsetzung eines Lean-Development-Ansatzes. Es wurde ursprünglich in der Softwaretechnik entwickelt und wird insbesondere bei der agilen Softwareentwicklung eingesetzt.

Im Folgenden werden die wichtigsten Begriffe zu Scrum erklärt. Abschließend wird das Scrum Framework[1] sowohl mit RE als auch mit DDD in Beziehung gebracht, da das Entwicklungsvorgehen nach Scrum Teil des Kontextes der untersuchten Fallstudie ist.

3.4.2 Definition

„*Scrum ist ein agiles Managementframework zur Entwicklung von Software*" (Pichler, 2013, S. 1). Diese eingängige Definition grenzt Scrum auf die Softwareentwicklung ein und berücksichtigt nicht, dass die Urheber Ken Schwaber und Jeff Sutherland das Rahmenwerk Anfang 1980er-Jahre allgemein für die „Entwicklung und Erhaltung komplexer Produkte" (Schwaber & Sutherland, 2013, S. 3) entwickelt haben. Da sich Scrum aber gut in der agilen Softwareentwicklung einsetzen lässt, ist es in der Praxis sehr erfolgreich und weit verbreitet (vgl. Wirdemann & Mainusch, 2017, S. 1).

Das Scrum Framework als Regelwerk für den Einsatz von Scrum verkörpert die Prinzipien des Agilen Manifests[2] (vgl. Beck et al., 2001). Dies spiegelt sich in der Definition der beiden Scrum-Urheber Schwaber und Sutherland (2013) wider (S. 3):

> *„Scrum (n): Ein Rahmenwerk, innerhalb dessen Menschen komplexe adaptive Aufgabenstellungen angehen können, und durch das sie in die Lage versetzt werden, produktiv und kreativ Produkte mit dem höchstmöglichen Wert auszuliefern."*

Im Mittelpunkt der Softwareentwicklung steht der Mensch, da nur durch die Kooperation und Interaktion von Menschen Software entstehen kann. Scrum ist nicht technologieorientiert, sondert fördert die Zusammenarbeit der Beteiligten. Außerdem formuliert das Agile Manifest die Optimierung von Kundenzufriedenheit und Wertschöpfung als Ziel der Softwareentwicklung (vgl. Pichler, 2013, S. 1).

Scrum definiert eine geringe Anzahl an Rollen, Artefakten und Regeln für die Durchführung von Entwicklungsprojekten als Framework. Je nach Projekttyp kann das Vorgehen an die vorliegenden Projektgegebenheiten angepasst werden.

3.4.3 Die Rollen im Scrum-Team

Scrum kennt drei Rollen: Product Owner, Entwicklungsteam und Scrum Master. Gemeinsam bilden sie das Scrum-Team. Beschrieben werden sie von Bless (2016) wie folgt:

Product Owner

> *„Der Product Owner definiert, WAS gebaut werden muss. Er sortiert das Product Backlog nach Priorität, kennt die Anwender des Systems und kommuniziert regelmäßig mit den Stakeholdern."*

1 Eine vollständige Definition des Scrum Frameworks ist im Scrum Guide unter https://www.scrumguides.org einzusehen.

2 »Im Februar 2001 haben 17 Experten aus dem Bereich der Softwareentwicklung am Rande einer Konferenz ein Dokument erstellt, das Werte und Prinzipien für eine moderne Form der Softwareentwicklung beschreibt: Das agile Manifest« Preußig (2018), S. 38.

Entwicklungsteam

> *„Das Entwicklungsteam setzt sich aus Mitarbeitern aller Disziplinen zusammen, die notwendig sind, um das gewünschte Produkt fertig umzusetzen.“*

Scrum Master

> *„Der Scrum Master sorgt dafür, dass das agile Rahmenwerk richtig zur Anwendung kommt und hilft dem Team dabei, so effizient wie möglich arbeiten zu können.“*

3.4.4 Artefakte

Verschiedene Artefakte unterstützen die Projektdurchführung und helfen damit dem Scrum-Team, das Projektziel zu erreichen. Die verschiedenen Artefakte können folgendermaßen definiert werden (vgl. Rupp, 2014, S. 58 ff):

Produktvision

Die Produktvision ist eine abstrakte Art von Inhalten, die in dem zu entwickelnden System realisiert werden sollen. Alle im Folgenden angeführten Artefakte sollen sich in ihr wiederfinden lassen. Der Product Owner ist für die Erstellung und Kommunikation der Produktvision verantwortlich. Während des Sprints prüft er die Ergebnisse der Entwicklung und stellt so fest, ob das Team seine Vision erfüllt hat (vgl. Gloger, 2016, S. 77). Eine ausführliche Beschreibung der Produktvision findet sich in Kapitel 3.3.3.5.

Product-Backlog

Das Product-Backlog stellt eine geordnete Sammlung aller für das zu entwickelnde System relevanten Anforderungen dar. Für die Erstellung, die Pflege und die Ordnung der Backlog Items ist der Product Owner verantwortlich.

User-Storys

User-Storys sind eine mögliche Form von Backlog-Items. Sie beschreiben die Anforderungen aus Sicht der Anwender. Die Akzeptanzkriterien präzisieren die Anforderungen.

Epics

Epics stellen grobgranulare Anforderungen dar, deren Umfang nicht abgeschätzt werden können. Um sie schätzbar zu machen, müssen sie weiter geschnitten, also untergliedert werden.

Sprint-Backlog

Das Sprint-Backlog enthält alle User-Storys, die das Entwicklungsteam während der aktuellen Iteration realisieren möchte.

3.4.5 Die Durchführung

Ein Scrum-Projekt besteht aus einer oder mehreren Iterationen. Sie werden im Scrum-Kontext Sprint genannt. Sprints besitzen eine definierte Dauer zwischen möglichst einer und vier Wochen.

Zu Beginn jeden Sprints erfolgt das Sprint-Planning. In diesem Meeting stellt der Product Owner dem Entwicklungsteam die Storys vor, die für den Sprint angedacht sind, und beantwortet aufkommende Fragen. Das Ziel des Meetings ist, dass das Entwicklungsteam die Storys versteht und diese im Schätzmeeting bezüglich ihres Aufwands eingeordnet werden können.

Zur Kontrolle des Sprint-Ziels während der Iteration und zur Anpassung der Arbeitsweise finden tägliche „Daily Scrum Meetings“ statt. Innerhalb von 15 Minuten beantwortet jedes Teammitglied folgende Fragen:

1. Was habe ich gestern getan?
2. Was werde ich heute tun?
3. Sehe ich Hindernisse, die mich oder das Team vom Erreichen des Sprint-Ziels abhalten?

Am Ende jeden Sprints erfolgt das Sprint-Review, in dem Stakeholder und Scrum-Team das entwickelte Inkrement inspizieren. Die Resultate sind ein abgenommenes oder nicht abgenommenes Inkrement. Zwischen dem Sprint-Review und der nächsten Iteration erfolgt die Sprint-Retrospektive zur Reflexion der bisherigen Arbeitsweise. Die erkannten positiven und negativen Aspekte fließen in Verbesserungsvorschläge für die nachfolgenden Sprints ein.

3.4.6 RE und Scrum

Scrum sieht eine explizit definierte Rolle des Requirements Engineers nicht vor. Rupp (2014) argumentiert, dass sowohl der Product Owner (PO) als auch das Entwicklungsteam verschiedene RE-Disziplinen für ihren Aufgabenbereich benötigen – z. B. verschiedene Ermittlungs- und Konsolidierungstechniken, die benötigt werden, um den Backlog mit Backlog-Items zu befüllen. Auch die Kommunikation zwischen dem PO und dem Entwicklungsteam ist als RE-Aufgabe in Scrum zu sehen, denn das Entwicklungsteam muss im Sprint-Planning oder während der Entwicklung Informationen zu User-Stories erfragen und analysieren. Den analytischen Tätigkeiten wird dabei in vielen Projekten ein solches Gewicht beigemessen, dass ein eigenes Team dafür beauftragt wird.

Für die Integration eines RE-Teams gibt es verschiedene Möglichkeiten. Eine Möglichkeit ist nach Rupp (2014, S. 63):

- Wenn die Anforderungen vor einem Sprint bearbeitet werden sollen, kann dies durch ein eigenständiges RE-Team durchgeführt werden. Damit arbeiten RE- und Realisierungsteam zeitversetzt an demselben Inhalt und kommunizieren neue Erkenntnisse am das jeweilige Team. Das RE-Team bereitet die User-Storys für das Realisierungsteam vor und unterstützt damit den Product Owner bei seiner Arbeit.

Diese Möglichkeit der Integration kann durchaus kritisch betrachtet werden, denn übernimmt ein RE-Team die Analyse der Anforderung sowie die Kommunikation der Ergebnisse in Richtung Entwicklungsteam, entsteht ein erhöhter Abstimmungsaufwand. Stellen die Entwickler z. B. Fragen zu einer User-Story, die der Requirements Engineer nicht selbstständig beantworten kann, und muss er zur Beantwortung Informationen vom PO einholen, etabliert sich eine Dreiecks-Kommunikation, die zusätzliche Zeit in Anspruch nimmt. Die direkte Kommunikation zwischen PO und Entwicklungsteam ist hier von Vorteil.

Andererseits ist vorstellbar, dass nicht jeder PO die nötigen RE-Kompetenzen besitzt, um die Anforderungen zu analysieren und dem Entwicklungsteam verständlich zu machen. Hier wäre die Unterstützung eines RE-Teams vorteilhaft.

3.4.7 DDD und Scrum

Da DDD weder Methode noch Technik noch Framework ist, darf DDD nicht als Konkurrenz zu Scrum gesehen werden. Vielmehr ist DDD eine Herangehensweise, die Scrum im Entwicklungsvorgehen ergänzen kann.

Diese These stützt sich auf eine Aussage von Evans (2003), in der er betont, dass DDD nicht an eine bestimmte Entwicklungsmethode gebunden ist, sich aber an der Familie der agilen Entwicklungsprozesse orientiert (S. xxii). Danach sind folgende Bedingungen als Voraussetzung für die Anwendung von DDD zu erfüllen (Evans, 2003, S. xxii):

1. Die Entwicklung ist iterativ. Eine iterative Entwicklung ist ein essenzieller Bestandteil von agiler Softwareentwicklung, der über Jahrzehnte praktiziert und befürwortet wird.
2. Die Entwickler und die Domänenexperten haben eine enge Beziehung.

Auch Millet und Tune (2015) setzen eine iterative Entwicklungsmethode für die Anwendung von DDD voraus (S. 132). In diesem Zusammenhang ist zu klären, wie sich die Rolle des Domänenexperten in ein Vorgehen mit Scrum integrieren lässt.

Nach der Scrum-Philosophie ist der Product Owner alleine für die Analyse der Anforderungen und deren Kommunikation in Richtung des Entwicklungsteams zuständig. Es ist vorstellbar, dass der Domänenexperte an den Meetings des Scrum-Teams teilnimmt. So könnte z. B. das Sprint-Planning dazu dienen die vom PO vorgestellten Storys am Domänen-Modell zu diskutieren und so ein gemeinsames Verständnis zu schaffen. Möglich wäre auch, dass der PO die Rolle des Domänenexperten einnimmt, wenn dieser über das notwendige Domänen-Wissen verfügt.

3.5 RE in DDD

3.5.1 Überblick

Nach der theoretischen Auseinandersetzung mit den Themen RE und DDD werden in diesem Kapitel beide Themen in Beziehung zueinander gestellt und vor dem Hintergrund der Forschungsfrage analysiert.

3.5.2 Grundlagen

Unabhängig davon, ob mit DDD vorgegangen wird oder nicht, sollten die Grundlagen eines Entwicklungsprojekts vor Projektbeginn ermittelt worden sein. Dazu zählen die Definition der Projektziele, die Identifikation der Stakeholder und die Ermittlung und Abgrenzung von Scope und Kontext (vgl. Rupp, 2014, S. 47; s. auch Kap. 3.3.3).

Die Rolle des RE

Diese grundlegenden Tätigkeiten zählen zu den Aufgaben des Requirements Engineerings. Die Probleme, die entstehen können, wenn die Ziele unklar sind, wurden in Kapitel 3.3.3.2 dargestellt. In der Verantwortung des Requirements Engineerings könnte z. B. liegen, mit den Stakeholdern des Projekts einen Zielfindungs-Workshop durchzuführen. Die Grundlage könnten sowohl durch ein eigenständiges RE-Team als auch durch einen Projektleiter oder PO geschaffen werden. Da diese Aufgaben von der Herangehensweise DDD unabhängig durchgeführt werden, gibt es hier keine DDD spezifischen Anforderungen an ein RE.

3.5.3 Kommunikation zwischen Domänenexperten und Entwicklern

Im DDD verzichtet man bewusst auf die Funktion eines Intermediärs, der zwischen dem Entwicklungsteam und den Experten vermittelt. Durch die direkte Zusammenarbeit soll verhindert werden, dass die Expertise und das Wissen der Fachkundigen im Entwicklungsprozess verloren gehen oder bei der Weitergabe verfälscht werden. Ein Requirements Engineer als Bindeglied zwischen Fachbereich und Entwicklung ist im DDD nicht erwünscht.

Die Vermittlung zwischen Entwicklungsteam und Domänenexperten ist jedoch eine Aufgabe, die dem Requirements Engineer zugeschrieben wird. Begründet wird der Bedarf eines Vermittlers mit dem Gebrauch verschiedener Sprachen. Der Requirements Engineer übernimmt die Übersetzung der fachlichen Sprache der Domänenexperten in eine für die Entwickler verständliche, technische Sprache und umgekehrt.

Im DDD ist diese Aufgabe überflüssig, denn das Team verwendet ausschließlich eine fachliche Sprache zur Kommunikation. Die Begriffe der UL werden in einem Glossar festgehalten, sodass die Begriffsdefinitionen für das Team explizit und transparent sind. Auf die einheitliche Verwendung der Begriffe wird geachtet.

Die Rolle des RE

Die Verantwortung für die Erstellung und Pflege eines UL-Glossars sowie die Überprüfung der einheitlichen Verwendung der Begriffe in der Team-Kommunikation kann entweder an das gesamte Projektteam übertragen oder von Einzelpersonen übernommen werden.

Eine Person (oder ein Team) in der expliziten Rolle des Requirements Engineers eignet sich für diese Aufgabe gut, denn ist er weder Teil des Entwicklungsteams noch Mitarbeiter des Fachbereichs. Er kann eine neutrale Position einnehmen, weil er in der Kommunikation weder an eine rein technische noch an eine rein fachliche Sprache gewöhnt ist. In der Position eines Wächters überprüft er bei der Kommunikation zwischen Fachbereich und Entwicklungsteam, ob die Teammitglieder die Begriffe der UL korrekt verwenden und kann gegebenenfalls korrektiv intervenieren. Ergeben sich neue Begrifflichkeiten oder Änderungen bestehender Definitionen, übernimmt er die Anpassung des Glossars. Er ist dafür verantwortlich, das Glossar korrekt und aktuell zu halten.

Ein Vorteil bei der Übertragung dieser Aufgabe an das gesamte Team ist, dass jedes Team-Mitglied an der Erstellung beteiligt ist, sich verantwortlich fühlt und dadurch das Glossar jedem präsent ist und vermutlich eher vom gesamten Team genutzt wird.

3.5.4 Analyse-Tätigkeiten in DDD

Im DDD analysieren die Domänenexperten mit dem Entwicklungsteam neue Anforderungen gemeinsam. Die Aufgabe des Domänenexperten ist es dabei, dem Entwicklungsteam die Geschäftsprozesse und Geschäftsregeln der zu analysierenden Domäne verständlich zu machen. Das Domänenmodell ist dabei Diskussionsgrundlage und wird im Projektfortschritt erweitert und angepasst. Sowohl von den Domänenexperten als auch vom Entwicklungsteam sind dabei gute kommunikative und analytische Fähigkeiten gefordert.

Aber auch schon vor der Diskussion zwischen Entwicklungsteam und Domänenexperten sind Analyse-Tätigkeiten erforderlich, denn der Domänenexperte ist nicht zwingend auch Auftraggeber oder Projektleiter und somit eventuell nicht für die initiale Erhebung, Dokumentation und Kommunikation der Anforderungen verantwortlich. Je nach Projektorganisation kann es auch einen Projektleiter oder PO geben, der mit den Aufgaben der Analyse betraut ist.

Die Rolle des RE

Ein eigenes RE-Team ist unter folgenden Voraussetzungen nützlich:

- Die kommunikativen oder analytischen Fähigkeiten aufseiten der Domänenexperten oder des Entwicklungsteams sind nur schwach ausgeprägt.

Hier kann der Requirements Engineer tätig werden und das Team unterstützen, indem er bei Kommunikationsproblemen als Moderator tätig wird oder durch RE-Techniken, wie dem Use Case oder Szenarien, die Analyse unterstützt.

- Der PO oder der Projektleiter ist für die Erhebung, Dokumentation, Kommunikation und Abstimmung der Anforderungen zuständig. Methoden und Techniken aus dem RE zur Durchführung dieser Aufgaben sind ihm nicht bekannt, oder es fehlt ihm die Zeit, diese anzuwenden.

Bei diesen Aufgaben kann ein RE-Team den PO oder den Projektleiter unterstützen, in dem es bestimmte Aufgaben übernimmt oder bei deren Durchführung behilflich ist. So ist vorstellbar, dass ein RE-Team Anforderungsworkshops mit den Auftraggebern oder den Anwendern des Systems durchführt. Auch bei der Dokumentation der Anforderungen kann ein RE-Team nützlich sein, denn die Methoden und Techniken dazu sind ihm bekannt.

Sind sowohl analytische als auch kommunikative Fähigkeiten aufseiten der Domänenexperten und der Entwickler ausreichend ausgeprägt, ist es vorstellbar, dass die Analyse-Tätigkeiten gemeinsam vom Team aus Entwicklern und Domänenexperten ausgeführt werden.

Weitere mögliche Aufgaben und Verantwortlichen eines Requirements Engineerings in einem Entwicklungsvorgehen mit DDD werden im Rahmen der folgenden Fallstudie analysiert und evaluiert.

4 FALLSTUDIE

4.1 Hintergrund

4.1.1 Das alte Softwaresystem: GSSN

GSSN – die Abkürzung steht für „Global Standard Source for Network Master Data“ –, im Weiteren GSSN-Classic genannt, ist ein System, in dem weltweite Vertriebs- und Servicenetzdaten aller Marken der Daimler AG verwaltet werden.

Heute hat GSSN mehr als 50 direkte und mehrere indirekte Abnehmersysteme, die in regelmäßigen Abständen die Daten abziehen und für verschiedenste händler- und/oder kundenrelevante Prozesse verwenden. GSSN-Classic wird seit über zehn Jahren weiterentwickelt. Durch das hohe Alter der Software und bedingt durch von Märkten getriebenen Sonderlösungen stößt GSSN-Classic an die Grenze der ökonomischen Wartbarkeit.

Darüber hinaus wurden Sicherheitslücken im System festgestellt. Die sicherheitsrelevanten Anforderungen lassen sich durch eine bloße Anpassung des Systems nicht umsetzen.

Aus diesem Grund wurde entschieden, GSSN-Classic durch ein neues, flexibles und auf dem Stand der heutigen Technik basierendes System abzulösen. Das neue System hat den Namen GSSN+ und soll in 197 Ländern ausgerollt werden. Mit der Implementierung wurde die Firma Daimler TSS in Ulm beauftragt (vgl. Daimler AG, 2016, S. 16).

4.1.2 Das Team GSSN+

Das GSSN+-Team (kurz: Team oder DDD-Team) besteht aktuell aus 11 Entwicklern, einem Scrum Master, zwei Product Ownern und drei Requirements Engineers. Drei der Entwickler waren bereits an der Entwicklung des GSSN-Classic-Systems beteiligt.

In der Rolle der Product Owner sind in GSSN+ zwei Personen des Fachbereichs »Customer Management & Retail Network Development« der Daimler AG tätig. Der fachlich erfahrenere Product Owner ist seit zehn Jahren im GSSN-Umfeld in verschiedensten Rollen tätig. Neben seiner Rolle als Product Owner für GSSN+ ist er aktuell als Administrator für das System GSSN-Classic für die technischen Belange sowie als Ansprechpartner für die Märkte in Bezug auf Anforderungen ans das System zuständig. Er wird in seinen Aufgaben von seinem Fachbereichskollegen in der Rolle des Product Owners für GSSN+ unterstützt.

Das RE-Team setzt sich aus einem Teamleiter und zwei weiteren Personen zusammen, wobei eine dieser Personen die Autorin der vorliegenden Arbeit und Durchführende der Fallstudie ist. Der Teamleiter des RE-Teams ist aufgrund seiner früheren Tätigkeit als Product Owner für GSSN-Classic mit GSSN fachlich vertraut.

Die Entwicklung der Oberflächenkonzepte liegt in den Händen eines UX-Designers, der Teil des Entwicklungsteams ist. Er ist außerdem zuständig für Erstellung eines Prototyps, der für die Entwicklung sowie für die Anwendertests eingesetzt wird.

4.1.3 DDD in GSSN+

Der Vorschlag, GSSN+ mit DDD zu entwickeln, kam vom Entwicklungsteam und wurde den Product Ownern vor Entwicklungsbeginn vorgebracht. Drei der Entwickler hatten bereits vor GSSN+ in anderen Projekten mit DDD Erfahrungen gesammelt. Diese bezogen sich allerdings nur auf einzelne Konzepte des DDD und gingen nicht über ein Experimentstadium hinaus. Zwei der Entwickler hatten vor Projektstart einen Workshop zu DDD absolviert. Die Erfahrungen wurden anschließend in internen Schulungen an das Team weitergegeben, zudem wurde Literatur im Team herumgereicht.

Von den beiden Product Ownern ist keiner vor Projektbeginn mit dem Thema DDD in Berührung gekommen, doch wurde aufgrund der bisher positiven Erfahrungen der Entwickler beschlossen, den Vorschlag umzusetzen.

Zu Projektbeginn wurden verschiedene Event-Storming Meetings durchgeführt. So sollte sich unter den Teammitgliedern ein gemeinsames Verständnis zur Fachlichkeit bilden. Diese Meetings sind nach Aussagen von verschiedenen Teammitgliedern nicht erfolgreich gewesen: Es fehlten die Erfahrung in der Durchführung und eine angemessene Moderation des Meetings. So konnten keine zufriedenstellenden Ergebnisse erzielt werden. Nach einigen erfolglosen Wiederholungen wurden die Meetings eingestellt.

Während der ersten *Iteration* (Siehe Kap. 4.2.4 für mehr Informationen zum Planungsprozess) wurde das Domänenmodell vor der Implementierung der Anforderungen durch das RE-Team erstellt. Dieses Vorgehen war wenig befriedigend und wurde in der zweiten Iteration geändert. Die Gründe werden im Kapitel 4.2.6 näher beschrieben. Aktuell erstellt das Entwicklungsteam das Modell nach der technischen Umsetzung der Anforderungen (siehe Kap. 4.2.7 für mehr Informationen zum Entwicklungsprozess).

4.2 Aktuelle Situation

4.2.1 Ansatz und Konzept

Das Team hält sich in der Entwicklung an das Scrum-Rahmenwerk und verfolgt damit einen agilen Ansatz. Darüber hinaus wird angestrebt, die Prinzipien des DDD zu befolgen und dessen Konzepte im Entwicklungsprozess anzuwenden. Weitere Einblicke zur aktuellen Projekt Situation geben die folgenden Unterkapitel.

4.2.2 Stakeholder

Im GSSN+-Projektumfeld existieren viele Stakeholder, die die Anforderungen direkt oder indirekt beeinflussen. Die Daimler IT, der Daimler Fachbereich und die Abnehmersysteme werden hier als Beispiel vorgestellt.

- Daimler IT:
  Interessen: Technische Entscheidungen in der Systementwicklung müssen transparent und nachvollziehbar sein. Die technologischen Einschränkungen und Auflagen müssen eingehalten werden.
  Kommunikation: Ein Vertreter der Daimler IT nimmt regelmäßig an allen Meetings teil
- Daimler Fachbereich Customer Management & Retail Network Development:
  Interesse: GSSN-Classic soll durch das neue System GSSN+ sukzessive abgelöst werden. Die Sicherheitslücken sollen geschlossen und die Wartungsprobleme, die ein monolithisches System hervorbringt, behoben werden. GSSN+ soll ein flexibles System sein, sodass neue Anforderungen der Märkte zeitnah umgesetzt werden können. Die Systementwicklung sollte trotz agilen Vorgehens die Zeit- und Budgetplanung einhalten.
  Kommunikation: Der GSSN-Fachbereich ist durch die Product Owner vertreten.
- Abnehmersysteme:
  50 Abnehmersysteme fragen in regelmäßigen Abständen Daten über GSSN-Classic ab. Diese Systeme bauen ihre fachliche Logik und ihre Prozesse auf den abgerufenen Daten auf. Durch die Ablösung von GSSN-Classic dürfen sich die fachlichen Konzepte und ihre Datenstrukturen nicht verändern. Eine Systemdokumentation über Strukturen und Fachlichkeit des neuen Systems ist erforderlich.
  Kommunikation: Die Vertretungen der Märkte stehen mit dem PO in Kontakt. Über regelmäßige Anwender-Tests wird das Feedback zur aktuellen Entwicklung eingeholt.

4.2.3 Geografische Verteilung des Teams

Das Entwicklungsteam ist mit acht Entwicklern am Unternehmensstandort der TSS GmbH in Ulm und mit drei Entwicklern am Standort Berlin ansässig. Drei der acht Entwickler in Ulm sind seit Mai 2018 im GSSN+-Team und befinden sich aktuell (Juli 2018) in der Einarbeitungsphase.

Die Product Owner reisen an drei Tagen in der Woche von Stuttgart zum TSS Standort nach Ulm. Ihr Büro vor Ort liegt dem des Entwicklungsteams gegenüber, in dem auch der UX-Designer, als Mitarbeiter der TSS, seinen Arbeitsplatz hat.

Das RE-Team verbringt zwei Tage der Woche am Entwicklungsstandort in Ulm und teilt sich dort die Räumlichkeiten mit den beiden Product Ownern.

4.2.4 Planungsprozess

Wie bei einem Vorgehen nach Scrum vorgesehen, werden die Entwicklungszyklen im Projekt in Sprints geplant. In GSSN+ dauert ein Sprint drei Wochen. Die Planungseinheit ist eine User Story, ihre Komplexität wird in Story Points geschätzt.

Für eine längerfristige Planung ist die gesamte Entwicklung in Iterationen mit eigenem Iterationsziel unterteilt. Die Dauer einer Iteration beträgt sechs Monate. Für eine Iterationsplanung werden den Iterationen durch den PO über eine Web-basiertes Story Map Epics und Stories zugeordnet. Dabei wird er durch den Teamleiter RE beraten. Auf die Story Map hat das gesamte Team Zugriff.

4.2.5 Meeting-Struktur

Es werden nur solche Meetings beschrieben, in denen das Requirements Engineering als Rolle vertreten ist. Das Daily Stand-Up Meeting wird aus diesem Grund nicht vorgestellt.

- **Carving**
  - **Teilnehmer:** 2 Entwickler, 0 – 1 Scrum Master, 1 PO, 1 – 2 Requirements Engineer/s
  - **Inhalt:** Neue fachliche Konzepte werden dem Entwicklungsteam anhand des Oberflächenkonzepts vorgestellt. Der Requirements Engineer stellt dazu die bereits geschriebene User Story vor. Die Entwickler haben die Möglichkeit, Fragen zu stellen und technische Einwände zu äußern.
  - **Ziel:** Zu neuen Fachkonzepten soll ein gemeinsames Verständnis im Team geschaffen werden. Durch die Rückmeldung der Entwickler soll die Qualität der User Stories erhöht werden, damit in Refinement Meetings weniger Diskussionen aufkommen.
  - **Zeitpunkt:** dienstags
  - **Häufigkeit:** wöchentlich werden zwei Meetings parallel durchgeführt
  - **Zeitrahmen:** 1,5 Stunden

Das Carving Meeting wurde initial mit allen Entwicklern durchgeführt. Da dieses Vorgehen häufig lange Diskussionen zur Folge hatte, wurde im Juni 2018 entschieden, zwei Meetings parallel zu veranstalten, bei denen jeweils zwei Entwickler als Vertreter des Teams anwesend sind.

- **Refinement**
  - **Teilnehmer:** Entwickler, Scrum Master, POs, RE-Team
  - **Inhalt:** Die Stories, die im Carving besprochen wurden, werden nun dem gesamten Team vorgestellt, diskutiert und durch die Entwickler geschätzt. Wird zu lange diskutiert, wird die User Story auf das nächste Refinement Meeting geschoben.
  - **Ziel:** Schätzung der User Stories

  - **Zeitpunkt:** mittwochs
  - **Häufigkeit:** wöchentlich
  - **Zeitrahmen:** 1,5 Stunden
- **Sprint-Planning**
  - **Teilnehmer:** Entwickler, POs, RE-Team
  - **Inhalt:** Die User Stories für den nächsten Sprint werden eingeplant. Dabei werden die einzelnen User Stories vom Entwicklungsteam durchgesehen. Fragen können geklärt werden.
  - **Ziel:** Planung des kommenden Sprints
  - **Zeitpunkt:** am Ende des letzten Sprints
  - **Häufigkeit:** einmal pro Sprint
  - **Zeitrahmen:** 1 Stunde
- **Review**
  - **Teilnehmer:** Entwickler, Scrum Master, POs, RE-Team, Daimler IT, Fachbereichsleitung der Daimler AG
  - **Inhalt:** Die Anwendung der umgesetzten User Stories wird durch einen Entwickler vorgestellt. Die Teilnehmer stellen Fragen und geben ihre Rückmeldungen dazu.
  - **Ziel:** Demonstration der neuen Funktionalitäten und Einholen von Feedback vonseiten des Auftraggebers
  - **Zeitpunkt:** am Ende des Sprints
  - **Häufigkeit:** einmal pro Sprint
  - **Zeitrahmen:** 1 Stunde
- **Retrospektive**
  - **Teilnehmer:** Entwickler, Scrum Master, POs, RE-Teamleitung
  - **Inhalt:** Die Entwickler, die POs und die RE-Teamleitung geben Rückmeldung und äußern Erwähnenswertes zum vergangenen Sprint. Eine Liste mit positiven und negativen Punkten wird vom Scrum Master erstellt und anschließend im Team priorisiert. Die Punkte mit der höchsten Priorität werden im gegebenen Zeitrahmen besprochen.
  - **Ziel:** Besprechung des vergangenen und Verbesserung des kommenden Sprints
  - **Zeitpunkt:** am Ende des Sprints
  - **Häufigkeit:** einmal pro Sprint
  - **Zeitrahmen:** 2 – 4 Stunden

4.2.6 Die Context Map

Der Begriff „Context Map“ wird im GSSN+-Projekt synonym für „Domänenmodell“ verwendet. Dies ist allerdings fachlich nicht ganz korrekt, da die Context Map in der DDD-Literatur eine Sicht auf das Domänenmodell ist, in der die Kommunikation zwischen den verschiedenen Bounded Contexts dargestellt wird. „Die Context Map ist eine Abbildung, in der das Verhältnis und die Integration von Bounded Contexts dargestellt wird“ (Vernon et al., 2017, S. 7).

Das DDD-Thema „Context Mapping“ ist kein Bestandteil dieser Arbeit, denn es handelt sich dabei um einen rein technischen Aspekt der Implementierung. Die „Context Map“ ist allerdings zu einem Begriff der gemeinsamen Sprache im Projektteam geworden.

Die Context Map im GSSN+-Projektumfeld ist im Confluence Tool modelliert und für jedes Teammitglied einsehbar. Beschrieben werden darin die verschiedenen Bounded Contexts mit den dazugehörigen Entwurfsmustern sowie alle benötigten Objektattribute, Feldlängen und Geschäftsregeln. Zu Beginn der Forschungsarbeit, September 2017, wurde die Context Map durch das RE-Team entworfen und seither weiterentwickelt. Jede neue Anforderung wurde noch vor der Umsetzung in die Context Map eingearbeitet.

Dabei wurden DDD-Entwurfsmuster wie Aggregat, Entity oder Value Objekt diskutiert und modelliert. Zusätzlich wurden die benötigten Attribute, Kommandos und Events beschrieben, die nach der Einschätzung des RE-Teams benötigt werden, um die fachliche Anforderung umsetzen zu können. Dieses Vorgehen führte zu mehreren Diskussionen zwischen RE- und Entwicklungsteam (siehe Anhang 0): Das Entwicklungsteam äußerte sich zu diesem Vorgehen kritisch, da die Entscheidung des Entwicklungsteams, welches Entwurfsmuster für die Umsetzung eines Fachobjekts genutzt werden sollte, aufgrund fehlender Erfahrung häufig erst während der Umsetzung getroffen werden konnte. Das Team entschied im Januar 2018, dass in der Context Map die Entwurfsmuster nicht mehr abgebildet werden sollten. Schon im Februar kam es erneut zur Diskussion; in Absprache zwischen RE- und Entwicklungsteam wird die Context Map von diesem Zeitpunkt an durch das Entwicklungsteam modelliert (s. Anhnag- Ergebnisse der teilnehmdenen Beobachtung). Der Grund dafür waren häufige Abweichungen zwischen der Context Map und dem Modell der Implementierung: Die Modellierungen des RE-Teams konnten häufig nicht in die technische Umsetzung übernommen werden. So ließ das RE-Team z. B. technische Aspekte außer Acht oder traf falsche konzeptionelle Entscheidungen.

4.2.7 Entwicklungsprozess

i. Die Anforderungen ergeben sich über die Grundfunktionalitäten des Bestands-Systems GSSN-Classic, entstehen durch das Feedback der Anwender bei regelmäßig organisierten Anwender-Tests oder werden durch den PO über direkte Kommunikation mit den Vertretungen der Märkte ermittelt.

ii. Anforderungen, die auf der Benutzeroberfläche umgesetzt werden müssen, werden in erster Instanz zwischen PO und UX-Team (im Weiteren mit UX abgekürzt) besprochen und konzipiert. Anforderungen, welche die Benutzeroberfläche nicht betreffen, werden direkt an das Requirements Engineering kommuniziert. Dabei werden die groben Anforderungen durch den PO erklärt.

iii. Sobald das Konzept der Benutzeroberfläche durch das UX-Team im Prototyp erstellt ist, wird es an das RE-Team übergeben. Dabei werden Details zur Fachlichkeit und Funktionsweise zwischen UX, RE-Team und PO geklärt.

iv. Auf Grundlage des Oberflächenkonzepts erstellt das RE-Team User Storys. Ist keine Oberfläche für die Umsetzung erforderlich, werden gegebenenfalls weitere Informationen über den PO eingeholt.

v. Im Carving Meeting werden die User Storys und das zugrundeliegende fachliche Konzept mit den Entwicklern besprochen und gegebenenfalls angepasst.

vi. Im Refinement werden die User Storys nochmals diskutiert und geschätzt. Unklare Storys werden erneut im Carving Meeting besprochen.

vii. Die geschätzten User Storys erweitern das Backlog.

viii. Die geschätzten User Storys fließen in die Sprint-Planung mit ein.

ix. Am Ende des Sprints passen die Entwickler die Context Map anhand der umgesetzten User Stories entsprechend an.

x. Am Ende des Sprints werden im Review die neuen Funktionalitäten vom Entwicklungsteam vorgestellt.

xi. Das RE-Team testet die umgesetzten Funktionalitäten über die Oberfläche oder über das *Backend* (Teil des Systems, der sich mit der Datenverarbeitung im Hintergrund beschäftigt).

4.2.8 Glossar

Es existiert ein Glossar mit den wichtigsten Begriffen im GSSN+-Umfeld. Das Glossar wird durch PO T. verwaltet und ist für den Projektbeteiligten einsehbar. Die Begriffe werden sowohl auf Deutsch als auch auf Englisch beschrieben.

5 ERGEBNISSE DER FALLSTUDIE

5.1 Darstellung und Diskussion der Interview-Ergebnisse

Im Folgenden werden die Ergebnisse der durchgeführten Befragung zur Rolle des Requirements Engineerings im Domain-Driven Design anhand der festgelegten Kategorien vorgestellt. Der Hintergrund zur Frage wird für jede Kategorie erklärt, um zu verdeutlichen, in welchem Zusammenhang die Frage mit der Forschungsfrage steht. Die Ergebnisse werden zuerst ohne Interpretation und Diskussion dargestellt – obwohl die Extraktion von Informationen aus den Interviewtexten und ihre Zuordnung zu einer bestimmten Kategorie und Dimension natürlich bereits mit Interpretationen der durchführenden Person verbunden sind (vgl. Gläser & Laudel, 2010, S. 218). Die Diskussion, in der die Ergebnisse analysiert und interpretiert werden, erfolgt in einem eigenen anschließenden Unterkapitel, um eine differenzierte Betrachtung zu ermöglichen. In den Diskussionen werden zusätzlich Methoden und Techniken zur Lösung aufgedeckter Probleme diskutiert. Die Analyse und Interpretation der Ergebnisse bildet die Grundlage für die Handlungsempfehlungen für das RE, die in Kapitel 7 vorgestellt werden.

5.1.1 Kategorie: Ziele

5.1.1.1 Hintergrund zur Frage

Sind dir die Projektziele und die Iterationsziele des Projekts GSSN+ stets klar?

Die Definition der Ziele gehört zu den Grundlagen eines jeden Projekts und zu den Aufgaben des Requirements Engineerings (siehe Kap. 3.3.3.2). Auch bei einem Vorgehen mit DDD sollten die lang- und kurzfristigen Ziele den Projektbeteiligten bekannt sein.

5.1.1.2 Ergebnisse

Die Ablösung von GSSN-Classic nannten vier der befragten Personen als Projektziel (Interview-Anhang A, B, D, E). Eine dieser Personen gab zusätzlich an, dass das Ziel „die Übersetzung der Vision in die Domänen“ ist (Interview-Anhang 0).

Fünf Personen sagten aus, dass die Iterationsziele nicht klar seien (Interview-Anhang A, C, D, E, F). Außerdem wurde das Fehlen des Gesamtüberblicks, des roten Fadens und der Richtung der Entwicklung genannt (Interview-Anhang C, E, F). Als Ursachen wurden die fehlende Kommunikation der Ziele, die häufige Änderung der Ziele und deren schwammige Definition angegeben.

Diese Punkte nannten die befragten Personen als Folge unklarer Zieldefinitionen:

- Es wird oft diskutiert, ob ein Thema Iteration 1 oder 2 relevant ist (Interview-Anhang E).
- Es gibt Abstimmungsbedarf (Interview-Anhang B).
- Die Entwickler haben sich ein eigenes Ziel definiert (Interview-Anhang F).

Der befragte Product Owner S. definiert das Ziel von GSSN+ wie folgt:

> *„Ein geiles System zu entwickeln, das Spaß macht. Der Nutzer soll dabei die Möglichkeit haben, die Fülle an Daten für sich zu nutzen. Es soll so aufgebaut sein, dass jeder unbewusst gern mit dem System arbeitet. Die Benutzer sollen Bock auf das System haben“* (Interview-Anhang PO).

5.1.1.3 Diskussion der Ergebnisse

Nach den Projekt- und Iterationszielen gefragt ergibt sich im Entwicklungs- und RE-Team ein eindeutiges Bild: Die „Ablösung von GSSN Classic“ als langfristiges Projektziel wird sowohl von den Mitgliedern des RE- als auch von denen des Entwicklungsteams genannt. Dabei deckt sich diese Formulierung nicht mit den Aussagen des Product Owners, dessen Ziel die Entwicklung eines Systems ist, das „Spaß macht“ und dem Benutzer die Möglichkeit bietet, die „Fülle an Daten für sich zu nutzen“ (Interview-Anhang PO).

Bereits auf den ersten Blick fällt auf, dass keines der genannten Ziele die SMART-Kriterien erfüllt (siehe Kap. 3.3.3.2). Das Ziel „Ablösung von GSSN Classic“ ist zwar spezifisch und realistisch, jedoch weder angemessen noch terminiert und messbar: Die Aspekte Attraktivität und Herausforderung fehlen in der Formulierung ebenso wie ein klarer Termin, bis wann das Ziel zu erreichen ist, sowie passende Metriken, um überprüfen zu können, ob das Ziel erreicht wurde.

Dem vom Product Ownern formulierten Ziel mangelt es zwar nicht an Attraktivität, jedoch ist die Formulierung unspezifisch, nicht messbar und nicht terminiert. Der unspezifische „Spaß“ bei der Arbeit ist kaum mess- und terminierbar.

Aus den Antworten kann geschlussfolgert werden, dass das längerfristige Projektziel nicht, oder nicht oft genug, im GSSN+ Team abgestimmt wurde.

In Bezug auf die Iterationsziele wird deutlich, dass diese unzureichend definiert und schlecht abgestimmt sind. Die Folgen werden genannt: Der Gesamtüberblick fehlt und die Richtung der Systementwicklung ist unklar. Dies wiederum führt dazu, dass es häufige Diskussionen darüber gibt, ob ein Thema relevant ist für die aktuelle Iteration, und dass sich die Entwickler ihr eigenes Ziel definiert haben – nämlich die „Ablösung von GSSN Classic“.

Die in den Interviews genannten Gründe für das Problem, also die schwammige Definition, häufige Änderungen und die fehlende Kommunikation der Ziele, stehen im Widerspruch zu den Anforderungen, denen eine Zieldefinition genügen muss (siehe Kap. 3.3.3.2). Da in der Formulierung des Ziels

„Ablösung von GSSN Classic“ Attraktivität und Herausforderung fehlen, besteht außerdem die Gefahr, dass die Teammitglieder unzureichend motiviert sind und das Ziel im schlimmsten Fall nicht erreicht wird.

Es ist offensichtlich, dass hier dringender Handlungsbedarf besteht. In Hinblick auf die Forschungsfrage ist zu konstatieren, dass ein Requirements Engineering in der untersuchten Projektsituation mit unklaren, unpräzise formulierten oder nicht abgestimmten Zielen von Nutzen ist, wenn es zur Beseitigung dieser Probleme beiträgt. Aufgabe eines Requirements Engineers ist in diesem Fall, Ziele, die nicht schon vor Projektbeginn definiert wurden, zu formulieren (siehe Kap. 3.3.3.2). Die Definition und Abstimmung von Zielen kann über einen Workshop nach der PAM-Methode (siehe Kap. 3.3.3.2) erfolgen. Nimmt das Entwicklungsteam am Workshop teil, ist dies ein wichtiger Schritt für ein gemeinsames Verständnis.

Die Aufgabe der Zielformulierung kann auch vom Product Owner übernommen werden (siehe abgeleitete Forschungsfrage **IV**, Kap. 1.3), denn In Scrum ist dieser für die Definition der Ziele verantwortlich. Hat er aber wenig Zeit oder fehlen ihm die methodischen Kompetenzen, kann der Requirements Engineer unterstützend tätig werden.

5.1.2 Kategorie: Vision

5.1.2.1 Hintergrund zur Frage

Wurde die Produkt-Vision besprochen?

Die Vision ist für die Motivation und als Orientierung für die Teammitglieder ausschlaggebend und die Kommunikation der Vision schafft im Projektteam ein gemeinsames Verständnis des Systems. Dieser Aspekt ist bei einem Vorgehen mit DDD von besonderer Bedeutung.

5.1.2.2 Ergebnisse

Alle Entwickler, die gefragt wurden, ob dem Team die Vision von GSSN+ vorgestellt wurde, antworteten mit „Ja“. F äußerte sich dazu wie folgt:

> *„Da hat der PO dann mal in einem Meeting die Vision vorgestellt. Das hatte geholfen, allerdings ändert sich das Ziel auch immer mal wieder. Und damals stand z. B. die Navigation im Vordergrund“* (Interview-Anhang F).

Person **C** hingegen sagte:

> *„Ich kenne zwar GSSN-Classic schon, aber habe keine Ahnung, welche Use Cases in GSSN+ umgesetzt werden sollen. Die Top-Five-Features fehlen. Meiner Meinung sollte sich die Vision in den Anwendungsszenarien wiederfinden, die dann wieder detaillierter in Use Cases spezifiziert werden“* (Interview-Anhang C).

Interviewerin: Was ist dir von der Vision hängen geblieben?

C:

> *„Hängen geblieben ist: Abbildung von Standort-Netzwerken, nicht nur für Händler …' "*

Nach der Besprechung der Ziele wurde Product Owner S. nach der Vision von GSSN+ gefragt. Folgendes gab er zur Antwort:

> *„Das habe ich eben ja schon gesagt. Das ist die Vision des Systems."*

Auf die Frage, ob die Vision an das Team kommuniziert wurde, antworteten der befragte PO:

> *„Ja, das haben wir gemacht, das wurde auch vom Entwicklungsteam auf Video aufgenommen"* (Interview-Anhang PO).

5.1.2.3 Diskussion

Aus den Antworten wird deutlich, dass die Abgrenzung zwischen Vision und Zielen im Projekt schwammig ist. **F** scheint in seiner Antwort die Begriffe „Ziel" und „Vision" gleichzusetzen. Auch Product Owner **S.** unterscheidet seine Zieldefinition nicht von der Produktvision.

F äußert außerdem, dass sich die Ziele „immer mal wieder" ändern. Es ist unklar, ob er die Projekt-Ziele oder doch die Vision damit meint.

Die Produktvision aufgrund von Änderungen der strategischen Ziele anzupassen, kann durchaus angemessen sein (siehe Kap. 3.3.3.5). Angesichts der kurzen Projektlaufzeit von 12 Monaten kann vermutet werden, dass „immer mal wieder" sich ändernde Visionen oder Ziele aber nicht auf eine Änderung der strategischen Projekt-Ausrichtung, sondern auf eine unsaubere Formulierung und fehlende Kommunikation zurückzuführen sind. Diese Vermutung wird durch die Antwort von **C** untermauert: „Es fehlen die Top-Five-Features und die Vision spiegelt sich nicht in den Anwendungsszenarien wider" (Interview-Anhang C).

Die Definition und Kommunikation der Vision sowie die damit verbundenen Probleme sind nicht DDD-spezifisch. Allerdings spielt auch in DDD die Vision eine wichtige Rolle, denn eine unklare oder schlecht kommunizierte Vision kann zu Unsicherheit und fehlender Motivation im Team führen. Die mögliche Folge davon ist, dass die Projekt-Ziele nicht erreicht werden (siehe Kap. 3.3.3.2).

Das *Domain Vision Statement* dient zwar dem gleichen Zweck wie die Produktvision – nämlich zur Richtungsweisung und Orientierung – beschreibt das zu entwickelnde System aber aus einem anderen Blickwinkel.

Während bei der Produktvision das zukünftige Produkt mit seinen wichtigsten Features sowie qualitativen Anforderungen (wie z. B. Skalierbarkeit und Performance) im Fokus steht, liegt der Fokus des *Domain Vision Statement* ausschließlich auf der Fachlichkeit. Wie in der theoretischen Betrachtung des *Domain Vision Statement* (siehe Kap. 3.2.8) dargestellt, verbindet es die Produktvision mit der Ausarbeitung der *Core Domain* (siehe Kap. 3.2.2). An dieser Stelle muss das Zitat von **B** aus den Ergebnissen zum Thema Ziele (Kap. 5.1.1.2) aufgegriffen werden. **B** sagt, dass das Projektziel „die Übersetzung der

Vision in die Domänen“ ist (Interview-Anhang B). Diese Zielformulierung widerspricht zwar in fast allen Punkten den SMART-Kriterien, macht aber den Zusammenhang zwischen Vision und Domänen deutlich. Um das *Domain Vision Statement* erstellen zu können, ist das Wissen über die *Core Domain* und die Abgrenzung zu ihren *Subdomains* notwendig. Das Problem, dass im GSSN+-Projekt keine Domänen definiert wurden, wird im Kapitel 5.2.2 behandelt.

Der Product Owner ist Eigner der Produktvision und daher für deren Erstellung und Kommunikation verantwortlich. Damit übernimmt er Aufgaben aus dem Requirements Engineerings – denn in Scrum ist keine Rolle für den Requirements Engineer vorgesehen (siehe Kap. 3.4.6).

Mangelt es dem Product Owner an den nötigen RE-Kompetenzen oder fehlt ihm die Zeit, die zur Erstellung und regelmäßigen Kommunikation der Produktvision nötig ist, ist es im Hinblick auf die Forschungsfrage vorstellbar, dass er von einem RE-Team unterstützt wird. Ein RE-Team könnte Workshops zur Erarbeitung der Produktvision mit den Stakeholdern organisieren und moderieren, denn der Requirements Engineer kennt die benötigten Ermittlungs- und Konsolidierungstechniken (siehe Kap. 3.3.4). Das Gleiche gilt für die *Domain Vision*. Auch hier könnte des Requirements Engineering den PO unterstützen und z. B. darauf achten, dass im *Domain Vision Statement* ausschließlich fachliche und lösungsneutrale Aspekte beschrieben werden.

5.1.3 Kategorie: DDD

5.1.3.1 Hintergrund zu den Fragen

- Wie viel Erfahrung hast du bereits praktisch und/oder theoretisch mit DDD sammeln können?
- Was ist für dich DDD? Wie würdest du in ein oder zwei Sätzen erklären, was DDD ist?
- Wo siehst du Vorteile in der Anwendung von DDD im aktuellen Projekt GSSN+?
- Welche Probleme gibt es aktuell in der Anwendung von DDD im aktuellen Projekt?

Die Frage nach der Erfahrung und der persönlichen Definition von DDD zielte darauf ab, in einem späteren Schritt herauszufinden, welche Probleme auf eine fehlende Erfahrung oder auf mangelndes Verständnis der Teammitglieder zurückzuführen sind. Die Frage nach den Vorteilen und den Problemen sollte Aufschluss darüber bringen, wie viele der Teammitglieder der Vorgehensweise aufgeschlossen gegenüberstehen und wo es aktuell Probleme in der Anwendung von DDD gibt.

In einem späteren Schritt wird auf Basis dieser Ergebnisse untersucht, ob die bestehenden Probleme eventuell einer falschen Anwendung der DDD-Prinzipien geschuldet sind und inwieweit eine Verschlossenheit gegenüber DDD zu Problemen führen kann.

5.1.3.2 Ergebnisse

Subkategorie: Erfahrung mit DDD

Die befragten Personen **A** und **B** des RE-Teams hatten vor dem Projektbeginn weder theoretische noch praktische Erfahrung mit DDD vorzuweisen. Theoretische Grundlagen haben sie sich nach Projektbeginn über Bücher und das Internet angeeignet (Interview-Anhang A, B). Die Personen **D**, **E** und **F** des Entwicklungsteams gaben bei der Befragung an, DDD in vergangenen Projekten „ausprobiert" und damit „rumexperimentiert" zu haben (Interview-Anhang D, E, F). **E** räumte ein, dass dabei viele Anfängerfehler gemacht worden sind. Dazu folgendes Zitat:

> *„Wir haben vor allem die Patterns angewendet und die Bounded Contexts nur stiefmütterlich betrachtet. Die Bounded Contexts gehören aber in den Vordergrund, und dann kommen die Pattern"* (Interview-Anhang E).

Das Wissen zu DDD haben sich die Teammitglieder durch das Lesen von Büchern, über Konferenzbesuche und durch die von Teammitgliedern durchgeführten internen Workshops, angeeignet.

Subkategorie: Definition DDD

Die Antworten der einzelnen Teammitglieder werden zitiert:

Person A:

> *„Man hat eine Context Map. Die Domänen werden unabhängig voneinander betrachtet, jeder Bereich hat seine Funktion in der Domäne. Die Konzeption erfolgt unabhängig zwischen den einzelnen Domänen"* (Interview-Anhang A).

Person B:

> *„DDD ist die Weiterentwicklung eines fachlichen Datenmodells mit Eigenständigkeit der Domänen"* (Interview-Anhang B).

Person C:

> *„In der ersten Instanz ist es eine Methode zur Analyse von fachlichen Problemstellungen für Transparenz und ein gemeinsames fachliches Verständnis. In GSSN+: Ich möchte Händler-Stammdaten verwalten"* (Interview-Anhang C).

Person D:

> *„Die Fachlichkeit steht im Vordergrund, es gibt Bounded Contexts. Entwickler und Domänenexperten sprechen die gleiche Sprache"* (Interview-Anhang D).

Person E:

> *„Ein Modellierungsansatz, bei dem die Fachlichkeit im Mittelpunkt steht"* (Interview-Anhang E).

Person F:

> *„Die Fachlichkeit steht im Vordergrund und diktiert alles. Es gibt keine technischen Begriffsdefinitionen. Eine fachliche Sprache wird verwendet"* (Interview-Anhang F).

Antworten der PO: *(Interview-Anhang PO)*

T:

> *„DDD bedeutet eigentlich nur, dass die Entwickler die Fachlichkeit besser verstehen. So bekommt der Developer die Fachlichkeit besser umgesetzt und es gibt weniger Rückfragen.“*

S:

> *„Es ist einfach nur ein weiterer Ansatz, das System zu entwickeln. In drei Jahren heißt es anders. Das ist wieder mal der Versuch, etwas Besonderes zu machen. Ich frage mich, warum macht man nicht einfach? DDD ist ein schöner Ansatz.“*

Abbildung 5 stellt die Häufigkeiten der in den Antworten genannten Begriffe, in einer sogenannten Tag-Cloud dar. Die Anzahl der genannten Begriffe bedingt die Wortgröße. Der Begriff „Fachlichkeit“ wurde mit vier Mal am häufigsten genannt, der Begriff „Verständnis“ nur einmal.

Abbildung 5: DDD-Wortwolke (Eigene Darstellung unter Verwendung von https://www.wortwolken.com)

Subkategorie: Vorteile durch DDD

Von den Teammitgliedern wurden diese Aspekte als Vorteile von DDD genannt:

- die Unterstützung der Anforderungsanalyse durch die gegebene Architektur mit vielen Services (Interview-Anhang A)
- die Dokumentationsfähigkeit der Geschäftsregeln (Interview-Anhang B)
- der fachlichere Ansatz für eine Microservice-Architektur (Interview-Anhang B)

- die gemeinsame Sprachebene (Interview-Anhang B)
- die Aufteilung in Bounded Contexts, die eine lose Kopplung gewährt (Interview-Anhang D)
- das Team ist im gleichen Problem Space unterwegs (Interview-Anhang E)
- Entwickler verstehen die Fachlichkeit besser (Interview-Anhang PO)
- es gibt weniger Rückfragen von den Entwicklern (Interview-Anhang PO)

Person **C** antwortete auf die Frage, welche Vorteile das DDD im aktuellen Entwicklungsprozess hat: „Es war zum Scheitern verurteilt" (Interview-Anhang C). Die Gründe für das Scheitern, die **C** anführt, werden im folgenden Kapitel beschrieben.

Subkategorie: Probleme des DDD

Nach den Problemen in der Anwendung von DDD im aktuellen Projekt gefragt, eröffnet sich der Interviewenden ein breites Problemfeld.

A nennt die fehlende fachliche Diskussion anhand der Context Map und fügt dem hinzu:

> *„Die Diskussion über die Context Map fehlt. Das, weiß ich, gehört eigentlich zu DDD. Es gibt Situationen, in denen die Bedeutung der Context Map für mich (als RE) nicht klar ist bzw. ich auch ohne auskommen könnte"* (Interview-Anhang A).

B äußert, dass ein agiles Projekt verlangt, dass sich der PO mit dem Thema DDD auseinandersetzt. Er ergänzt, dass sich der PO auf die Context Map einlassen muss.

> *„Die Akzeptanz beim Kunden muss vorhanden sein, dass das funktioniert"* (Interview-Anhang B).

C nennt die Problematik, dass nicht alle Kollegen die Methodik verstanden haben. Er sagt:

> *„Man bräuchte Leute (…), die wissen, wie sie technisch funktioniert in der Umsetzung, und die das auch wertschätzen. Das sollte nicht von extern motiviert sein. Alle zusammen in eine Schulung packen und dann entscheiden, ob man das möchte oder nicht. Und das nicht während der Projektlaufzeit, sondern schon im Vorprojekt."*

Als Folge von Unwissenheit nennt er fehlende Motivation bei einigen Kollegen im Entwicklungsteam. So führt er als Beispiel an, dass die Context Map nicht von allen Kollegen genutzt wird (Interview-Anhang C).

D beschreibt folgendes Problem:

> *„Auf allen Seiten sollte aber das Bewusstsein für eine gemeinsame Sprache da sein; das fehlt immer mal wieder. Wir Entwickler sollten aber nicht so dogmatisch sein und die Fachlichkeit vorgeben wollen bei der Begriffsdefinition"* (Interview-Anhang D).

F erklärt:

> *„Aus Entwicklersicht betreiben wir DDD. Auf Seite des Fachbereichs nicht. Ich weiß aber nicht, ob es falsch ist. Es funktioniert. Wir wissen nicht, ob es anders besser funktionieren würde"* (Interview-Anhang F).

Die Product Owner wurden nach den Problemen in der Entwicklung im Allgemeinen befragt. Folgende Kommunikation wurde dabei protokolliert (Interview-Anhang PO):

S:

> *„Das Rollenverständnis. Wenn du deine Rolle nicht akzeptierst, wird alles diskutiert. Das ist nicht effizient. Der Anspruch an die Software ist, dass der Anwender damit etwas anfangen kann. Bei uns spielt der aber oft keine Rolle."*

Frage: „Du meinst, bei den Entwicklern spielt das keine Rolle?"

S:

> *„Ja auch, aber auch auf anderen Ebenen. Das ist ein Großkonzernproblem. Da geht es nur um Budget und Termine. Das Produkt an sich ist dabei leider oft zweitrangig."*

T:

> *„Mir passt nicht, dass die TSS nicht offen für den Wettbewerb sein muss, weil sie eine Daimlertochter ist. Die haben ein ganz anderes Selbstverständnis. Mit anderen Unternehmen würde das anders laufen. Das Problem ist, dass die Leute immer mitreden und mitdiskutieren wollen. Und am Ende behaupten sie, dass die Erde flach sei."*

S:

> *„Die Entwickler maßen sich an, mehr wissen zu wollen als der Enduser. Jeder hat seinen Bereich und seine Aufgaben. Sie müssen nicht mit dem Enduser sprechen."*

5.1.3.3 Diskussion

Aus den Ergebnissen der Kategorie **Erfahrung DDD** wird deutlich, dass die Projektentwicklung mit sehr wenig DDD-Erfahrung innerhalb des gesamten GSSN+-Teams begonnen hatte. Das RE-Team sowie die Product Owner hatten keinerlei Erfahrung in der Anwendung von DDD. Die praktische Erfahrung im Entwicklungsteam bezog sich hauptsächlich auf das Experimentieren mit den unterschiedlichen „Patterns", also den Entwurfsmustern des DDDs.

Das Zitat von E, in dem er sagt, dass der Bounded Context in DDD in den Vordergrund gehört und dann erst die „Patterns" relevant sind, zeigt, dass sich das Team mit dem Problem einer unzureichenden Umsetzung von DDD (s. Kap. 3.2.9.4) bereits auseinandergesetzt hat oder entsprechende Erfahrung sammeln konnte.

In der Subkategorie **Definition DDD** wurden die Antworten aller Personen zur Frage, was sie unter DDD verstehen, zitiert. Über die Antworten kann auf ein mangelndes oder falsches Verständnis der DDD-Herangehensweise und innerhalb des Teams geschlossen werden.

A stellt die Context Map in das Zentrum von DDD. Dabei beschreibt er die Unabhängigkeit zwischen den einzelnen Domänen und spricht von Bereichen, die eine Funktion innerhalb der Domänen haben. Der Begriff „Domäne" wird hier unsauber verwendet. Denn in DDD werden nicht die Domänen unab-

hängig voneinander betrachtet, sondern die Domänenmodelle innerhalb eines Bounded Contexts. Dieser kann sich über mehrere Domänen erstrecken (Kap. 3.2.5.2) oder einer von mehreren Kontexten innerhalb einer Domäne sein. Die unsaubere Verwendung der Begriffe „Bounded Context“ und „Domäne“ verdeutlicht, dass die Bedeutung der Begriffe nicht klar ist.

Auch die Definition von **B** ist unsauber. Er weist auf die Eigenständigkeit der Domänen hin und meint damit die Eigenständigkeit der Bounded Contexts. Auch die Verwendung des Begriffs „Datenmodell“ greift zu kurz, denn der Fokus sollte auf der Domänen-Logik liegen und nicht auf den Daten des Modells (siehe Kap. 3.2.1).

Die unsaubere Verwendung der Begriffe lässt sich durch Aufnahme der DDD-spezifischen Begriffe in ein Glossar vermeiden. Denn wie in Kapitel 3.2.5.3 beschrieben sind die Fachbegriffe des DDD Teil der Ubiquitous Language.

Auch die Definitionen „Methode zu Analyse von Problemstellungen“ und „Modellierungsansatz“ werden den Absichten des DDD nicht gerecht. DDD ist mehr als nur eine „Art der Durchführung“ und wird aus diesem Grund in der Literatur auch als „Herangehensweise“ oder „Philosophie“ definiert. Denn es geht im DDD auch um Haltung und Werte; so wird z. B. auf eine enge Zusammenarbeit zwischen Entwicklern und Domänenexperten großer Wert gelegt.

Über die Antworten der PO wird deutlich, dass ihnen die Bedeutung und Vorteile des DDDs durch das Entwicklungs- und das RE-Team nicht vermittelt werden konnten. Dass das Ziel von DDD nicht die Reduzierung der Rückfragen durch die Entwickler ist, muss an dieser Stelle nicht weiter kommentiert werden. Auch die Aussage, dass DDD „nur“ bedeutet, dass die Entwickler die Fachlichkeit besser verstehen, greift zu kurz, denn die gemeinsame Erarbeitung der Fachlichkeit hilft auch den Domänenexperten, ihr Wissen zu präzisieren und auf Wissenslücken zu stoßen (siehe Kap. 3.2.4).

Abbildung 5*: DDD-Wortwolke* macht deutlich, dass DDD im Team hauptsächlich mit dem Begriff „Fachlichkeit“ assoziiert wird. Indem die Fachlichkeit in den Vordergrund gestellt wird, will man im DDD ein gemeinsames Verständnis für die Problemdomäne schaffen. Das gemeinsame Verständnis wird nur von einer Person genannt, sollte aber in den Vordergrund gerückt werden, denn nur so kann die enge Zusammenarbeit zwischen Domänenexperten und Entwicklern im Team motiviert werden.

In der Subkategorie **Vorteile DDD** werden von den Befragten einige Vorteile von DDD genannt. Das übergeordnete Ziel des DDD, die Komplexität von Problemen zu verringern, wird allerdings nicht erwähnt.

Die Probleme, die in der Subkategorie **Probleme mit DDD** genannt werden, beziehen sich zum größten Teil auf die Aspekte des DDD, die als eigene Kategorien Teil der Interviews sind. Daher wird auf die hier genannten Probleme in den entsprechenden Kapiteln der Kategorien gesondert eingegangen.

A spricht das Problem an, dass die Diskussion über das Domänenmodell fehlt. Das Problem wird in den Ergebnissen zu „Domänenmodell“ in Kapitel 5.1.7 behandelt.

Das von **D** angeführte Problem in Bezug auf eine gemeinsame Sprache wird in Kapitel 5.1.6.3 aufgegriffen.

In der Antwort von **F** wird anhand der Formulierung deutlich, dass DDD nicht als eine gemeinsame Herangehensweise wahrgenommen wird. F spricht von „Entwicklersicht“ und „Fachbereichsseite“ und davon, dass „nur“ aus Entwicklersicht DDD „betrieben“ wird.

Mit „DDD aus Entwicklersicht“ ist wahrscheinlich das taktische Design des DDDs gemeint, denn im strategischen Design ist immer der Domänenexperte, also die Fachbereichsseite, involviert. Aus der Aussage kann also geschlussfolgert werden, dass das strategische Design vernachlässigt wird.

Die Product Owner sehen ein Problem im Rollenverständnis der Entwickler, wenn diese mitreden und mitdiskutieren wollen. Sie empfinden es als Anmaßung, wenn die Entwickler mehr wissen wollen als die Benutzer oder mit diesen sprechen wollen.

Diese Aussage verdeutlicht ein Problem in der Anwendung von DDD im Projekt GSSN+: Eine fachliche Diskussion im Sinne von DDD ist nur möglich, wenn die Domänenexperten offen für fachliche Diskussionen sind. Wenn der Wissensdurst der Entwickler als Anmaßung empfunden wird, gestaltet sich dies schwierig. Auch der Wunsch des Entwicklungsteams, mit den Benutzern des geplanten Systems sprechen zu wollen, sollet als positives Zeichen dafür verstanden werden, dass die Entwickler ernsthaft an der Entwicklung eines wertvollen Produkts für die Benutzer interessiert sind. Wird dieser Wunsch durch die Product Ownern nicht erfüllt und auf ein falsches Rollenverständnis zurückgeführt, führt dies zur Demotivation im Team.

Die Aufgabe des Requirements Engineers muss hier sein, den Domänenexperten zu motivieren und ihm deutlich zu machen, dass auch das Entwicklungsteam die Probleme verstehen muss, weil dies die Voraussetzung einer wertvollen Umsetzung von DDD ist.

Hier zeigt sich das Problem, dass der PO als Projektverantwortlicher auch als Domänenexperte tätig ist. In Kapitel 5.1.5 wird auf dieses Problem näher eingegangen.

In Bezug auf die Forschungsfrage kann als Ergebnis der Befragung in der Subkategorie **Erfahrung DDD** Folgendes festgehalten werden: Das Requirements Engineering ist nicht für das DDD-Wissen im Team verantwortlich und kann auch die fehlende Erfahrung nicht kompensieren. Allerdings ist es erforderlich, dass sich der Requirements Engineer selbst gründlich mit DDD auseinandersetzt. Nur mit dem Wissen, dass in DDD die direkte Kommunikation zwischen Domänenexperten und Entwicklungsteam ein wichtiges Prinzip darstellt und es nicht erforderlich ist, fachliche Anforderungen in eine technische Sprache zu übersetzen, kann er seiner Rolle in DDD gerecht werden und wertvoll für die Projektentwicklung sein.

5.1.4 Kategorie: RE

5.1.4.1 Hintergrund zur Frage

- Welche Rolle spielt die Funktion RE im aktuellen Vorgehen für dich?
- Ist es deiner Meinung nach notwendig, dass im aktuellen Vorgehen ein RE-Team Teil des Vorgehens ist oder könnten die Aufgaben und Funktionen auch von anderen Rollen aus dem Team übernommen werden?
- Könnte das RE deiner Meinung nach weitere Aufgaben und Verantwortlichkeiten übernehmen, um das aktuelle Vorgehen zu unterstützen?

Da das Forschungsziel dieser Arbeit die Auseinandersetzung mit der Rolle des Requirements Engineerings im DDD ist, ist es unabdingbar, diese Frage auch aus dem GSSN+-Team heraus beantworten zu lassen. Die unterschiedlichen Wahrnehmungen und Standpunkte der einzelnen Teammitglieder sollen ermittelt werden. Da in der anschließenden Diskussion von Bedeutung ist, aus welcher Rolle heraus diese Antwort gegeben wurde, wird beim Zitieren der befragten Personen jeweils die entsprechende Rolle angegeben.

5.1.4.2 Ergebnisse

A (RE) sieht ein eigenständiges RE-Team als notwendig an. Er begründet seine Aussage damit, dass die POs nicht alle User Storys selbst schreiben könnten. Fehlendes technisches Verständnis auf PO-Seite würde durch das RE-Team kompensiert, indem es die fachlichen Anforderungen für das Entwicklungsteam übersetzt. Das RE-Team nehme dabei eine Schnittstellenposition ein. Durch die Abstimmung mit dem Entwicklungsteam und UX nehme das RE-Team den POs außerdem viel Arbeit ab.

Auf die Frage, welche Aufgaben und Verantwortlichkeiten das RE-Team zusätzlich übernehmen könnte, antwortet **A** (Interview-Anhang A):

> *„Den Prozess zur Lösungsfindung so zu unterstützen, damit nicht ständig umentschieden wird. Die Entscheidungen sind oft sprunghaft, da könnte das RE besser unterstützen."*

Auch **B (RE)** sieht die Verantwortung des RE-Teams in der Übersetzung der fachlichen Anforderungen für die Entwicklung. Ein technisches Verständnis ist für **B** die Voraussetzung für eine Diskussion mit dem Entwicklungsteam. Auf die Frage, ob die RE-Aufgaben auch von anderen am Projekt beteiligten Rollen übernommen werden könnten, antwortet er (Interview-Anhang B):

> *„Wir sind ja dem Fachbereich angegliedert, also verlängerter Arm des POs. Wir sind Anwalt des POs und nicht des Entwicklungsteams. Das ist natürlich was Anderes, als wenn das RE vom Entwicklungsteam geleitet werden würde. Das Schwierige daran ist aber, dass wir als weitere Partei in der Abstimmung dazwischenstehen. Der Abstimmungsaufwand steigt."*

Weitere mögliche Aufgaben sieht **B** in einer „aktiveren Rolle“ bei der Erstellung des Domänenmodells (Interview-Anhang B).

Auf die Frage, ob die Aufgaben des REs auch von anderen am Projekt beteiligten Rollen übernommen werden könnten, äußern sich die Mitglieder des Entwicklungsteams folgendermaßen:

D (Entwicklung):

> *„Für die Entwicklung wäre das zu viel. Es ist sinnvoll, dass es das RE-Team gibt. Es wäre schwierig, wenn wir die Storys auch noch machen würden. Es ist hilfreich, wenn das vom RE-Team gemacht wird. Auf der anderen Seite sind wir zu viele einzelne Teams. Da gibt es viel Abstimmungsbedarf“* (Interview-Anhang D).

E (Entwicklung):

> *„Irgendjemand muss die Storys schreiben. Wir sind agil, aber man braucht Leitplanken. Das Aufbrechen von Epic zu Stories muss von jemandem gemacht werden. Die Entwicklung könnte anfangen, Stories zu schreiben. Der PO hat keine Zeit und keine Lust. Auf der anderen Seite bekommen wir mit dem RE-Team die Informationen über einen Mittelsmann rein. Das führt zu Dreiecksdiskussionen“* (Interview-Anhang E).

F (Entwicklung):

> *„Aus Entwicklersicht ist kein RE nötig. Das könnte auch von Entwicklern übernommen werden. Je direkter die Kommunikation, desto besser. Warum über einen Mittelsmann kommunizieren, wenn die Kommunikation auch direkt laufen könnte? Aus PO-Sicht sehe ich aber einen Grund: Das RE-Team puffert die Themen zum PO hin“* (Interview-Anhang F).

Auf die Frage, welche weiteren Aufgaben und Verantwortlichkeiten das RE-Team in DDD übernehmen könnten, wurde Folgendes geantwortet:

> *„Das RE sollte Struktur reinbekommen, den Kunden stressen. Fragen, warum brauche ich das?‘, dem Kunden richtig auf den Sack gehen“* (Interview-Anhang C).
>
> *„Das RE könnte im Review mit einer Roadmap aufzeigen, was die aktuellen Themen sind und wo wir in der Entwicklung gerade stehen“* (Interview-Anhang E).
>
> *„Die Moderation des Event Storming Workshops könnte von RE übernommen werden“* (Interview-Anhang E).
>
> *„Da sehe ich die Rolle eher moderierend. Im Event Storming zum Beispiel“* (Interview-Anhang F).

5.1.4.3 Diskussion

A und **B** sehen die Verantwortung des RE-Teams in der Übersetzung der fachlichen Anforderungen für die Entwicklung. Das fehlende technische Verständnis auf PO-Seite soll damit kompensiert werden. Für **B** ist das technische Verständnis sogar eine Voraussetzung für die Diskussion mit dem Entwicklungsteam.

Dieses eigene Rollenverständnis des RE-Teams ist nicht mit DDD zu vereinbaren. In DDD ist die Übersetzung von fachlichen Anforderungen in eine technische Sprache nicht notwendig, denn das Team

spricht eine rein fachliche Sprache – die Ubiquitous Language. Das RE in der Rolle des Übersetzers ist in DDD nicht erforderlich. Ist es dennoch notwendig, dass es eine Übersetzung zwischen Fachbereich und Entwicklungsteam stattfindet, ist dies ein Zeichen dafür, dass eine gemeinsame Sprache nicht existiert.

Ein weiteres Problem nennt **B**, in dem er sagt, dass das RE-Team als weitere Partei in der Abstimmung zwischen Fachbereich und Entwicklungsteam steht und der Abstimmungsaufwand dadurch steigt.

Auch **D** spricht dieses Problem an. **E** nennt die Dreiecksdiskussion als Folge davon, dass die Informationen über den Mittelsmann „RE" „reinkommen". Auch **F** äußert sich kritisch, indem er fragt, warum die Information über einen Mittelsmann läuft, wenn sie auch direkt laufen könnte. Die Probleme, die dadurch entstehen können, wurden in Kapitel 3.5.3 beschrieben.

In Hinblick auf die Forschungsfrage ist zu konstatieren, dass die durch das RE-Team unterbrochene direkte Kommunikation zwischen Entwicklungsteam und Fachbereich nicht mit DDD zu vereinbaren ist.

Im Gegensatz dazu wird das RE-Team aber von fast allen Befragten als notwendig erachtet. Von den meisten Befragten wird dabei die Entlastung des POs durch das Schreiben der User Storys vom RE-Team angeführt. Auf die Frage, ob das RE auch von anderen Rollen im Team übernommen werden könnte, antwortet **F** als einziger, dass die Aufgaben besser durch das Entwicklungsteam durchgeführt werden sollten.

5.1.5 Kategorie: Domänenexperte

5.1.5.1 Hintergrund zur Frage

Wer übernimmt deiner Meinung nach im aktuellen Projekt die Rolle des Domänenexperten?

Der Domänenexperte spielt im DDD eine zentrale Rolle. Da dessen Einbindung zu den häufigen Problemen von DDD gehört (siehe Kap.3.2.9.2), ist dieser Aspekt Teil der Untersuchung.

5.1.5.2 Ergebnisse

Beide Befragten des RE-Teams sehen PO S. in der Rolle des Domänenexperten, sagen aber auch, dass das RE-Team diese Rolle einnimmt.

A:

> *„(…) wir sind dann die Experten auf einem kleinen Gebiet"* (Interview-Anhang A).

B:

> *„Der PO ist der Domänenexperte. Das RE-Team nimmt diese Rolle aber auch ein, weil der PO auf dieser Ebene nicht diskutieren möchte. Das Abstraktionsvermögen fehlt da"* (Interview-Anhang B).

Zu obiger Interviewfrage gibt es auch Stimmen aus dem Entwicklungsteam:

C:

> *„Eigentlich haben wir keinen. Wir haben Lösungsexperten. Wer müsste das sein? Leute aus dem MPC (Market Performance Center) (…). Die Anwender sind mit dem Business vertraut. Sie kennen sich mit Rechtsstrukturen oder Rechtsformen aus. Sie wissen, wie die Betriebe aufgebaut sind, Beispiel: Haupt- und Nebensitz. Ist der PO der Experte dafür?“* (Interview-Anhang C).

D:

> *„Eigentlich S. Der RE-Teamleiter vielleicht auch. Wir als Software Engineers nicht“* (Interview-Anhang D).

E:

> *„S. ist der Domänenexperte, aber das passt nicht ganz genau. Wir müssten eigentlich mit dem Händler reden. Der weiß, wie er die Verträge verwaltet. Die Schwierigkeit dabei wäre, dass wir dann mit 50 Händlern sprechen müssten. Jetzt wird es über S. zentralisiert. Da ist er schon der richtige Ansprechpartner. Eigentlich ist der Domänenexperte jemand, der täglich mit dem Geschäft zu tun hat. Bei uns geht es aber nicht anders“* (Interview-Anhang E).

F:

> *„Das Problem ist, dass wir nicht an den richtigen Domänenexperten rankommen. Wir haben aber niemanden, der geeigneter wäre, als den PO, also passt das schon“* (Interview-Anhang F).

5.1.5.3 Diskussion

Die Identifikation der Stakeholder gehört zu den grundlegenden Aufgaben des Requirements Engineerings (siehe Kap. 3.3.3.3). Zu den Stakeholdern wird auch der Domänenexperte gezählt, denn durch sein Mitwirken beeinflusst er die Anforderungen direkt.

A und **B** aus dem RE-Team antworten beide, dass PO S. der Domänenexperte ist, diese Rolle aber auch durch das RE-Team eingenommen wird. Diese Aussage kann angezweifelt werden, denn der Domänenexperte ist die Person, die am meisten über einen bestimmten Bereich des Unternehmens weiß (siehe Kap. 3.2.5.4). Zwar ist dies unabhängig von der Rolle der Person – es kann sich also dabei um einen PO oder einen Requirements Engineer handeln –, jedoch kann aufgrund eines begrenzten Domänenwissens der RE-Teammitglieder gesagt werden, dass das RE-Team für diese Rolle nicht gänzlich geeignet ist. (Die Probleme, die entstehen können, wenn der PO in der Rolle des Domänenexperten tätig ist, werden in Kapitel 5.2.4 aufgezeigt; zu den Problemen, wenn anstelle eines Domänenexperten ein Business Analyst oder ein Requirements Engineer eingesetzt wird, siehe Kap. 3.2.9.3)

Die Entwickler machen in ihren Antworten deutlich, dass PO S. nicht die geeignete Person für die Rolle des Domänenexperten ist. Als potenzielle Domänenexperten werden die Händler und die MPCs genannt, die in GSSN+ durch PO S. vertreten werden. Auf den Wunsch des Entwicklungsteams zu Projektbeginn, einen direkteren Kontakt zu den Benutzern des Systems herzustellen, wurde nicht eingegangen, denn der Fachbereich sieht die Notwendigkeit dafür nicht (s. Anhang 0).

Eine Aufgabe des Requirements Engineerings in DDD kann im Sinne der Forschungsfrage die Identifikation eines geeigneten Domänenexperten sein. Mit diesem nimmt der Requirements Engineer Kontakt auf, um ihn für seine Mitarbeit am Projekt zu motivieren. Auf die Problematik, einen geeigneten und motivierten Domänenexperten zu finden, wurde bereits in Kapitel 3.2.9.2 eingegangen. Sind für die Entwicklung eines Projekts verschiedene Experten notwendig, kann es sinnvoll sein, einen „Domänenexperten-Ausschuss" zu gründen. Dieser wird zu „Workshops" eingeladen, in denen die Entwickler gemeinsam mit den Experten fachliche Diskussionen führen. Die Workshops können z. B. als Event Storming oder User Story Mapping (s. 3.3.4.3 „Workshop-Methoden") durchgeführt werden. Der Requirements Engineer nimmt dabei die Rolle des Moderators und Initiators ein.

Durch den offiziellen Charakter eines „Ausschusses" könnte die Motivation der Domänenexperten gesteigert werden, ihr Wissen mit dem Team zu teilen. Denn wer möchte nicht ein Teil eines ausgewählten Experten-Kreises sein?

5.1.6 Kategorie: Gemeinsame Sprache

5.1.6.1 Hintergrund zur Frage

Wird deiner Meinung nach im Projekt Wert gelegt auf eine gemeinsame Sprache?

Die gemeinsame Sprache ist im DDD ein wichtiges Instrument, um ein gemeinsames Verständnis im Team zu schaffen. Aus diesem Grund wurde sie in den Interviews thematisiert.

5.1.6.2 Ergebnisse

Alle befragten Personen sehen Probleme und Schwierigkeiten bei der Entwicklung und der Verwendung einer gemeinsamen Sprache im GSSN+-Team:

C:

> *„(…) man tut sich schwer, die Fachlichkeit auf der Sprachebene zu vermitteln. Beispiel „Business Site" und „Outlet". Der Kunde spricht nur von Outlet. Outet ist aber nur eine View"* (Interview-Anhang C).

B:

> *„Wir vermischen manchmal die Begriffe aus Classic und GSSN+."*

Wie oft schaust du in unser Glossar?

B:

> *„Nicht oft genug"* (Interview-Anhang A).

D:

> *„Die gemeinsame Sprache könnte besser sein. Beiden Seiten (Drei Seiten) sollte es wichtig sein, dass man vom Gleichen spricht. Wir Entwickler sollten aber nicht so dogmatisch sein und die Fachlichkeit vorgeben wollen bei der Begriffsdefinition"* (Interview-Anhang D).

E:

> *„Die gemeinsame Sprache ist nicht so optimal. Das sieht man an der Frage, was ein Outlet ist. Ich glaube nicht, dass die aktuelle Definition die richtige ist. Ein Outlet wird geöffnet und geschlossen. Wir sagen aber in GSSN+, wir erstellen und löschen ein Outlet. Wir benutzen also Begriffe, die ein Händler nicht benutzen würde. Das wäre aber eine natürliche Sprache. Aber wir haben uns davon verabschiedet und die Diskussionen darüber gelassen. Das bekommen wir nicht mehr weg. Jetzt haben wir einen Button „Create“ und „Delete“. Die Diskussionen darüber waren zu lang“* (Interview-Anhang E).

F:

> *„Am Anfang waren wir da aber auch zu akademisch mit der Definition. Wir dachten, dass es heißen sollte, „Outlet gründen“ anstatt „Outlet erstellen“, weil das der natürlichen Sprache entspricht, aber dass der Anwender des Systems wahrscheinlich auch „Outlet erstellen“ sagt und nicht „Outlet gründen“, haben wir in dieser Diskussion nicht berücksichtigt. Der MPC legt es eben auch nur an. Es gibt aber auch auf Fachbereich-Seite keine Offenheit, die Begriffe zu hinterfragen“.*

Schaust du manchmal ins Glossar?

F:

> *„Nein, am Anfang habe ich das mal“* (Interview-Anhang F).

5.1.6.3 Diskussion

D spricht das Problem an, dass die Entwickler bei der Begriffsdefinition zu dogmatisch sind und diese vorgeben wollen. Dies bezieht sich auf die Thematik der Antworten von **E** und **F.** Dabei geht es um die Frage, ob ein Outlet „erstellt“ oder „gegründet“ wird.

Als Argument bringt **E**, dass „gründen“ der natürlichen Sprache des Händlers entspricht. **F** hingehen räumt ein, dass das Entwicklungsteam in dieser Annahme zu „akademisch“ vorgegangen ist, denn der zukünftige Benutzer des Systems ist nicht der Händler, sondern die MPC der jeweiligen Länder. Ob diese bei der Händlerverwaltung von „gründen“ oder „erstellen“ sprechen, ist nicht bekannt. Die Unsicherheiten bei der Begriffsdefinition lassen sich darauf zurückführen, dass die Benutzer des Systems nicht in die Entwicklung einbezogen wurden.

Als weiteres Problem wird die fehlende Offenheit auf der Fachbereich-Seite genannt, Begriffe zu diskutieren. Diese Aussage lässt sich darauf zurückzuführen, dass Diskussionen im Team über Begriffsdefinitionen durch den PO wiederholt „verkürzt“ wurden, in dem er die Begriffe festlegte und zu keinen weiteren Diskussionen bereit war. Dies ist Ausdruck des Problems, das der Domänenexperte auch in der Rolle des PO tätig ist. Es wird in Kapitel 5.2.4 näher darauf eingegangen.

Die Begriffe aus dem Alt-System Classic und dem neuen System GSSN+ werden vermischt. Im Team wird offensichtlich zu wenig Wert auf eine einheitliche Verwendung der Begriffe gelegt. Im aktuellen Vorgehen erstellt der PO das Glossar. Über die teilnehmende Beobachtung konnte festgestellt werden, dass das Glossar nicht vollständig ist und nicht dem aktuellen Entwicklungstand entspricht. Die

Erklärung dafür ist, dass der PO die Begriffe mit ihrer Definition bereits kennt und die Motivation, das Glossar vollständig und aktuell zu halten, nicht vorhanden ist.

Eine mögliche Rolle des RE in Bezug auf die gemeinsame Sprache wurde bereits in Kapitel 3.5 skizziert: Der Requirements Engineer übernimmt die Verantwortung für das DDD-Glossar und ist verantwortlich für die korrekte Verwendung durch das Team während Diskussionen oder in der Dokumentation der Anforderungen.

5.1.7 Kategorie: Domänenmodell

5.1.7.1 Hintergrund zur Frage

1. Welche Vor- und Nachteile haben die jeweiligen Vorgehensweisen?
2. Nach der Literatur zu DDD entsteht die Context Map in Zusammenarbeit zwischen Domänenexperten und dem Entwicklungsteam. Könntest du dir dieses Vorgehen auch für das GSSN+ Projekt vorstellen?

Die Vorgehensweisen zur Modellierung des Domänenmodells im aktuellen Projekt wurden in Kapitel 4.2.6 dargestellt. Im durchgeführten Interview wurden diesbezüglich sowohl das RE- Team als auch das Entwicklungsteam zu den jeweiligen Vor- und Nachteilen der Vorgehensweisen befragt.

Bis vor wenigen Wochen war das RE-Team dafür zuständig die Context Map zu erstellen. Aktuell liegt dies in der Verantwortung der Entwickler.

5.1.7.2 Ergebnisse

Welche Vor- und Nachteile haben die jeweiligen Vorgehensweisen?

Vorgehen: Das RE-Team arbeitet neue Anforderungen vor der Umsetzung in das Domänenmodell ein.

- Vorteile:
  - „Was ich selbst schreibe, verstehe ich besser.“ (Interview-Anhang A)
  - Gemeinsames Verständnis (Interview-Anhang B)
  - Erstellung der User Storys auf Basis der Context Map möglich (Interview-Anhang B)
- Nachteile:
  - „Den Entwicklern fehlt das Verständnis.“ (Interview-Anhang A)
  - Hoher Abstimmungsaufwand (Interview-Anhang D)
  - Entwickler müssen Freiheitsgrade abgeben (Interview-Anhang B)
  - Nachdokumentation nötig (Interview-Anhang F)
  - Unklar, ob das, was dokumentiert war, schon umgesetzt wurde (Interview-Anhang B)

Vorgehen: Das Entwicklungsteam arbeitet neue Anforderungen nach der Umsetzung in das Domänenmodell ein.

- Vorteile:
  - „Dinge, die mich nicht interessieren, muss ich nicht modellieren." (Interview-Anhang A)
  - „Wir sind freier, die Domänen zu gestalten." (Interview-Anhang D)
  - „Es ist sinnvoll, dass dokumentiert wird, was umgesetzt wurde." (Interview-Anhang E)
  - „Das Entwicklungsteam kann auf Implementierungsebene bei der Modellierung mitreden. Das ist notwendig." (Interview-Anhang E)
- Nachteile:
  - Hoher Aufwand, ein nachträgliches Verständnis im RE-Team zu schaffen (Interview-Anhang B)
  - Fehlende Transparenz (Interview-Anhang B)
  - Context Map ist eine Dokumentation. Die Diskussion über die Fachlichkeit fehlt (Interview-Anhang A)

In der DDD-Literatur entsteht die Context Map in Zusammenarbeit zwischen Domänenexperten und dem Entwicklungsteam. Könntest du dir dieses Vorgehen auch für das GSSN+-Projekt vorstellen?

A:

„Weiß ich nicht. Kenn ich eben so nicht" (Interview-Anhang A).

B:

„Vorstellen ja, bei anderer personeller Besetzung. Beim Fachbereich fehlt einfach die Bereitschaft, sich auf die Domänen einzulassen" (Interview-Anhang B).

D:

„Ich denke, das wäre sinnvoll. So kann man gemeinsam über die Fachlichkeit diskutieren" (Interview-Anhang D).

F:

„Ja, das wäre schön" (Interview-Anhang F).

5.1.7.3 Diskussion

Im DDD erarbeiten und verfeinern Domänenexperten und Entwickler das Domänenmodell gemeinsam (Kap. 3.2.3). Ziel ist, ein gemeinsamen Verständnis zu schaffen und eine enge Bindung zwischen dem Modell der Analyse und der Implementierung herzustellen. Das Vorgehen im untersuchten Projekt

weicht davon ab. Die Probleme, die dabei entstehen, wurden in den Ergebnissen von den Teammitgliedern benannt; sie entsprechen den typischen Problemen, die im DDD entstehen, wenn Analyse- und Codemodell voneinander getrennt werden (siehe Kap. 3.2.9.5).

Ein weiteres Problem ist, dass der Domänenexperte nicht am Prozess der Modellerstellung und Verfeinerung beteiligt ist. Ohne die Beteiligung des Domänenexperten kann aber kein tiefes Domänenwissen im Team aufgebaut werden (s. Kap. 3.2.3). Die fachliche Diskussion anhand des Modells wird vernachlässigt und das Domänenmodell wird als Dokumentation der Implementierung verwendet.

Die Probleme lassen sich im Hinblick auf die Forschungsfrage vermeiden, wenn das Modell in Zusammenarbeit von Domänenexperten und Entwicklern bearbeitet wird. Die befragten Personen äußerten sich über dieses mögliche Vorgehen größtenteils positiv.

Das Requirements Engineering sollte in diesem Prozess in einer moderierenden und unterstützenden Rolle tätig sein. Die direkte Kommunikation zwischen den Domänenexperten und Entwicklern darf weder unterbrochen werden, noch sollte der Requirements Engineer anstelle des Experten eingesetzt werden.

Der Requirements Engineer kann im Prozess der Domänenmodell-Modellierung auf sein Repertoire an modellbasierten Techniken zugreifen und das Team bei der der Modellierung aktiv unterstützen. Dabei hat er außerdem auf die korrekte Verwendung der Begriffe der UL zu achten. Neue Begriffe werden sofort in das UL-Glossar aufgenommen und abgestimmt.

5.1.8 Kategorie: Carving Meeting

5.1.8.1 Hintergrund zur Frage

Zweck des Carving Meetings ist, ein gemeinsames Verständnis zwischen Fachbereich und Entwicklungsteam zu schaffen. In diesem Sinne ist es Teil eines „Knowledge Crunching"-Prozesses, der ein wichtiger Bestanteil der DDD-Vorgehensweise ist (siehe Kap. 3.2.4).

Die Interview-Teilnehmer wurden zu den Aspekten „Häufigkeit", „Dauer", „Anzahl Teilnehmer", und „Methodik" des Carving Meetings als wichtigem Teil des DDD-Vorgehens befragt. Zusätzlich sollten sie weitere Kritikpunkte und Verbesserungsvorschläge nennen.

5.1.8.2 Ergebnisse

Allgemein:

C, **D** und **E** nennen das Carving Meeting als positiven Aspekt der Zusammenarbeit im Team (Interview-Anhang C, D, E).

Häufigkeit und Dauer:

A antwortet, dass die Carving Meetings schon häufig ausgefallen seien, weil stattdessen, bei dringenden Themen, ein Schätzmeeting abgehalten wurde. Als Folge davon seien dann bestimmte Themen ohne Vorbesprechung in das nächste Schätzmeeting aufgenommen worden, was häufig Diskussionen zur Folge hatte. Seiner Meinung nach wäre es sinnvoll gewesen, diese Themen im Carving besprochen zu haben. Die Dauer von 1,5 Stunden findet er ausreichend und sagt, dass die Zeit häufig gar nicht benötigt würde (Interview-Anhang A).

B dagegen hätte gern mehr Zeit für den Termin zur Verfügung (Interview-Anhang B).

E findet die Häufigkeit und die Dauer der Carving-Termine angemessen, schlägt aber vor, bei größeren fachlichen Themen einen einmaligen Workshop von ein oder zwei Tagen Dauer durchzuführen. Als Workshop-Methode schlägt er das Event Storming vor (Interview-Anhang E).

B und **D** finden die Häufigkeit des Carving-Termins „ok" (Interview-Anhang B, D).

Anzahl Teilnehmer:

A äußert sich dazu wie folgt:

> *„Es wäre wichtig, dass auch die richtigen Entwickler dabei sind. Manchmal waren dann welche mit im Carving, die sich bei einem bestimmten Thema nicht so gut auskannten. Da haben wir dann etwas besprochen und eine Woche später hieß es dann, „ne", das können wir so nicht machen. Aber das kam dann erst auf, nachdem die sich im Team dann nochmal abgestimmt hatten. Dann hätten wir das nicht besprechen müssen"* (Interview-Anhang A).

B hält die Anzahl der Carving-Teilnehmer für „ausreichend" (Interview-Anhang B).

E gibt zur Antwort, dass es deutlich besser sei, wenn, so wie im aktuellen Vorgehen, nur zwei Entwickler am Carving-Meeting beteiligt sind, weil dadurch weniger Zeit in Anspruch genommen werde. Andererseits hätten 20 Entwickler mehr Ideen als vier. Durch die Reduzierung der Teilnehmeranzahl ginge auch etwas verloren (Interview-Anhang E).

F findet die Reduzierung der Teilnehmer in den Carving-Meetings „ok", sagt aber gleichzeitig, dass dies nur eine Krücke dazu sei, einen anstehenden Team-Split nicht angehen zu müssen. Er meint, dass eigentlich das ganze Team zu den Carving-Meetings gehen sollte und nicht nur die Vertreter der Teams:

> *„Das Carving ist dazu da, das Gleichgewicht zwischen Entwicklern und Fachbereich herzustellen. Wenn nur Team-Repräsentanten zu den Carving-Terminen hingehen, führt das zu einem Wissensungleichgewicht. Das ist gefährlich: Wenn wir in den Meetings rotieren, dann hängt zu viel Wissen bei einzelnen Leuten"* (Interview-Anhang F).

Methodik und Art der Durchführung:

B empfindet das Vorgehen als „grottig". Er kritisiert, dass man sich den Themen von der Oberfläche her nähert, anstatt über die „Domänen" zu diskutieren. Als Grund nennt er PO S., der unterbunden hätte, dass über die „Domänen" diskutiert wird (Interview-Anhang B).

D antwortet, dass im Carving Meeting keine User Storys besprochen werden müssten. Die fachliche Diskussion sei wichtiger (Interview-Anhang D).

E äußert sich folgendermaßen:

> *„Wir sind immer sehr konkret im Carving; ein andres Format wäre besser“* (Interview-Anhang E).

F schlägt als ideale Methodik einen Event-Storming Workshop vor (Interview-Anhang F).

Weitere Kritikpunkte:

A kritisiert, dass in den Carving-Terminen vom Entwicklungsteam oftmals keine Rückfragen zu den in den User Storys beschriebenen Akzeptanzkriterien kommen würden. Er glaubt, dass einerseits ohne eine Diskussion die User Story von den Entwicklern nicht verstanden werden kann. Andererseits ist er sich nicht sicher, ob es sinnvoll sei, alle Kriterien vorzulesen (Interview-Anhang A).

D sagt, dass es schade sei, dass nicht alle Entwickler am Carving dabei sein können, und fügt dem hinzu:

> *„Ein gemeinsames Event Storming wäre gut. Das darf aber nicht ziellos sein, sondern nur für ein neues Thema, das gerade aktuell ist“* (Interview-Anhang D).

Verbesserungsvorschläge:

C gibt folgende Anstöße:

> *„Mit dem Carving versuchen wir uns über fachliche Themen Gedanken machen. (…) Die Themen drehen so Runden und es wird Feedback generiert. Besser wäre aber, eine Ebene grober auf Epic Ebene zu diskutieren. (…) Das wäre die richtige Granularität. Wir sollten über Epics sprechen. (…) Viele Stories sind eigentlich Epics. Die müssten runtergebrochen werden. Die Aufgabe des RE ist es, das als Epic zu erkennen. In den Epics sollten die Anforderungen aufgeschrieben werden, dann schauen wir gemeinsam, was die Stories dazu sein könnten. Dann bekommt man das in den Griff, dass die Storys oft falsch geschnitten sind“* (Interview-Anhang C).

5.1.8.3 Diskussion

Das Carving Meeting als Teil des „Knowledge Crunching“-Prozesses, bei dem ein gemeinsames Verständnis im Team geschaffen werden soll, wird von den Entwicklern größtenteils als positiver Aspekt der Zusammenarbeit wahrgenommen. Ein häufig angesprochener Punkt ist allerdings die Reduzierung der Teilnehmer des Carving Meetings. Wie in Kapitel 4.2.5 beschrieben, nehmen statt elf Entwicklern nur noch zwei an einem Carving-Termin teil, um den Diskussionsaufwand zu reduzieren. In der Literatur zu DDD wird nicht darauf eingegangen, ob im Rahmen des Knowledge Crunchings das ganze Entwicklungsteam oder nur einzelne Entwickler teilnehmen sollten. Im Rahmen des Interviews wurden allerdings einige Probleme durch die reduzierte Teilnehmerzahl deutlich:

- ➔ Weniger Entwickler generieren auch weniger Ideen.
- ➔ Zur Besprechung bestimmter Themen fehlen die Entwickler mit entsprechendem Know-how.

Als weitere Probleme wurden die Annäherung an Themen anhand des Oberflächenentwurfs genannt sowie eine zu konkrete Herangehensweise. Auch die Besprechung der User Storys wurde als Kritikpunkt angeführt. Ein anderes Format für den Termin ist vom Team gewünscht.

Das Problem der oberflächengetriebenen Vorgehensweise wurde auch über die teilnehmende Beobachtung erkannt und wird in Kapitel 5.2.3.2 an einem Beispiel erläutert.

Im DDD steht das Domänenmodell im Mittelpunkt des Entwicklungsprozesses und ist Kommunikationsgrundlage sowohl für die Analyse als auch für die Umsetzung. Die fachlichen Zusammenhänge, Geschäftsregeln und -prozesse stehen dabei im Vordergrund und sind frei von Aspekten der Oberflächengestaltung. Im aktuellen Vorgehen des GSSN+-Projekts (siehe Kap. 4.2.7) jedoch dienen die User Storys und das Oberflächenkonzept im Carving Meeting als Kommunikationsgrundlage für Domänenexperten, RE- und Entwicklungsteam. Dies widerspricht der DDD-Vorgehensweise.

Im Hinblick auf die Forschungsfrage ist ein mögliches Format der Kommunikation, das auch von den Entwicklern genannt wurde, das Event Storming. Diese Workshop-Methode wurde in Kapitel 3.2.7 vorgestellt und in Kapitel 5.2.2.2 im Zusammenhang mit der Domänenanalyse bereits diskutiert. Die Methode würde sich gut für den Carving-Termin eignen, da sich dadurch das Problem der lösungsorientierten und oberflächengetriebenen Vorgehensweise lösen ließe. Der Requirements Engineer würde dabei in der Rolle des Moderators unterstützen. Auf seine Aufgaben wird dabei in der Diskussion des folgenden Kapitels näher eingegangen.

5.1.9 Kategorie: Event-Storming

5.1.9.1 Hintergrund zur Frage

Was hältst du von Event Storming?

Warum hat das Event Storming nicht funktioniert?

Zu Projektbeginn wurden im GSSN+-Team mehrere mehrstündige Event-Storming Workshops durchgeführt. Nachdem diese Termine, nach Aussage des Teams, nicht zu dem erhofften Ergebnis geführt hatten, wurde auf eine weitere Durchführung des Workshops verzichtet. Die Frage nach dem Event-Storming Workshop wurde gestellt, um zu untersuchen, aus welchem Grund die Methode nicht funktioniert hat.

5.1.9.2 Ergebnisse

D:

> *„Wir hatten keine Erfahrung und haben das erste Mal einen Event Storming Workshop durchgeführt. Wir haben es nicht geschafft, die Vorteile zu vermitteln, die man sich erhofft hatte. Umgekehrt war die Bereitschaft nicht da, sich von den Oberflächen zu lösen. Wir hatten viele Chancen, aber es hat nicht gezündet.*

Wir hatten uns überlegt, von extern eine Moderation zu holen, jemand, der weiß, wie man so einen Workshop durchführt, aber es war dann zu spät.
Dann hat der Fachbereich sich eine Alternative überlegt“ (Interview-Anhang D).

D:

„Das alte Fass mit dem Event Storming würde ich nicht mehr aufmachen. An unserem aktuellen Vorgehen würde ich nicht mehr rütteln. Es funktioniert so, wie es gerade läuft.
Ein anderes Vorgehen wäre richtiger, aber wir sind jetzt durch mit dem Ausprobieren“ (Interview-Anhang D).

E:

„Wir bräuchten mehr Führung. Einer moderiert, es werden Fragen gestellt und einer klebt die Events auf. Nicht alle auf einmal“ (Interview-Anhang E).

5.1.9.3 Diskussion

Die Gründe für das Scheitern werden anhand der Antworten deutlich: Das Team hatte keine Erfahrung mit der Workshop-Methode und die Rolle des Moderators war nicht besetzt.

Im Hinblick auf die Forschungsfrage kann der Requirements Engineer diese Probleme lösen. Voraussetzung dafür ist, dass er sich mit der Methode auseinandergesetzt hat und die Regeln kennt.

Mögliches Vorgehen:

Zu Beginn des Workshops stellt der Requirements Engineer dem Team die Methode vor und übernimmt im Verlauf des Workshops die Moderation. Die wichtigsten Regeln sind auf einem Whiteboard jederzeit vom Team nachzulesen. Neue Begriffe werden durch den Requirements Engineer notiert und im Anschluss in das Glossar übernommen. Außerdem achtet er auf unsaubere Begriffsverwendungen und Begriffe, die synonym verwendet werden. Diese werden noch während des Workshops mit dem Domänenexperten diskutiert und geklärt.

In der Rolle des Moderators unterstützt der Requirements Engineering die Zusammenarbeit der Domänenexperten und Entwickler, ohne dabei die direkte Kommunikation zu unterbrechen.

5.2 Eigene Beobachtungen

5.2.1 Hintergrund

Die eigenen Beobachtungen ergänzen die Interviews. Sie beruhen auf dem persönlichen Erleben als Mitglied des Projektteams und veranschaulichen Problembereiche, die bei der Zusammenführung von Requirements Engineering und Domain-Driven Design auftreten können.

5.2.2 Domäne ist nicht definiert

5.2.2.1 Beobachtung

- **Die Domäne:** Das Team arbeitet mit dem Konzept des Bounded Contexts, hat aber die Definition der Domäne mit ihrer Einteilung in Core- und Subdomänen nicht bedacht.

5.2.2.2 Diskussion

Die Definition der **Domäne** und ihre Unterteilung in verschiedene Subdomänen wurde in Kapitel 3.2.2 erläutert. Eine Missachtung der Unterteilung oder die falsche Definition einer Domäne – etwa bei der Core Domain (Kerndomäne), die definiert, mit welchem Bereich das Unternehmen herausragend sein will, um sich von den Wettbewerbern absetzen zu können –, kann dazu führen, dass das Team sich zu schnell auf Lösungen fokussiert, anstatt zuerst das Problem vollumfänglich zu verstehen. Dies kann zu falschen Entscheidungen bezüglich der Umsetzung führen und die strategische Ausrichtung des Projekts beeinflussen. Zuerst sollte also die Domäne gründlich analysiert werden. Die Erforschung der Domäne ist Teil eines Knowledge Crunching Prozesses (siehe Kap. 3.2.4).

In den theoretischen Grundlagen zum Thema Requirements Engineering wurden bereits einige Methoden zur Anforderungsermittlung erläutert, die sich gut für die Analyse der Domäne eignen. So ist z. B. das *User-Story-Mapping* (siehe Kap. 3.3.4.3) eine Workshop-Methode, die mittels User Storys Aktivitäten und Aufgaben des Benutzer darstellt; Ziel ist, ein gemeinsames Verständnis unter den Teilnehmern davon zu schaffen, welche Probleme ein Produkt lösen muss. Ebenfalls geeignet ist der Event-Storming Workshop. Er wurde in Kapitel 3.2.7 beschrieben und wird in Kapitel 5.1.9.3 diskutiert. Die Methode eignet sich gut, wenn die Domäne am Anfang des Projekts noch unbekannt ist und das schreiben einer Story schwerfällt.

Auch modellbasierte Techniken des Requirements Engineering (siehe Kap. 3.3.4.2) eignen sich für die Analyse.

Im Hinblick auf die Forschungsfrage ist es bei allen Aktivitäten vorrangig, dass die Domänenexperten mit dem Entwicklungsteam zusammenarbeiten. Der Requirements Engineer führt die Workshops durch, moderiert und notiert sich Begriffe für das Glossar der Ubiquitous Language.

5.2.3 Oberflächengetriebenes Vorgehen

5.2.3.1 Beobachtung

Folgende Anforderung wurde durch das RE-Team im Carving Meeting vorgestellt:

> Der *Brand Code*, zur Identifikation eines *Outlets* in Bezug auf die dort angebotenen Fahrzeug-Marken, soll in der Oberfläche dargestellt werden. Dem Code können mehrere Marken hinzugefügt werden. Ein Outlet kann mehrere Codes haben.

Abbildung 6 stellt einen Ausschnitt des Oberflächenkonzepts des Prototyps dar, das im Carving präsentiert wurde.

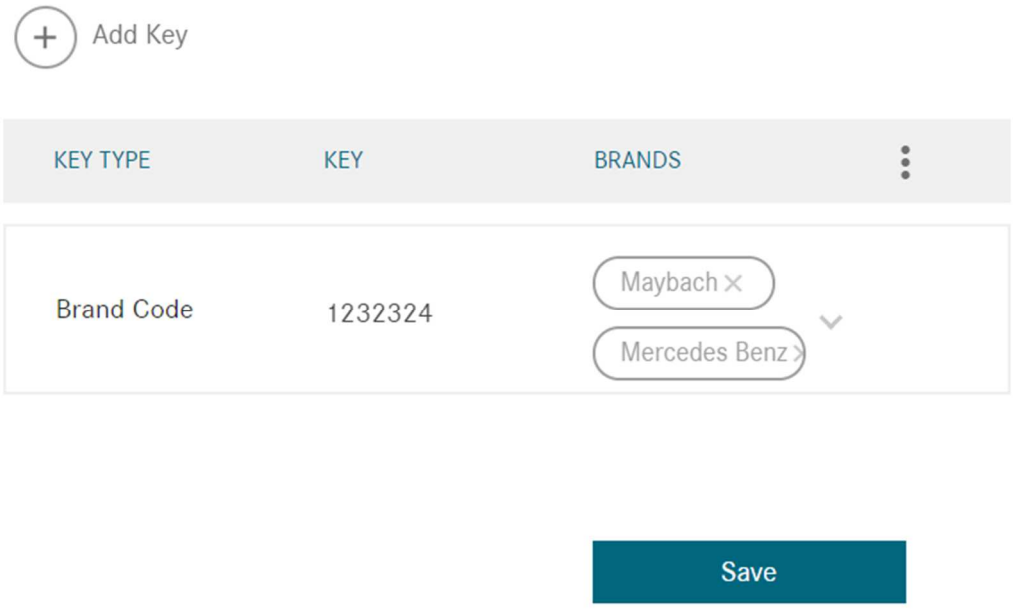


Abbildung 6: Oberflächenkonzept für „Brand Code“

Nach der Besprechung im Carving-Meeting wurde die Anforderung geschätzt und im folgenden Sprint umgesetzt. Bei einem Sprint-Review-Termin kam es zur Diskussion.

Das Entwicklungsteam löste das Problem wie in der Oberfläche dargestellt und legte den Brand Code mit den verschiedenen zugeordneten Marken in der Datenbank ab. Im Sprint-Review wurde klar, dass diese Umsetzung ein Problem darstellt, denn die Bereitstellung der Daten für die Abnehmersysteme erfordert eine 1:1-Zuordnung des Codes zu einer Marke in der Datenbank.

Der PO reagierte mit Unverständnis. Für ihn war klar, dass das Zusammenfassen des Codes mit mehreren Marken nur ein „Oberflächen-Ding“ ist und „natürlich“ im Backend auf andere Art umgesetzt werden muss. Darauf reagierte das Entwicklungsteam mit Erstaunen, denn es hat die Anforderung anhand des Oberflächenkonzepts umgesetzt.

5.2.3.2 Diskussion

Es wird deutlich, dass ein oberflächengetriebenes Entwicklungsvorgehen problematisch ist, denn über die Oberfläche müssen nicht alle fachlichen Zusammenhänge und Abhängigkeiten abgebildet sein. Werden diese in der Umsetzung missachtet, entspricht die Software nicht allen Anforderungen und kann somit den Bedürfnissen des Kunden nicht gerecht werden.

Wie in Kapitel 3.3.4.5 beschrieben, ist es Aufgabe des Requirements Engineers, im Rahmen der Implementierung durch Abstraktion vom Kern des Problems zu lösen.

Eine mögliche Vorgehensweise zur Anforderungsermittlung im DDD in Zusammenarbeit zwischen dem Stakeholder und den Entwicklern ist die des Business Use Cases (siehe Kap. 3.3.4.4). Diese sind

Grundlage für die Entwicklung des Domänenmodells, das mit dem Domänenexperten entwickelt wird. Dieses Vorgehen kann jedoch Probleme aufwerfen, wenn es dem Stakeholder oder den Entwicklern an methodischen Kompetenzen fehlt oder, wie in den Beobachtungen beschrieben, der Stakeholder zu früh in die Lösungsentwicklung einsteigt. Im Hinblick auf die Forschungsfrage kann hier ein Requirements Engineer helfen, indem er lösungsorientierte Diskussionen auf das eigentliche Problem zurücklenkt und bei der Erstellung des Business Use Cases unterstützt. Auch hier könnte er wichtige Begriffe sofort in das Glossar aufnehmen.

Der Business Use Case hilft, das Geschäftsproblem zu verstehen, ohne dabei auf dessen Lösung einzugehen. Das Problem der oberflächengetriebenen Entwicklung ließe sich dadurch lösen. Außerdem würde schon früh das Entwicklungsteam in die Anforderungserhebung eingebunden werden.

5.2.4 Der PO in der Rolle des Domänenexperten

5.2.4.1 Beobachtung

- In Diskussionen zwischen dem Domänen-Experten und dem Entwicklungsteam über fachliche Konzepte oder Begriffe der Domäne scheint der Domänen-Experte oft ungeduldig. Wenn ihm die Diskussionen zu lange dauern, bricht er sie ab, um in seiner Agenda fortzufahren.
- Der Domänenexperte entscheidet in seiner Rolle als PO, dass die Carving-Termine nicht mehr mit dem ganzen Team geführt werden. Grund ist die Länge der Diskussionen. Der Fokus in den Carving Meetings soll außerdem auf den User Storys liegen. Die Storys sollen mit den Entwicklern soweit abgesprochen sein, dass es in den Refinement-Terminen zu weniger Diskussionen kommt und so mehr Storys geschätzt werden können.

5.2.4.2 Diskussion

Hier besteht ein Rollenkonflikt. Der PO hat die Vision, wie das Produkt GSSN+ funktionieren und aussehen soll. Dabei ist er verantwortlich für den Projekterfolg. Sein Fokus liegt auf der Verwaltung und Priorisierung des Backlogs (vgl. Vernon et al., 2017, S. 26). Der Domänenexperte hingegen versorgt das Team mit den benötigten Informationen, damit sich ein gemeinsames Verständnis des Problems im Team entwickeln kann. Seine Verantwortung besteht darin, sein Wissen zu vermitteln und bei Rückfragen zur Verfügung zu stehen. Wird die Rolle des Domänenexperten nicht vollständig ausgefüllt, kann dies schwerwiegende Folgen für das Projekt haben.

Die beiden Rollen sollten nicht durch eine Person ausgeführt werden, denn die unterschiedlichen Verantwortlichkeiten der Rollen stehen im Widerspruch zueinander.

In den Beobachtungen wurde ersichtlich, dass die fachliche Diskussion im Team durch das Bestreben des Product Owners, zügig in der Entwicklung voranzukommen, vernachlässigt wird. Damit fehlt dem Team die Voraussetzung, ein gemeinsames Verständnis zu entwickeln. Wichtige Konzepte können

dadurch von den Entwicklern nicht in Gänze verstanden werden; das entwickelte Produkt entspricht am Ende der Entwicklung nicht den fachlichen Anforderungen.

Im Hinblick auf die Forschungsfrage ist es Aufgabe des Requirements Engineers, Konflikte im Entwicklungsprozess zu erkennen und aufzulösen (siehe Kap. 3.3.6).

Die Konfliktauflösung besteht in diesem Fall darin, einen Domänen-Experten in das Projekt einzubinden, der über die Wissensvermittlung hinaus keine politischen oder strategischen Interessen an der zu entwickelnden Software hegt. Wie in der Diskussion zur Kategorie „Domänenexperte“ (siehe Kap. 5.1.5.3) bereits beschrieben, sollte der Requirements Engineer bei der Ermittlung und Pflege der Stakeholder bereits mögliche Domänenexperten identifiziert haben. Gemeinsam mit dem Product Owner sollte zu Projektbeginn entschieden werden, welcher Domänenexperte sich besten für das Projekt eignet. Den Experten dafür zu motivieren ist eine Herausforderung, auf die hier nicht weiter eingegangen wird.

6 FAZIT

6.1 Zusammenfassung

Das Forschungsziel der Arbeit war die kritische Auseinandersetzung mit der Rolle des Requirements Engineering im Domain-Driven Design. Dabei sollte die Forschungsfrage beantwortet werden, ob das Requirements Engineering mit den Prinzipen des Domain-Driven Design zu vereinbaren ist und wie es einen Entwicklungsprozess, in dem Domain-Driven Design angewendet wird, unterstützen kann.

Daraus abgeleitet wurden die Fragen untersucht, welche Funktionen des Requirements Engineering den Prinzipien von Domain-Driven Design widersprechen, welche Aufgaben und Verantwortlichkeiten aus dem Requirements Engineering in einem Vorgehen mit DDD ausgeführt werden können und welche Methoden und Techniken dazu eingesetzt werden können. Darüber hinaus sollte überprüft werden, welche Verantwortlichkeiten aus dem Requirements Engineering von anderen am Projekt beteiligten Rollen übernommen werden können.

Zu diesem Zweck wurde eine Fallstudie durchgeführt, bei der das qualitative Interview und die teilnehmende Beobachtung als Methoden der Datenerhebung angewendet wurden.

Als Ergebnis zeigte sich, dass das Requirements Engineering in einer Schnittstellenfunktion zwischen Fachbereich und Entwicklungsteam Probleme aufwirft, die einem aus den DDD-Prinzipien abgeleiteten Vorgehen zur Softwareentwicklung im Weg stehen. Diese Erkenntnis deckt sich mit den Aussagen der DDD-Literatur, in der ein Requirements Engineer niemals der direkten Kommunikation zwischen Domänen-Experten und Entwicklern im Wege stehen darf.

Gleichzeitig konnten aber verschiedene Probleme identifiziert werden, die ein Requirements Engineering im DDD lösen kann und womit es der Projektentwicklung nützt. Diese Erkenntnis ist ein Beitrag zur Schließung der Forschungslücke hinsichtlich der Rolle des Requirements Engineering in Domain-Driven Design.

Probleme, die mithilfe der Fallstudie zum Einsatz von RE im DDD identifiziert werden konnten, zählen unklare Ziele, eine unklare Vision, wenig Erfahrung mit DDD, der Domänenexperte als Product Owner, das Domänenmodell, das Carving Meeting, das RE, die gemeinsame Sprache, die oberflächengetriebene Vorgehensweise und nicht definierte Domänen.

Unklare Ziele

Die Projektentwicklung kann in die falsche Richtung laufen, wenn dem Entwicklungsteam die kurz- und langfristigen Ziele der Projektentwicklung unklar sind. Als Gründe für die Unklarheit konnten identifiziert werden: die schwammige Definition der Ziele, häufige Änderungen der Ziele sowie die fehlende Kommunikation der Ziele durch den Product Owner. Die Folge im betrachteten Projekt war, dass

sich das Entwicklungsteam ein eigenes Projektziel definierte, das nicht mit den Zielen des Product Owners übereinstimmte.

Das Requirements Engineering kann einen Entwicklungsprozess auf Basis des Domain-Driven Design fördern, wenn es die Zielformulierung und die Kommunikation verantwortet oder unterstützt. Geeignet sind dazu Workshop-Formate, die auf der PAM-Methode basieren.

Unklare Vision

Die Vision soll dem Entwicklungsteam den Gesamtüberblick über die Projektentwicklung geben. Fehlt die Vision oder ist sie nicht klar, ist auch die Richtung der Entwicklung unklar. Eine unklare Vision kann zurückgeführt werden auf sich häufig ändernde Produktvisionen und deren fehlende Kommunikation.

Bei der Verbindung von RE und DDD kann der Requirements Engineer den Product Owner bei der Aufgabe, eine klare Vision zu formulieren und regelmäßig zu kommunizieren, unterstützen. Als Methode dafür eignet sich der Produktvision Workshop. Die Verbindung zwischen Produktvision und Domänenmodell schafft das Domain Vision Statement.

Wenig Erfahrung mit DDD

Mangelnde Erfahrung mit DDD beeinträchtigt den Erfolg der Projektentwicklung. Im betrachteten Fall hatte das gesamte Team hatte zu Projektbeginn wenig oder keine Erfahrung mit der Herangehensweise DDD.

Der Requirements Engineer kann bei der Verbindung von RE und DDD das Entwicklungsteam stärken, wenn er sich selbst gründlich mit DDD auseinandergesetzt hat. Er ist allerdings nicht dafür verantwortlich, das Team in DDD zu unterrichten.

Der Domänenexperte

Der Domänenexperte als Fachexperte ist – neben dem Product Owner als Produktexperte – eine der Schlüsselfiguren des Entwicklungsprozesses. Wenn der Domänenexperte gleichzeitig in der Rolle des Product Owners tätig ist, besteht ein Rollenkonflikt: Die strategischen Interessen und die Fokussierung auf den Projektfortschritt beschränken die Zusammenarbeit zwischen Entwicklungsteam und Domänenexperte soweit, dass ein an den DDD-Prinzipen orientiertes Vorgehen unmöglich ist. In konkreten Fall wurde außerdem die Eignung des Domänenexperten hinsichtlich seines Domänenwissens durch das Entwicklungsteam infrage gestellt.

Die Aufgabe des Requirements Engineers bei der Zusammenführung der Prinzipien von RE und DDD kann es sein, im Rahmen der Stakeholder-Analyse die geeigneten Domänenexperten ausfindig zu machen.

Das Domänenmodell

Das Domänenmodell ist die Grundlage der Entwicklung und Implementierung eines neuen Produkts. Die Modellierung des Domänenmodells ohne die Beteiligung des Domänenexperten ist dabei nicht zielführend: Ein solches Vorgehen führt weder zu einem gemeinsamen Verständnis im Team noch zu einer gemeinsamen Sprache.

Im betrachteten Fall wurden zwei vom Team bereits angewendete Vorgehensweisen zur Modellierung der Domäne gegenübergestellt und diskutiert – ohne Beteiligung des Domänenexperten. Nachteile lassen sich im Sinne der Forschungsfrage vermeiden, wenn das Domänenmodell in Zusammenarbeit zwischen Experten, Entwicklungsteam und RE-Team erstellt wird. Der Requirements Engineer kann dabei eine moderierende Rolle einnehmen und ist hilfreich, wenn er dabei auf die korrekte Verwendung der Ubiquitous Language achtet.

Das Carving Meeting

Carving Meetings sind wöchentliche Termine zur Vorstellung neuer fachlicher Themen durch das RE-Team in Anwesenheit des Domänenexperten. Im konkreten Fall wurden sie vom Entwicklungsteam mehrheitlich als positiver Aspekt der Zusammenarbeit angeführt, doch behindert die aktuelle Vorgehensweise eine fachliche und lösungsneutrale Diskussion: Nicht das Domänenmodell wird als Kommunikationsgrundlage verwendet, sondern User-Stories. Darin widerspricht das Vorgehen den DDD-Prinzipen.

Bei einer Einbindung von RE in DDD eignet sich das Workshop-Format „Event Storming“ besser. Der Requirements Engineer kann dabei die Rolle des Moderators einnehmen. Er leitet das Team durch den Workshop, achtet auf die korrekte Verwendung der gemeinsamen Sprache und nimmt neue Begriffe in das Glossar der Ubiquitous Language auf.

Requirements Engineering

Das RE-Team nimmt eine Schnittstellenfunktion ein und agiert als „Mittelsmann“ zwischen dem Fachbereich und dem Entwicklungsteam. Es unterstützt den Product Owner, in dem es das Schreiben der User Storys übernimmt und sich dazu mit dem Entwicklungsteam während der Carving-Termine abstimmt. Die Folge davon ist ein erhöhter Abstimmungsaufwand durch Dreiecksdiskussionen, die sich über diese Konstellation ergeben.

Insgesamt wird das der Einbezug des RE in das DDD aber vom Team als nützlich erachtet, da weder der PO noch das Entwicklungsteam Zeit hätten, die User Storys zu schreiben.

Gemeinsame Sprache

Die gemeinsame Sprache (Ubiquitous Language) ist eine wichtige Voraussetzung für das gemeinsame Problemverständnis und die Implementierung. Werden Begriffe aus dem Alt- und dem Neusystem vermischt, verschiedene Begriffe synonym verwendet und Begriffe der Domäne nicht gründlich genug

diskutiert, kommt es zu Diskussionen sowie Missverständnissen bei der Modellierung und der anschließenden Umsetzung der Konzepte.

Als Ursache stellte sich im betrachteten Fall die fehlende Offenheit des Domänenexperten heraus, die Begriffe und Konzepte der Problemdomäne zu diskutieren, sowie dessen Abwesenheit bei der Modellierung des Domänenmodells. Das Projekt-Glossar wird zwar durch den Product Owner gepflegt, ist aber nicht vollständig und wird vom Team nicht genutzt.

Bei der Verbindung von RE und DDD könnte der Requirements Engineer das Entstehen einer gemeinsamen Sprache fördern, indem er die Meetings und Workshops moderiert und ein Glossar anlegt und pflegt.

Oberflächengetriebene Vorgehensweise

Die Fokussierung der Kommunikation auf die Oberfläche des zu entwickelnden Produkts anstatt auf die Domänen läuft dem Prinzip des DDD entgegen. So wird eine fachliche und lösungsneutrale Diskussion über die Domänenkonzepte verhindert. Dies führt dazu, dass das eigentliche Problem vom Team nicht richtig verstanden wird und die entwickelte Software die Anforderungen nicht erfüllt.

Im konkreten Fall wurde die Fokussierung des Product Owners und Domänenexperten auf die Benutzeroberfläche über alle Kategorien des Interviews hinweg vom RE- und Entwicklungsteam kritisiert.

Bei der Kombination von RE und DDD ist es die Aufgabe des Requirements Engineers, bei der Anforderungserhebung mittels Abstraktion den Kern des Problems von den Aspekten der Implementierung zu trennen.

Nicht definierte Domänen

Die Domäne mit ihrer Einteilung in Core- und Subdomänen ist die Kommunikations- und Arbeitsgrundlage aller an der Entwicklung Beteiligten. Wird die Domäne nicht richtig abgegrenzt und beschrieben, fehlt das Problemverständnis und damit die strategische Ausrichtung und es wird zu früh auf Problemlösung gesetzt.

Im betrachteten Fall arbeitet das Team mit dem Konzept des Bounded Contexts, hat aber die Definition der Domäne nicht bedacht. Dieses Problem kann auf die fehlende Erfahrung und Unwissenheit des Teams mit Domain-Driven Design zurückgeführt werden.

In der Verbindung von RE und DDD sollte darauf geachtet werden, dass zumindest der Requirements Engineer ein ausreichendes Wissen über die Vorgehensweisen im DDD hat.

6.2 Schlussfolgerung

Durch die Arbeit konnte die Forschungsfrage beantwortet werden.

Die Erkenntnisse stützen die Hypothese:

> H1: Das Requirements Engineering kann einen auf Domain-Driven Design basierenden Softwareentwicklungsprozess unterstützen, wenn es nicht klassisch in einer Schnittstellenfunktion zwischen Fachbereich und Entwicklungsteam agiert

Die Probleme, die im Rahmen der Interviews und teilnehmenden Beobachtung festgestellt werden konnten, lassen sich zum größten Teil auf eine falsche oder unzureichende Anwendung und Umsetzung der DDD-Prinzipien zurückführen.

Als Problemlösungen bieten sich an:

Bei Carving-Meetings sollte, wie in DDD vorgesehen, das Domänenmodell als Kommunikationsgrundlage verwendet werden. Dadurch wird der Fokus automatisch auf die Problemdomäne gelenkt. So können die relevanten Informationen herausgearbeitet werden, das gemeinsame Verständnis im Team wächst und eine gemeinsame Sprache kann sich entwickeln.

Bei der Modellierung des Domänenmodells sollte der Domänenexperte, wie in DDD vorgesehen, an der Modellierung beteiligt werden. Dies hätte auch positiven Einfluss auf die gemeinsame Sprache im Team.

Die Zusammenarbeit zwischen dem Domänenexperten und dem Entwicklungsteam kann verbessert werden, wenn der Rollenkonflikt zwischen Product Owner und Domänenexperte gelöst wird, also die Personalunion aufgehoben wird: Der Domänenexperte darf keine politischen, strategischen oder organisatorischen Interessen an der Entwicklung der Software haben. Sein einziges Interesse darf die Vermittlung seines eigenen Wissens sein. Ohne einen geeigneten Domänenexperten ist es nicht möglich, die Vorteile einer DDD-Herangehensweise auszuschöpfen.

Das Problem eines unerfahrenen Teams in Bezug auf DDD kann das Requirements Engineering nicht beheben. Dennoch kann das Requirements Engineering in einem Vorgehen mit DDD nützlich sein. So kann bei einem Scrum-Vorgehen der Requirements Engineer z. B. den Product Owner in seinen Aufgaben, zum Beispiel bei der Definition der Vision und Ziele, unterstützen. Außerdem hilft das Requirements Engineering, die Anforderungen eines allzu visionären Product Owners von Aspekten der Implementierung zu trennen, damit das zugrundeliegende Problem erkannt werden kann. Auch durch das Schreiben der User Storys kann der Requirements Engineer den Product Owner unterstützen.

Verfügt der Product Owner selbst über Kompetenzen und Fähigkeiten eines Requirements Engineers und ist die Unterstützung durch einen Requirements Engineer nicht notwendig, kann Letzterer dennoch wichtige Aufgaben und Verantwortlichkeiten in einem Vorgehen mit DDD übernehmen. Verantwortet er das Glossar und achtet er in den Diskussionen auf die Einhaltung der gemeinsamen Sprache, ist er dem Team sehr nützlich; als nicht an der Diskussion beteiligter Außenstehender ist er dafür bestens geeignet. In der Rolle des Moderators kann der Requirements Engineer außerdem helfen, die Diskussionen und Workshops des „Knowledge Crunchings“-Prozesses zu leiten – und auch hier auf die Verwendung der gemeinsamen Sprache achten.

Nicht zu vereinen ist das Requirements Engineering mit dem DDD in seiner Funktion als Schnittstelle zwischen Fachbereich und Entwicklungsteam. Verhindert das RE die direkte Kommunikation zwischen Domänenexperten und Entwicklern, führt das zu Problemen, die eine Herangehensweise mit DDD verhindert. Diese Erkenntnis deckt sich mit dem Stand der aktuellen Forschung in diesem Punkt.

6.3 Kritische Würdigung

Da im untersuchten Entwicklungsprozess zu viele Probleme ein Vorgehen mit DDD verhindert haben, war es nur begrenzt möglich, wissenschaftliche Erkenntnisse zur Rolle des Requirements Engineers im Domain-Driven Design zu gewinnen. Es ist anzunehmen, dass eine Untersuchung unter Projektbedingungen, die eine Herangehensweise nach DDD erlauben, zu mehr Erkenntnissen in Bezug auf die Forschungsfrage führt. Dies bleibt nun für zukünftige Forschungsprojekte offen.

Im Rahmen der Arbeit konnte nicht auf alle Probleme eingegangen werden. So bleibt z. B. die Frage unbeantwortet, inwieweit sich der Domänenexperte mit den Konzepten und Prinzipien des Domain-Driven Designs auskennen sollte.

Des Weiteren war es im Rahmen des Projekts nicht zu realisieren, mögliche Verantwortlichkeiten und Aufgaben eines Requirements Engineers in DDD in der Praxis zu testen und auf ihre Anwendbarkeit zu überprüfen. Die Arbeit beschränkt sich daher auf die Entwicklung von Handlungsempfehlungen.

Zu beachten ist außerdem, dass die Erkenntnisse auf einer Fallstudie mit nur begrenztem Umfang an Interviews beruhen: Von den elf Entwicklern des GSSN+-Projekts hatten sich nur sechs mit der Herangehensweise DDD auseinandergesetzt. Im Zeitfenster, in denen die Interviews mit dem Team geführt wurden, konnten leider zwei dieser Entwickler nicht an den Befragungen teilnehmen. Daher war die Teilnehmeranzahl der Entwickler beschränkt.

7 HANDLUNGSEMPFEHLUNGEN

Aus den Untersuchungsergebnissen und den Recherchen in der Fachliteratur lassen sich für das Requirements Engineering in einem Softwareentwicklungsprozess, in dem DDD angewendet wird, die folgenden Handlungsempfehlungen ableiten.

Identifiziere den Domänenexperten

Identifiziere zu Projektbeginn, im Rahmen der Stakeholder-Analyse, einen geeigneten Domänenexperten. Das ist die Person im Unternehmen, die sich, unabhängig von ihrer Rolle, am besten in der betrachteten Domäne auskennt. Achte darauf, dass der Domänenexperte nicht in anderen Rollen an der Projektentwicklung beteiligt ist, um einen Rollenkonflikt zu vermeiden (s. Kap. 5.2.4).

Die Integration eines Experten muss natürlich mit dem Auftraggeber abgestimmt sein und durch ihn beauftragt werden. Um den Auftraggeber für die Einbindung eines Domänenexperten in den Entwicklungsprozess zu gewinnen, muss ihm dessen Nutzen für das Projekt deutlich gemacht werden.

Gibt es keine Möglichkeit, einen Domänenexperten in das Projekt zu involvieren, sollte nicht nach DDD vorgegangen werden. Ohne den Domänenexperten ist die Herangehensweise nicht zielführend.

Produktvision und Ziele

Die Produktvision und die langfristigen Ziele sollten zum Projektbeginn festgelegt und an das DDD-Team kommuniziert worden sein. Überprüfe dies mit dem Projektverantwortlichen.

Sind Vision und Ziele noch nicht definiert, ist es deine Aufgabe, dies in Zusammenarbeit mit dem Projektverantwortlichen und dem Entwicklungsteam zu tun. Die Einbindung des Entwicklungsteams folgt dabei den DDD-Prinzipien. Die Zusammenarbeit hat das gemeinsame Verständnis im DDD-Team zum Ziel. Wenn ein geeigneter Domänenexperte bereits identifiziert ist, sollte dieser ebenfalls in diese Aufgabe involviert werden.

Für die Definition der Ziele im Team eignet sich z. B. das Workshop-Format PAM (s. Kap. 3.3.3.2). Stelle sicher, dass die Produktvision und die Ziele regelmäßig an das Team kommuniziert werden.

Domain-Workshop

Dringe darauf, dass zu Beginn der Projektentwicklung eine Domänenanalyse durchgeführt wird, damit das DDD-Team die Geschäftsprozesse, Aktivitäten und Geschäftsregeln der betrachteten Domäne zu verstehen lernt. Das Ziel ist die Unterteilung der Domäne in verschiedene Subdomänen und die Identifizierung der *Core Domain* (s. Kap. 3.2.2). Für die Domänenanalyse eignet sich z. B. ein Event-Storming Workshop (s. Kap. 3.2.7), an dem der Domänenexperte, das Entwicklungsteam und die relevanten Stakeholder teilnehmen sollten.

Als Requirements Engineer kann es deine Aufgabe sein, diesen Workshop zu organisieren und zu moderieren. Dafür solltest du dich mit der Methode auseinandergesetzt haben, um den Ablauf des Workshops dem Team zu Beginn erklären zu können.

Das Glossar der Ubiquitous Language

Erstelle ein Projekt-Glossar, das allen Projektbeteiligten zugänglich ist. Das Glossar sollte die wichtigen Begriffe der Domäne und des DDD enthalten. Dadurch lässt sich z. B. vermeiden, dass Begriffe wie „Bounded Context“ und „Domäne“ synonym verwendet werden.

Achte darauf, dass das DDD-Team die Begriffe korrekt verwendet. Gibt es Unstimmigkeiten oder Missverständnisse bei der Verwendung bestimmter Begriffe, kläre diese umgehend mit dem Domänenexperten und dem Entwicklungsteam.

Achte bei allen Diskussionen zwischen Domänenexperten und Entwicklungsteam darauf, ob Begriffe verwendet werden, die noch nicht in das Glossar aufgenommen wurden, und vervollständige mit diesen das Glossar.

Kein Mittler zwischen den zwei Welten

Achte darauf, nicht als Vermittler von Informationen zwischen dem Domänenexperten und dem Entwicklungsteam zu agieren. Die Kommunikation zwischen den beiden Parteien sollte immer direkt laufen. Dadurch werden Dreiecksdiskussionen und der Verlust von Informationen vermieden. Nimm an den Terminen des *Knowledge Crunchings* (s. Kap. 3.2.4) als Moderator teil und achte auf die Verwendung der gemeinsamen Sprache.

GLOSSAR

Fachbegriff Englisch	Fachbegriff Deutsch	Erklärung
Aggregate	Aggregat	Transaktionale Konsistenzgrenze um zusammengehörige Entities und Value Objects (Vernon, Lilienthal, & Schwentner, 2017, Deckblatt Innenseite).
Bounded Context	Begrenzter Kontext	Semantische Grenze, innerhalb derer die Elemente des Softwaremodells eine bestimmte, konsistente Bedeutung haben (Vernon et al., 2017, Deckblatt Innenseite).
Command	Kommando	Aktion (meinst vom Benutzer ausgelöst), die auf einem Aggregat ausgeführt wird und ein Domain Event auslöst (Vernon et al., 2017, Deckblatt Innenseite).
Context Map	Kontextlandkarte	Abbildung, in der das Verhältnis und die Integration von Bounded Contexts dargestellt wird (Vernon et al., 2017, Deckblatt Innenseite).
Core Domain	Kerndomäne	Subdomain, in der das Unternehmen herausragend sein will, um sich am Markt von Wettbewerbern abzuheben (Vernon et al., 2017, Deckblatt Innenseite).
	DDD-Team	Das Projektteam bestehend aus Domänenexperten, Entwicklern, Projektverantwortlichen und dem Requirements Engineering in einem Vorgehen mit DDD
Domain Expert	Domänenexperte	Person, die ein tief greifendes Verständnis von der Domäne und ihren Abläufen hat (Vernon et al., 2017, Deckblatt Innenseite).
Entity	Entität	Ein individuelles Objekt oder Konzept aus der Domäne. Hat immer eine Identität und meistens einen veränderlichen Zustand (Vernon et al., 2017, Deckblatt Innenseite).
Outlet	Betrieb	Ein Betrieb kann eine rechtliche Einheit oder Betriebsstätte sein. Ein Betrieb ist eine gewerbliche Einheit.

Fachbegriff Englisch	Fachbegriff Deutsch	Erklärung
Subdomain	Teildomäne	Teil der das ganze Unternehmen umfassenden Geschäftsdomäne. Mittel, um den Problemraum besser zu verstehen. Fachgebiet mit spezieller Expertise (Vernon et al., 2017, Deckblatt Innenseite).
Supporting Domain	Unterstützende Teildomäne	Subdomain, ohne die die Core Domain keinen Erfolg haben kann, in der aber keine Exzellenz nötig ist. Eine Individualsoftware ist erforderlich, weil es keine Standardlösung gibt (Vernon et al., 2017, Deckblatt Innenseite).
Ubiquitous Language	Allgegenwärtige Sprache	Gemeinsame Sprache von Softwareentwicklern und Domänenexperten. Wird sowohl von den Teammitgliedern gesprochen als auch im Softwaremodell implementiert (Vernon et al., 2017, Deckblatt Innenseite).
Value Object	Wertobjekt	Ein Wert ist unveränderlich und besitzt keine Identität. Äquivalenz wird durch Vergleich der Attribute bestimmt (Vernon et al., 2017, Deckblatt Innenseite).

ANHANG

E-Mail

„Hallo zusammen,

wie ihr ja schon mitbekommen habt schreibe ich aktuell an meiner Master-Thesis. Das Thema ist ungefähr: Die Rolle des Requirements Engineerings in Domain-Driven Design.

Es geht darum zu überprüfen ob, wo und wie ein RE Team in einem Vorgehen mit DDD Sinn macht. Dazu beschreibe ich erst mal unser Vorgehen als Ist-Analyse und möchte dann alles Weitere auf Grundlage von Interviews untersuchen. Kritik, Anmerkungen und Verbesserungsvorschläge zu den Themen DDD, RE, Zusammenarbeit, Carving, Context Map und User Stories würde ich gern von euch abfragen.

Würdet ihr mir ein Interview geben? Geplant sind Einzelgespräche. Die Interviews sind anonym, werden nicht auf Tonband aufgenommen und die Master-Thesis wird nicht veröffentlicht. Ich schreibe die Arbeit nicht in Kooperation mit Daimler.

Da ich selbst Teil des RE Teams bin ist mir klar, dass es etwas problematisch sein könnte, wenn ihr euch kritisch zum Thema RE äußern sollt. Ich wäre aber sehr froh, wenn ihr das trotzdem tätet, weil es für mich in aller erster Frage darum geht, ob ein RE Team überhaupt mit DDD zu vereinbaren ist.

Mit Philipp habe ich das Interview gestern zum Test durchgeführt. Es hat in etwa eine Stunde gedauert. Wir haben aber auch viel diskutiert dabei. Wahrscheinlich geht es auch schneller, je nachdem wie viel man zu sagen hat.

Wenn ihr Lust dazu hättet wäre ich sehr froh, bin aber nicht beleidigt, wenn jemand nicht mitmachen möchte oder keine Zeit hat. Falls ihr noch andere Kollegen aus dem Team wisst, die da gern was zu sagen würden, bitte die Mail einfach weiterleiten. Ich habe auch nicht alle Mail Adressen.

Zeitlich würde ich mich nach Euch richten. Ich komme auch nach der Arbeit oder am Wochenende gern nach Ulm.

Ihr könnte mir ja bei Interesse einfach kurz Rückmeldung geben und den Rest könnten wir dann vor Ort besprechen.

Danke schon mal und schönes Wochenende

Mirka“

Interview A

Interview zum Thema „Die Rolle des Requirements Engineerings (RE) in Domain-Driven Design (DDD)“

Kennung: A
Datum: 03.05.2018
Berufserfahrung: 6 Monate

Einleitung

- Was ist deine Rolle im aktuellen Projekt GSSN+?

  PO Unterstützung als Teil im RE Team

- Wieviel Spaß macht dir deine Arbeit im Projekt GSSN+ aktuell?

  Es gibt immer wieder interessante Themen, bei denen die Arbeit richtig Spaß macht. Ab und zu gibt es allerdings Abhängigkeiten zum PO und UX und man muss warten – das ist nicht so toll.

- Wie viel Erfahrung hast du bereits praktisch und/oder theoretisch mit DDD sammeln können?

  Mit DDD bin ich das erst erste Mal in diesem Projekt in Berührung gekommen. Theoretische Grundlagen habe ich mir am Anfang des Projekts in Büchern und über das Internet angelesen. Praktisch arbeiten wir damit im Projekt, aktuell machen wir mit DDD nicht mehr so viel, da die Context Map jetzt von den Entwicklern gepflegt wird.

- Wieviel Erfahrung hast du mit Requirements Engineering, sowohl praktisch als auch theoretisch?

  In Hochschulprojekten habe ich in der Entwicklung von der Konzeption bis zum Prototyp RE Tätigkeiten durchgeführt. Auch in meiner Master-Thesis war RE ein Thema und ich habe Anforderungen aufgenommen und dokumentiert. Theoretische Grundlagen habe ich an der Uni vermittelt bekommen. Aktuell lerne ich im Projekt GSSN+ aus den Rückmeldungen meiner direkten Teamkollegen aus dem RE und auch über die Rückmeldungen aus den Refinement Meetings.

- Hast du selbst schon einmal Methoden, Techniken oder Vorgehensweisen aus dem RE angewendet?

  Ja, in der Uni und im Projekt: User-Centered Design und Domain-Driven Design, Use Cases, User Stories, und Szenarios.

- Sind dir die Projektziele und Iterationsziele des Projekts GSSN+ stets klar?

  Projektziel ist klar: Ablösung von GSSN-Classic. Die Iterationsziele sind nicht alle klar bzw. ergeben sich im Laufe der Iteration.

Zusammenarbeit im aktuellen Vorgehen

- Wie bewertest du die Zusammenarbeit im ganzen Team oder zwischen den einzelnen Teams? Welche Aspekte der Zusammenarbeit gefallen dir gut? Welche Aspekte gefallen dir nicht so gut? Was könnte man verändern damit sich die Zusammenarbeit im gesamten Team oder zwischen den einzelnen Funktonen verbessert?

RE und Entwicklungsteam:

Ganz gut momentan, es gibt viele Diskussionen, gemeinsam kommt man zu einer Lösung. Nicht so gut: Dinge werden vom Entwicklungsteam häufig in Frage gestellt. Brauchen wir das überhaupt? Wenn der PO das will, ist es so. Einwände sind ja schon ok. Wenn der PO aber etwas entschieden hat, dann sollte das auch akzeptiert werden. Oder es sollten dann wenigstens alternative Lösungsvorschläge gemacht werden. Außerdem dauert eine Reaktion auf Mails an den Verteiler häufig lange. Es scheint sich keiner verantwortlich zu fühlen. Schreibt man Kollegen direkt an funktioniert die Kommunikation aber gut. Es fehlt ein Ansprechpartner für allgemeine Fragen.

RE und Fachbereich:

Eigentlich ganz gut. Es gibt aber viele Abhängigkeiten. Entscheidungen sind Hü und Hot. Es gibt viele Umentscheidungen durch den PO. Manchmal erscheint eine Entscheidung willkürlich. Es würde helfen, wenn man eine Begründung für eine Entscheidung oder Umentscheidung mitbekommen würde.

Entwicklungsteam und Fachbereich:

ähnlich wie bei RE

Allgemein:

DDD

- Was ist für dich DDD? Wie würdest du in ein oder zwei Sätzen erklären, was DDD ist?

  Man hat eine Context Map. Die Domänen werden unabhängig voneinander betrachtet, jeder Bereich hat seine Funktion in der Domäne. Konzeption erfolgt unabhängig zwischen den einzelnen Domänen.

- Wo siehst du Vorteile der Anwendung von DDD im aktuellen Projekt GSSN+?

  Ich sehe keine Vorteile, aber auch keine Nachteile. In unserem Projekt unterstützt es die Anforderungsanalyse durch die gegebene Architektur mit vielen Services.

- Wo siehst du Schwächen in der Anwendung von DDD im aktuellen Projekt GSSN+?

  Siehe Frage drüber.

- Welche Probleme gibt es aktuell in der Anwendung von DDD im aktuellen Projekt?

  Die Diskussion über die Context Map fehlt. Das weiß ich gehört eigentlich zu DDD. Es gibt Situationen in denen die Bedeutung der Context Map für mich (als RE) nicht klar ist bzw. ich auch ohne auskommen könnte. Für andere Situationen macht aber oft schon Sinn, dass ich mal was nachschauen kann und verstehe was da beschrieben steht und man die Hintergründe kennt.

- Wie könnten diese gelöst werden?
  Werden deiner Meinung nach fachliche Themen zwischen Fachbereich und Entwicklern ausreichend diskutiert, sodass sich gemeinsames Verständnis entwickeln kann?

  Es gibt ja Dinge über die muss man einfach nicht diskutieren. Die sind einfach so, oder werden vom PO so vorgegeben. Fachliche Themen besprechen wir im RE ja mit dem Fachbereich und schreiben dazu die Stories. Diese werden dann den Entwicklern im Refinement vorgestellt. Diese fachliche Diskussion vorab zwischen RE und dem Fachbereich fehlt dem Entwicklerteam dann natürlich. Das Carving fällt ja auch oft aus.

Wir machen stattdessen dann ein Schätzmeeting. Manche Themen wie die Suche werden vom PO und UX diskutiert und uns dann als Lösung zum ersten Mal präsentiert. Vielleicht hätte da eine Besprechung im Team etwas gebracht.

- In DDD arbeitet der Domänenexperte direkt mit dem Entwicklerteam zusammen.
  Wer übernimmt deiner Meinung nach im aktuellen Projekt die Rolle des Domänenexperten?

  Das RE Team und die POs sind eigentlich die Domänenexperten. Sasa und Tim sind die richtigen Experten, und wir sind dann die Experten auf einem kleinen Gebiet.

- Hat dieser/diese deiner Meinung nach ausreichendes Domänenwissen um das Projekt voran zu bringen?

  Ja, die richtigen Domänenexperten Sasa und Tim wissen was sie wollen. Haben einen Plan was das System können soll. bringt das Projekt voran

RE

- Welche Rolle spielt die Funktion RE im aktuellen Vorgehen für dich?
  Welche Aufgaben und Verantwortlichkeiten übernimmt das RE deiner Meinung nach im aktuellen Vorgehen?
  - *Unterstützung Fachbereich,*
  - *Stories schreiben,*
  - *Ansprechpartner für die Entwickler beim Umsetzungsproblemen*
  - *Besprechung Testergebnisse mit den Entwicklern.*
  - *Klärung von Fehlern in der Umsetzung (wurde was vergessen, wurde was falsch umgesetzt, wurde was nicht sauber beschrieben)*
  - *Unstimmigkeiten*

- Erfüllt das RE Team diese Aufgaben und Verantwortlichkeiten zu deiner Zufriedenheit?

  Ja, die machen das gut ;-)

- Welche Verbesserungsvorschläge fallen dir ein?

  Keine

- Ist es deiner Meinung nach notwendig, dass im aktuellen Vorgehen ein RE Team Teil des Vorgehens ist oder könnten die Aufgaben und Funktionen auch von anderen Rollen aus dem Team übernommen werden?

  Ja notwendig, weil die POs nicht alle Stories selber schreiben können. Wir sind ja nah an der Konzeption dran, man braucht dafür mehrere Köpfe. Fehlendes Technische Verständnis auf Seiten der POs wird durch uns übersetzt. RE als Schnittstellenfunktion also. Von den POs könnte das schon übernommen werden. Aber das wäre eher problematisch. Dadurch dass wir uns viel Abstimmen mit den Entwicklern und dem UX Team, nehmen wir den POs schon viel ab.

- Könnte das RE deiner Meinung nach weitere Aufgaben und Verantwortlichkeiten übernehmen um das aktuelle Vorgehen zu unterstützen?

Den Prozess zur Lösungsfindung so zu unterstützen damit nicht ständig um-entschieden wird. Die Entscheidungen sind oft sprunghaft, da könnte das RE besser unterstützen

- Ist das RE im aktuellen Vorgehen hilfreich um in fachlichen Diskussionen zwischen Entwicklerteam und Fachbereich zu vermitteln?

  Ja

Carving

- Wie beurteilst du die Carving-Termine in Bezug auf:

  Häufigkeit

  Die Termine sind schon oft ausgefallen. Entweder weil es nichts zu besprechen gab, oder weil wir stattdessen ein Schätzmeeting machen mussten bei dringenden Themen. Manchmal haben wir dann Stories im Refinement vorgestellt ohne das Thema vorher im Carving besprochen zu haben. Das gab dann viele Diskussionen. Da hätte es schon Sinn gemacht das im Carving davor besprochen zu haben. Zum Beispiel beim Thema „Suche".

  Dauer

  1,5 Stunden sind ok, oft brauchen wir die gar nicht

  Anzahl Teilnehmer

  Zwei Entwickler sind immer dabei. Es wäre wichtig, dass auch die richtigen Entwickler dabei sind. Manchmal waren dann welche mit im Carving die sich bei einem bestimmten Thema nicht so gut auskannten. Da haben wir dann etwas besprochen, und eine Woche später hieß es dann, ne, das können wir nicht so machen. Aber das kam dann erst auf, nachdem die sich im Team dann nochmal abgestimmt hatten. Dann hätten wir das nicht besprechen müssen.
  Wenn wir die Themen nicht vorher bekannt geben dann wissen die Entwickler auch nicht wen Sie schicken sollen. Mir wurde mal gesagt, dass wir für das Carving keine Mail raus schicken müssen mit den Themen die wir besprechen. Eine kurze Mail mit einer Erklärung um was es geht wäre meiner Meinung nach besser. Dass die POs und wir vom RE alle da sind ist sinnvoll. Warum die Daimler IT vertreten ist frage ich mich aber auch manchmal, das ist wohl, weil sie früher die PO Rolle eingenommen haben.

  Art der Durchführung (Methodik)

  weiß ich gerade nichts zu

  Diskussionskultur

  In den Diskussionen wird man öfter unterbrochen und kann seine Sätze nicht zu Ende bringen. Als ich dann einmal etwas am Whiteboard erklären wollte hatte mir keiner mehr zugehört, weil sie einfach weiter diskutiert haben.

  Kreativität der Ideenfindung.

  Nicht so, man könnte mehr visualisieren

- Fließen deine Anmerkungen, Lösungsvorschläge und Kritiken in die Lösungsfindung mit ein?

Kommt darauf an.
Kommt drauf an, mit wem man diskutiert. Bei Diskussionen mit den POs in einer Gruppe habe ich das Gefühl, dass meine Anmerkungen nicht ernst genommen werden.

- Hast du weitere Kritikpunkte zum Carving?

  Wenn wir vor dem Refinement Stories rumschicken oder Stories im Carving besprechen, kommen oftmals keine Fragen zu bestimmten Akzeptanzkriterien, die meiner Meinung nach aber einer Diskussion bedürfen. Die Information steht zwar im Akzeptanzkriterium drin, aber ich denke die Entwickler können das gar nicht verstehen, wenn wir da nicht drüber gesprochen haben. Es macht für mich aber auch keinen Sinn, wenn ich alle Kriterien einzeln vorlesen würde. Lesen können ja alle selbst.

- Hast du Verbesserungsvorschläge für das Carving?

  Diskussionskultur einhalten. Scrum Master sollte dafür sorgen, dass einer nach dem anderen spricht.

Context Map

- Bis vor wenigen Wochen war das RE Team dafür zuständig die Context Map zu erstellen. Aktuell liegt dies in der Verantwortung der Entwickler.
  Welche Vor- und Nachteile haben die jeweiligen Vorgehensweisen?

  Vorteil: wenn wir das machen haben wir dann gleich das Verständnis. Der Vorteil davon, wenn die Entwickler das machen ist, dass die Dinge die mich nicht betreffen ich auch nicht modellieren muss. Wenn ich es selber schreibe verstehe ich es besser. Dann verstehen es aber Entwickler wiederum nicht mehr. Sie müssen es aber umsetzen. Meine Gedanken stehen in der Story. Im aktuellen Vorgehen schreibe ich die Story, dann wird die Context Map von den Entwicklern modelliert. Das finde ich gut so. Allerdings haben wir das neue Vorgehen noch nicht so lange. Da müssen wir erst mal sehen wie sich das in Zukunft entwickelt.

- Nach der Literatur zu DDD entsteht die Context Map in Zusammenarbeit zwischen Domänenexperten und dem Entwicklungsteam.
  Könntest du dir dieses Vorgehen auch für das GSSN+ Projekt vorstellen?

  Weiß ich nicht. Kenn ich eben so nicht.
  In manchen Stories haben wir ja die Tabellen mit den Funktionen drin, die dann auch so in der Context Map stehen bzw. übernommen werden. Wenn wir die dann besprechen und über die Regeln diskutieren, diskutieren wir ja eigentlich schon über die Context Map. Das ist sozusagen die Vorstufe. Durch die Diskussion modellieren wir also gemeinsam die Context Map.

- Werden deiner Meinung nach die fachlichen Zusammenhänge und Abhängigkeiten des Systems GSSN+ in der Context Map klar?

  Ohne die Gliffy Diagramme nicht, denn die Tabellen enthalten nicht alle Informationen der Zusammenhänge der Events/Kommandos. Wenn ich die Diagramme anschaue dann verstehe ich das schon. Gliffy ist aber zu langsam. Änderungen im Gliffy habe ich nicht immer parallel mit den Änderungen in den Tabellen durchgeführt, weil es einfach aufwändig und lästig ist die Gliffy Diagramme anzupassen. Die Diagramme sind aber notwendig um Abhängigkeiten darzustellen.

- Ist die Context Map deiner Meinung nach ausreichend modelliert und beschrieben um als Systemdokumentation den Abnehmersystemen zur Verfügung gestellt zu werden?

Wenn alle Regeln aus den Akzeptanzkriterien drin stehen dann schon. So wie es jetzt ist noch nicht. Da fehlen teilweise Informationen in der Context Map die in den Stories stehen. Wenn tatsächlich alle Informationen drin sind ja. Wenn Context Map auf dem Stand der Implementierung ist dann ja.

User Stories

- Wie beurteilst du die Qualität der User-Stories in Bezug auf:

  Schnitt

  Wir im RE Team stimmen uns dazu ab. Wenn das für die Entwickler so nicht passt, dann können sie es ja zusammenfassen oder uns die Info geben, dann können wir Stories zusammenfassen.

  Lösungsneutralität

  In manchen Fällen macht es Sinn Bezug auf die Context Map zunehmen und konkret ein Lösungsdetail zu beschreiben, wenn ich weiß, dass es einfach nicht anders umgesetzt werden kann oder sollte. Wenn es verschiedene Lösungsmöglichkeiten gibt, dann sollte man nichts vorgeben.

Interview B

Interview zum Thema „Die Rolle des Requirements Engineerings (RE) in Domain-Driven Design (DDD)“

Kennung: B
Datum: 04.05.18
Berufserfahrung: Seit Abschluss des Informatikstudiums 1884 in der Softwareentwicklung tätig

Einleitung

- Was ist deine Rolle im aktuellen Projekt GSSN+?

  Die Projektleitung RE

- Wieviel Spaß macht dir deine Arbeit im Projekt GSSN+ aktuell?

  Zurzeit habe ich ziemlich viel Spaß.

- Wieviel Erfahrung hast du bereits praktisch und/oder theoretisch mit DDD sammeln können?

  Die Theorie habe ich aus Büchern, GSSN+ ist das erste praktische Projekt, das ich mache.

- Wieviel Erfahrung hast du mit Requirements Engineering, sowohl praktisch als auch theoretisch?

  Seit 5 Jahren bin ich im Bereich RE tätig. Die Theorie habe ich im Studium gelernt. Eine Zertifizierung „Agiles RE“ habe ich gemacht.

- Hast du selbst schon einmal Methoden, Techniken oder Vorgehensweisen aus dem RE angewendet?

  Ich habe schon in früheren Projekten für das eigene Verständnis fachliche Datenmodelle erstellt. Das war dann die Grundlage für die Storys, die ich darauf geschrieben habe. Agile Methoden verwende ich, wie z.B.

die User Storys. Aber die Hauptaufgabe war immer, die Übersetzung zu leisten zwischen dem Fachbereich und der Entwicklungsabteilung.

- Sind dir die Projektziele und Iterationsziele des Projekts GSSN+ stets klar?

  Im Groben ja, im Detail gibt es Abstimmungsbedarf. Das Ziel ist die Ablösung von GSSN-Classic. Das neue System soll ein modernes Bedienkonzept haben. Auch das Thema Netzwerkplanung soll mit GSSN+ möglich werden. Ziel ist die Übersetzung der Vision in die Domänen. Das ist die Aufgabe des RE´s.

- Sind die Ziele irgendwo dokumentiert?

  Die stehen im HLD.

Zusammenarbeit im aktuellen Vorgehen

- Wie bewertest du die Zusammenarbeit im ganzen Team oder zwischen den einzelnen Teams?
  Welche Aspekte der Zusammenarbeit gefallen dir gut?
  Welche Aspekte gefallen dir nicht so gut?
  Was könnte man verändern, damit sich die Zusammenarbeit im gesamten Team oder zwischen den einzelnen Funktonen verbessert?

  Die Zusammenarbeit im ganzen Team funktioniert. Es ist aber kein Best Practice. Es ist das Beste, was wir hinbekommen haben, eine Kompromisslösung.
  Die Zusammenarbeit mit dem PO ist auf Domänenebene nicht mehr vorhanden. Der PO lässt sich nicht auf die Domänen ein. Der Wille sich damit auseinanderzusetzen fehlt. Er sollte sich aber auf die Denkweise einlassen. Vielleicht fehlt das Vermögen, abstrakt zu denken. Über die Oberfläche zu diskutieren, fällt ihm einfacher.
  Die Zusammenarbeit zwischen dem RE Team und der TSS (Entwicklung) ist viel besser geworden. Mit Einschränkungen allerdings. Die TSS betrachtet das RE nicht immer als vollwertigen Partner. Die würden gern die Domänen ganz allein bearbeiten.
  Das können wir verbessern, indem wir zeigen, dass wir die Domänen verstehen und da auch mitreden können. Im Allgemeinen ist eine Zusammenarbeit schwierig wegen der aktuellen Gesetzeslage zum Thema Fremdfirmen. Wir, als externe Dienstleister haben es da schwieriger.

DDD

- Was ist für dich DDD? Wie würdest du in ein oder zwei Sätzen erklären, was DDD ist?

  DDD ist die Weiterentwicklung eines fachlichen Datenmodells mit Eigenständigkeit der Domänen.

- Wo siehst du Vorteile der Anwendung von DDD im aktuellen Projekt GSSN+?

  Bei der Dokumentationsfähigkeit der Geschäftsregeln. Beim fachlichen Ansatz für eine Microservice Architektur und der gemeinsamen Sprachebene.

- Achten wir denn auf die Sprache?

  Ja, ich finde schon. Wir vermischen manchmal die Begriffe aus Classic und GSSN+. Aber wir achten schon darauf.

- Wie oft schaust du in unser Glossar?

  Nicht oft genug.

- Wo siehst du Schwächen in der Anwendung von DDD im aktuellen Projekt GSSN+?

  Es verlangt in einem agilen Projekt, dass sich der PO mit dem Thema DDD auseinandersetzt. Der PO muss sich auf die Kontext Map einlassen.
  Die Akzeptanz beim Kunden sollte vorhanden sein, damit das funktioniert. Das Denken in Domänen ist komplizierter als im Objektorientierten.

- Welche Probleme gibt es aktuell in der Anwendung von DDD im aktuellen Projekt?

  Bei der Entwicklung der Domänen und ihrer Abgrenzungen. Das ist zu technisch in der Modellierung. Außerdem fehlt auf Seiten des Fachbereichs die Akzeptanz, mit DDD zu arbeiten.

- Wie könnten diese gelöst werden?

  Zum Thema Akzeptanz: Wir müssten zeigen, dass es funktioniert. So können wir ihn überzeugen. Zum Thema Domänen: Die Domänen müssten anders geschnitten werden. Sie sind zu wenig fachlich und zu feingranular.

- Werden deiner Meinung nach fachliche Themen zwischen Fachbereich und Entwicklern ausreichend diskutiert sodass sich gemeinsames Verständnis entwickeln kann?

  Besprochen ja, diskutiert nein. Die fachlichen Themen zwischen Fachbereich und Entwicklern werden nicht ausreichend diskutiert.
  Es gibt zwei Problematiken: 1. Die TSS müsste ergebnisoffener sein. Die TSS hat oft ein Wunschergebnis und lenkt die Diskussion in diese Richtung.
  2. Sasa lässt sich nicht genug auf die Sprache und Denkweise ein.

- In DDD wird viel Wert gelegt auf die Entwicklung einer gemeinsamen Sprache zwischen Entwicklungsteam und dem Fachbereich. Diese ist eine domänengetriebene Sprache, d.h. es werden vor allem Begriffe aus dem Fachbereich verwendet. Die Begrifflichkeiten sollten klar definiert sein und nicht durch andere Begriffe synonym verwendet werden. Entwickler und Fachbereich sollten die gleichen Begriffe verwenden.
  Wird deiner Meinung nach im Projekt Wert gelegt auf eine gemeinsame Sprache?

  Ja, das tun wir. Irgendwann etabliert sich die Begrifflichkeit dann auch beim PO.

- Sind die Begriffe klar definiert?
  Gibt es manchmal Verständnisprobleme bei der Verwendung bestimmter Begriffe?

  Ja, Probleme haben wir. Wir haben oft auch Schwierigkeiten, die richtige Definition zu finden. Beispiel: Outlet und Business Site.

- Gibt es Diskussionen über eine einheitliche Verwendung von Begriffen?

  Ja, gibt es. Leider. Die Schwierigkeit bei DDD ist: du brauchst Mitarbeiter, die diskussionswillig sind und auch aktiv an einer Diskussion teilnehmen.

- In DDD arbeitet der Domänenexperte direkt mit dem Entwicklerteam zusammen.
  Wer übernimmt deiner Meinung nach im aktuellen Projekt die Rolle des Domänenexperten?

  Der PO ist der Domänenexperte. RE nimmt diese Rolle aber auch ein, weil der PO auf dieser Ebene nicht diskutieren möchte. Das Abstraktionsvermögen fehlt da.

- Hat dieser deiner Meinung nach ausreichendes Fachwissen, um das Projekt voranzubringen?

  Ja, der PO hat ausreichend Fachwissen.

- Welche Rolle spielt die Funktion RE im aktuellen Vorgehen für dich?
  Welche Aufgaben und Verantwortlichkeiten übernimmt das RE deiner Meinung nach im aktuellen Vorgehen?

  Das RE ist verantwortlich für die Erstellung der Storys und für die Vollständigkeit und Qualität des Domänenmodells, verantwortlich für die Qualitätsabsicherung der Umsetzung der Storys. Wir interviewen den PO, um die Anforderungen zu erfassen und für die Entwickler zu übersetzen. Wir besprechen mit dem Entwicklungsteam die Domänen. Wir kontrollieren die Domänen und wir testen.

- Erfüllt das RE Team diese Aufgaben und Verantwortlichkeiten zu deiner Zufriedenheit?

  Ja, sehr

- Welche Verbesserungsvorschläge fallen dir ein?

  Aktuell nicht viel. Verbesserungspotential versuche ich immer sofort umzusetzen. Technisches Verständnis ist wichtig für die Diskussion mit den Entwicklern.

- Ist es deiner Meinung nach notwendig, dass im aktuellen Vorgehen ein RE Team Teil des Vorgehens ist, oder könnten die Aufgaben und Funktionen auch von anderen Rollen aus dem Team übernommen werden?

  Wir sind ja dem Fachbereich angegliedert, also der verlängerte Arm des POs. Wir sind Anwalt des POs und nicht des Entwicklungsteams. Das ist natürlich etwas anderes als für das RE vom Entwicklungsteam geleitet werden. Das Schwierige daran ist, dass wir als weitere Partei in der Abstimmung dazwischenstehen. Der Abstimmungsaufwand steigt.

- Könnte das RE deiner Meinung nach weitere Aufgaben und Verantwortlichkeiten übernehmen, um das aktuelle Vorgehen zu unterstützen?

  Ja, wir könnten eine aktivere Rolle beim Aufbau der Domänen übernehmen.

- Ist das RE im aktuellen Vorgehen hilfreich, um in fachlichen Diskussionen zwischen Entwicklerteam und Fachbereich zu vermitteln?

  Ja, siehe Company und Outlet Beispiel. Es ist notwendig, dass da jemand vermittelt zwischen Fachbereich und Entwicklung.

Carving

- Wie beurteilst du die Carving-Termine in Bezug auf:

  Häufigkeit

  ok

  Dauer

  Die Termine könnten etwas länger dauern, die 1.5 Stunden sind oft zu knapp.

  Anzahl der Teilnehmer

Ausreichend

Art der Durchführung (Methodik)

Grottig. Wir sollten uns den Themen über die Domänen annähern und nicht über die Oberfläche. Erst diskutieren wir über die Oberfläche und dann überlegen wir, wo es in den Domänen umgesetzt werden soll. Sasa hat unterbunden, dass wir über Domänen diskutieren. Das ist nicht gut.

Diskussionskultur

Sie ist familiär, wir kommen zu einem Ergebnis, manchmal würgt der PO aber die Diskussion ab.

Kreativität der Ideenfindung

Wenig. Wir haben eine abgewürgte Diskussionskultur. Die Ursachen liegen in schlechter Erfahrung mit dem Event Storming. Viel Zeit und Geld wurde da verbrannt. Alle Parteien müssen gewillt sein, ergebnisoffen zu diskutieren. Zum Beispiel beim Thema ›Namenszusatz‹. Es macht keinen Sinn, dass wir da zehnmal darüber diskutieren. Über einen Namen muss man nicht so viel diskutieren. Wichtiger ist, dass die Strukturen passen. Die falschen Sachen werden diskutiert. Den Entwicklern fehlt die Sinnhaftigkeit, sagen sie. Da herrscht ein falsches Rollenverständnis. Als Entwickler muss ich nicht immer den Sinn verstehen, es ist wichtiger, die Bedeutung zu verstehen. Was soll passieren, nicht warum. Das steht dem Domänen-Gedanken entgegen. Erklärungen des Domänenexperten sollten den Entwicklern ausreichen, Beweise sollten nicht eingefordert werden. Der fragt sich dann, ob da nicht das Vertrauen auf Seiten der Entwickler fehlt.

- Fließen deine Anmerkungen, Lösungsvorschläge und Kritiken in die Lösungsfindung mit ein?

  Ja, ich kann mich nicht beklagen. Im Großen und Ganzen ja.

- Hast du weitere Kritikpunkte zum Carving?
  Hast du Verbesserungsvorschläge für das Carving?

Context Map

- Bis vor wenigen Wochen war das RE Team dafür zuständig, die Context Map zu erstellen. Aktuell liegt dies in der Verantwortung der Entwickler.
  Welche Vor- und Nachteile haben die jeweiligen Vorgehensweisen?

  Früher: Nachteil: Der hohe Abstimmungsaufwand im Vorfeld mit der Entwicklung. Außerdem mussten die Entwickler Freiheitsgrade der Lösungsfindung schon in der Abstimmungsphase abgeben, als wir die Domänen definiert haben. Manchmal musste die Context Map auch nachdokumentiert werden, weil die Entwickler dann etwas Anderes umgesetzt hatten. Vorteil: Schon vor der Entwicklung schon hatten wir ein gemeinsames Verständnis. Der aktuelle Stand innerhalb der Domäne (Context Map) war uns immer bekannt. Wir konnten die Erstellung der Storys auf die Domänen abstützen.
  Heute: Nachteil: Wir haben einen erhöhten Aufwand, ein nachträgliches Verständnis zu schaffen. Außerdem haben wir keinen Termin, um dann die von den Entwicklern erstellte Context Map zu besprechen. Es ist nicht transparent, wie die Storys sich dann auf die Context Map auswirken. Alle neuen Storys setzen ja auf die Domänen auf, deswegen brauchen wir ja das Verständnis.

- Nach der Literatur zu DDD entsteht die Context Map in Zusammenarbeit zwischen Domänenexperten und dem Entwicklungsteam.
  Könntest du dir dieses Vorgehen auch für das GSSN+ Projekt vorstellen?

Vorstellen ja, bei anderer personeller Besetzung. Beim Fachbereich fehlt einfach die Bereitschaft, sich auf die Domänen einzulassen.

- Werden deiner Meinung nach die fachlichen Zusammenhänge und Abhängigkeiten des Systems GSSN+ über die Context Map klar?

  Ja, großer Vorteil. Genau das wird klar.

- Ist die Context Map deiner Meinung nach ausreichend modelliert und beschrieben, um als Systemdokumentation den Abnehmersystemen zur Verfügung gestellt zu werden?

  Aktuell ja, es kommt aber auf das neue Vorgehen an, wie die Rückkopplung der Story-Inhalte auf die Context Map funktioniert. Das wird man dann sehen.

- Findest du das aktuell eingesetzte Tool (Confluence) geeignet, um die Context Map zu modellieren?

  Nur bedingt.

User Stories

- Wie beurteilst du die Qualität der User-Stories in Bezug auf:

  Schnitt

  abgestimmt und gut

  Lösungsneutralität

  Verständlichkeit

  schwankend

  Abgestimmtheit

  Stil

- Habe ich ein Thema nicht abgefragt, das aber noch wichtig wäre?

  Du könntest fragen, wie die anderen das Event Storming erlebt haben. Fanden sie es gut?
  Wie schaffe ich es, die Oberflächen aus Benutzerperspektive mit den Domänen abzugleichen?
  Wir haben aktuell ein UX getriebenes Vorgehen anstatt eines domänen-getriebenen Vorgehens.
  Beispiel: Das Sprachprofil existiert in den Domänen nicht. Wie bekommen wir das in Einklang? Auch von der Sprache her.

Interview C

Interview zum Thema „Die Rolle des Requirements Engineerings (RE) in Domain-Driven Design (DDD)“

Kennung: C
Datum: 05.05.2018
Berufserfahrung: 12 Jahre Software Engineering, Technical Lead

Einleitung

- Was ist deine Rolle im aktuellen Projekt GSSN+?

  Ich bin Software Engineer

- Wieviel Spaß macht dir deine Arbeit im Projekt GSSN+ aktuell?

  Mittlerweile bin ich da schmerzfrei, kann entspannen und Spaß haben. Die Dinge laufen wie sie laufen, man könnte sich aufregen, aber das bringt nichts.
  Es macht Spaß. Das hängt ja auch von vielen Faktoren ab: Nicht nur vom Projekt, sondern z.B. auch von den die Kollegen, mit denen man zusammen- arbeitet.

- Wieviel Erfahrung hast du bereits praktisch und/oder theoretisch mit DDD sammeln können?

  Ich hatte davon gehört, hatte aber keine Erfahrung vor dem Projekt. Literatur wurde herumgereicht. Jemand hat das moderiert und gecoacht.

- Wieviel Erfahrung hast du mit Requirements Engineering, sowohl praktisch als auch theoretisch?

  Was versteht man überhaupt darunter?
  Anforderungen an ein System in fachlicher Hinsicht zu analysieren, zu dokumentieren und diese transparent zu machen.
  Mir fehlt in GSSN+ der Gesamtüberblick. Ich kenne zwar GSSN-Classic schon, habe aber keine Ahnung, welche Use Cases in GSSN+ umgesetzt werden sollen. Die Top-Five Features fehlen. Meiner Meinung sollte sich die Vision in den Anwendungsszenarien wiederfinden, die dann wieder detaillierter in Use Cases spezifiziert werden. Dieses Vorgehen sowohl für die fachliche als auch die technische Sicht. Man sollte sich fragen »Was will ich denn?« und »Was brauche ich?«
  Von der Planung her wäre das hilfreich. Die Themes fehlen. Daraus lassen sich dann die Epics erstellen, die man wiederum in die Storys aufbricht. Als Entwickler hast du den Anspruch, dir über die fachlichen Probleme Gedanken zu machen und Lösungen zu finden. Die Technik ist nur Mittel zum Zweck. Du brauchst aber das Problem, sonst bist du nur der Weisungsempfänger. Man hat uns oft gesagt »Ihr kennt das System doch?«. Aber nur ein kleiner Teil von uns kennt das alte System.

- Wurde die Vision mal besprochen?

  Ja, die wurde uns vorgestellt.

- Zurück zu deiner Erfahrung

  Im klassischen Umfeld habe ich viel RE Arbeit gemacht: Mit Kunden Anforderungen klamüsert, durchdacht, Nachdokumentationen aufgebaut, aus der Anforderungsdokumentation Anwendungs-Szenarien und Use Cases beschrieben.
  Die Release Spezifikation schmeiße ich weg (Sprintdokumentation), die Systemspezifikation sollte aber nachhaltig sein.

- Sind dir die Projektziele und Iterationsziele des Projekts GSSN+ stets klar?

  Die Iterationen sind einmal festgelegt worden, mir ist aber nicht klar, ob das noch aktuell ist. Die Ziele sind zu schwammig definiert.
  Ist dir das wichtig? Ja, wenn man in Iterationen plant, ist es sinnvoll. Eigentlich ist das ein Reporting-Thema.

Aus Entwicklersicht bräuchte ich die Iterationsziele aber nicht, wenn wir tatsächlich agil wären. »As soon as possible«, eins nach dem anderen. Da braucht es keine Meilensteine. Aber wir sind nicht agil. Alle tun nur so, als wäre es ein agiles Projekt.
Man muss sich fragen, ob Agilität überhaupt bei der Auftragsfertigung (Projektgeschäft) passend ist? Mit GSSN+ verdient man kein Geld. Da schauen sie natürlich auf Zeit und Budget. Anders ist es, wenn eine Firma selbst ein Produkt entwickelt und damit Geld verdient. Das wird dann so weiterentwickelt, wie es eben erforderlich ist. Da passt Agilität besser.

Zusammenarbeit im aktuellen Vorgehen

- Wie bewertest du die Zusammenarbeit im ganzen Team oder zwischen den einzelnen Teams?
  Welche Aspekte der Zusammenarbeit gefallen dir gut?
  Welche Aspekte gefallen dir nicht so gut?
  Was könnte man verändern, damit sich die Zusammenarbeit im gesamten Team oder zwischen den einzelnen Funktonen verbessert?

  Gut: Mit dem Carving versuchen wir, uns über fachliche Themen Gedanken machen. Besser wäre aber, eine Ebene grober auf Epic Ebene zu diskutieren. Zum Beispiel zum Thema »Reporting« oder »User Scoping«. Das wäre die richtige Granularität. Um das umsetzen zu können, muss ich die Themen kennen. Nicht gut: Der Faden durch das System fehlt. Wir machen uns z.B. im letzten Carving über die Migration Gedanken, obwohl die Funktionalitäten, um die Migrationsthemen umsetzen zu können, fehlen. Die Grundvoraussetzung fehlt. Damit wird dann das Team belastet. Die Reihenfolge der Themen fehlt. Nur Themen sollten im Carving besprochen werden, die auch Thema des nächsten Sprints sein werden. Da wäre eine bessere Absprache gut. Der Backlog ist z.B. nicht richtig priorisiert. Dem Kunden ist das egal, Hauptsache, der Sprint ist voll. Der Faden fehlt. Wie ist mein Reiseplan?
  Wir schätzen häufig Storys, die in den nächsten Sprints keine Themen sind. Das Team wird aus dem Kontext gerissen, und wenn es soweit ist, hat man das Thema wieder vergessen.
  Die Zusammenarbeit funktioniert an sich gut. Nicht klar ist mir manchmal, mit wem ich eigentlich reden soll? Manchmal ist es nicht ganz klar, an wen ich die fachliche Frage richten soll. Intuitiv an den PO, aber dann eigentlich RE, weil diese die Story schreiben. Wer ist Ansprechpartner? PO setzt sich eh nicht mit den Storys auseinander.

- Ist eine direkte Kommunikation mit dem Fachbereich möglich?

  Sie ist unproblematisch.

DDD

- Was ist für dich DDD?
  Wie würdest du in ein oder zwei Sätzen erklären, was DDD ist?

  In der ersten Instanz ist es eine Methode zur Analyse der fachlichen Problemstellung für Transparenz und einem gemeinsamen fachlichen Verständnis. In GSSN+: ich möchte die Stammdaten der Händler verwalten.

- Machen wir das so aktuell?

  Nein.

- Wo siehst du Vorteile der Anwendung von DDD im aktuellen Projekt GSSN+?

Der Wunsch wäre gewesen, GSSN+ neu zu denken. Auf die Problemebene zu gehen, dem Problem-Space und das System neu zu durchdenken und neu aufzubauen. Die Tiefe ist egal. Es war zum Scheitern verurteilt. Zuviel Betriebsblindheit. Von den gleichen Leuten, die das Alt-System betreuen und betreiben, kannst du GSSN+ nicht neu bauen lassen. Das war eine Fehleinschätzung. Geeigneter wäre es gewesen, das System von nicht vorbelasteten Leuten bauen lassen. Für bestimmte Fälle hätten dann Experten hinzugezogen werden können. Sasa ist nicht der Domänenexperte. Der weiß, wie es momentan läuft. Aber er ist nicht der Experte für die Domäne. Der Domänenexperte darf nicht PO sein. Das System nähert sich immer mehr Classic an. Ein Beispiel ist, dass die Business Site zum Outlet wird. Wenn man es neu denken will, dürfen nicht die gleichen Leute daran arbeiten. Man landet immer bei der gleichen Lösung.

- Wo siehst du Schwächen in der Anwendung von DDD im aktuellen Projekt GSSN+?
  Welche Probleme gibt es aktuell in der Anwendung von DDD im aktuellen Projekt?

  Man bräuchte Leute, die die Methodik verstanden haben. Die wissen, wie sie technisch funktioniert in der Umsetzung, und die das auch wertschätzen. Das sollte nicht von Extern motiviert sein. Alle zusammen in eine Schulung packen und dann entscheiden ob man das möchte oder nicht. Das „Commitment" von allen ist nötig. Dafür müssen es aber alle verstanden haben. Und das nicht während der Projektlaufzeit, sondern schon im Vorprojekt. Bis heute sind einige nicht motiviert. Die Context Map wird z.B. nicht von allen Beteiligten genutzt

- Wie könnten diese Probleme gelöst werden?

  Eine große Seite an der Wand mit allen Aggregaten und Commandos. Im Tool geht das nicht. Auf einer 10 Meter Wand kann man einen Überblick bekommen.
  Wir haben uns nicht über den Inhalt unterhalten. Wir waren da eine Ebene drüber. Erst mal sollte man die Domänen-Ebene und die Bounded Context klarkriegen. Und dann geht man rein. Sobald ich ein Aggregat erstellt habe, bin ich schon bei der Lösung.
  Die Domänen ergeben sich aus den Themes/Epics; wenn man klassisch unterwegs ist, aus den Use Cases und Anwendungsfällen. Die würden dann in den Domänen gebündelt werden. Wir diskutieren aber auf unterster Ebene.

- Werden deiner Meinung nach fachliche Themen zwischen Fachbereich und Entwicklern ausreichend diskutiert sodass sich gemeinsames Verständnis entwickeln kann?

  Inzwischen haben wir das relativ gut im Griff. Durch die Carvings. Die Themen drehen so Runden, und es wird ein Feedback generiert. Aber die Carving-Ebene sollte einen Schritt zurück auf die Epic Ebene.

- In DDD wird viel Wert gelegt auf die Entwicklung einer gemeinsamen Sprache zwischen Entwicklungsteam und dem Fachbereich. Sie ist eine domänengetriebene Sprache, d.h. es werden vor allem Begriffe aus dem Fachbereich verwendet. Die Begrifflichkeiten sollten klar definiert sein und nicht durch andere Begriffe synonym verwendet werden. Entwickler und Fachbereich sollten die gleichen Begriffe verwenden.
  Wird deiner Meinung nach im Projekt Wert gelegt auf eine gemeinsame Sprache?

  Nee, man tut sich schwer, die Fachlichkeit auf der Sprachebene zu vermitteln. Beispiel »Business Site› und »Outlet«. Der Kunde spricht nur von Outlet. Outlet ist aber nur eine View. Ein weiteres Beispiel ist der »Namenszusatz«. Die Anforderung kommt nicht aus einer fachlichen Domäne heraus. Wir brauchen den nur für die Migration.

- Sollte RE da mehr tun?

  RE sollte Struktur reinbekommen, den Kunden stressen. Fragen »Warum brauche ich das?«, dem Kunden richtig auf den Sack gehen. Strukturierung geht immer, das Hinterfragen sollte aber nur bis zu einem gewissen Punkt gehen.

- Sind die Begriffe klar definiert?
  Gibt es manchmal Verständnisprobleme bei der Verwendung bestimmter Begriffe?
  Gibt es Diskussionen über eine einheitliche Verwendung von Begriffen?
  In DDD arbeitet der Domänenexperte direkt mit dem Entwicklerteam zusammen.
  Wer übernimmt deiner Meinung nach im aktuellen Projekt die Rolle des Domänenexperten?

  Eigentlich haben wir keinen, wir haben Lösungsexperten. Wer müsste das sein? Leute aus dem MPC+ Rechtsexperten. Die Anwender sind mit dem Business vertraut. Sie kennen sich mit Rechtsstrukturen oder Rechtsformen aus. Sie wissen, wie die Betriebe aufgebaut sind, z.B. Haupt- und Nebensitz. Ist der PO der Experte dafür?
  Wenn wir die Daimlerwelt abbilden wollen, dann ist der PO vielleicht schon der Experte.
  Das Domänenwissen wird aber immer von mehreren kommen. Außer in einem abgeschossenen Bereich, wie zum Beispiel beim Online-Banking. Domäne ist Stammdatenverwaltung. Ist der Sasa dafür Experte? Gibt es dafür einen Domänenexperten? Daten verwalten ist generisch.

- Ich habe mal gelesen, dass DDD nur angewandt werden soll, wenn die Domänen-Logik komplex ist. Ist das in GSSN+ gegeben?

  In GSSN+ werden die Daten der Händler bereitgestellt. Nur die Daten bereitzustellen, ist sehr einfach. Auch die Zuweisung von Verträgen zu den Händlern ist sehr einfach.
  Mit dem Regelwerk-Daimler wird es dann aber komplex. Stichwort: Berechtigungen, Service und Dienstleistungen und Produkte. Services kann der Händler nur anbieten, wenn dies und das gegeben ist. Das ist das ausgedachte Regelwerk-Daimler. Keiner fragt was man damit eigentlich lösen will.

- Hat dieser/diese deiner Meinung nach ausreichendes Domänenwissen, um das Projekt voranzubringen?
  Findet eine direkte Zusammenarbeit zwischen Domänenexperten und Entwicklungsteam statt?

  Wenn der PO ein Domänenexperte ist, dann schon.

RE

- Welche Rolle spielt die Funktion RE im aktuellen Vorgehen für dich?

  Story-Klopfer.

- Welche Aufgaben und Verantwortlichkeiten übernimmt das RE deiner Meinung nach im aktuellen Vorgehen?

  Mirka und Philipp sind Story-Klopfer. Jörg redet auch fachlich mit.

- Erfüllt das RE Team diese Aufgaben und Verantwortlichkeiten zu deiner Zufriedenheit?

Zu UX: Mit der Usability passiert nicht viel. Der Weg im Projekt wird über die Oberfläche gegangen. Die UX Zentriertheit ist ein Problem.
Zuerst sollte die Fachlichkeit geklärt werden und dann die UI. Sasa muss sich das vorstellen können. Er kann nicht in Abläufen denken.
Ein roter Faden wäre wünschenswert, ein Kompass durch das System.

- Welche Verbesserungsvorschläge fallen dir ein?
  Ist es deiner Meinung nach notwendig, dass im aktuellen Vorgehen ein RE Team Teil des Vorgehens ist, oder könnten die Aufgaben und Funktionen auch von anderen Rollen aus dem Team übernommen werden?
  Könnte das RE deiner Meinung nach weitere Aufgaben und Verantwortlichkeiten übernehmen, um das aktuelle Vorgehen zu unterstützen?
  Ist das RE im aktuellen Vorgehen hilfreich, um in fachlichen Diskussionen zwischen Entwicklerteam und Fachbereich zu vermitteln?
  Werden Anforderungen verständlich und ausreichend dokumentiert?

Interview D

Interview zum Thema „Die Rolle des Requirements Engineerings (RE) in Domain-Driven Design (DDD)“

Kennung: D
Berufserfahrung: 8 Jahre als Software Engineer

Einleitung

- Was ist deine Rolle im aktuellen Projekt GSSN+?

  Software Engineer

- Wieviel Spaß macht dir deine Arbeit im Projekt GSSN+ aktuell?

  Sehr viel Spaß

- Wie viel Erfahrung hast du bereits praktisch und/oder theoretisch mit DDD sammeln können?

  Im Audit Tool Projekt haben wir DDD ausprobiert, das ging einige Jahre. Dazu kommt meine praktische Erfahrung in GSSN+. Da lerne ich viel »On the Job«. Da bekomme ich einiges mit. Außerdem habe ich ein Buch gelesen und an einem Workshop auf einer Software Konferenz teilgenommen.

- Wieviel Erfahrung hast du mit Requirements Engineering, sowohl praktisch als auch theoretisch?

  Teilweise erstellen wir die Storys in GSSN+.

- Sind dir die Projektziele und Iterationsziele des Projekts GSSN+ stets klar?

  Nein. Die Iterationsziele sind nicht klar. Aber in der Story Map kann ich das nachschauen. Darin stehen die Iterationsziele. Das Projektziel ist klar: Die Ablösung von Classic und der Lifegang.

Zusammenarbeit im aktuellen Vorgehen

- Wie bewertest du die Zusammenarbeit im gesamten Team oder zwischen den einzelnen Teams?
  Welche Aspekte der Zusammenarbeit gefallen dir gut?
  Welche Aspekte gefallen dir nicht so gut?
  Was könnte man verändern, damit sich die Zusammenarbeit im gesamten Team oder zwischen den einzelnen Funktonen verbessert?

  Ich finde gut, dass man die Carving Termine verbessern will. Wir sollen ja jetzt die Storys zusammen durchgehen und schauen, dass sie dann für das Refinement fertig sind. So bekommt dann das RE Team mit, wie wir die Storys haben wollen. Die sind oft nicht so toll geschrieben. Da steht oft viel drin, was man an Information gar nicht braucht. In der Vergangenheit war das leider oft so, dass wir die Liste der Storys für das Refinement erst freitags bekommen haben, da bleibt dann keine Zeit, die Storys mit dem RE Team zu besprechen. Montags ist das RE Team ja auch nicht in Ulm. Positiv ist, dass wir jetzt über die Storys sprechen in den Carving Terminen mit nur wenigen Entwicklern. Allerdings waren die Carving Termine früher in der großen Runde, in denen wir die Requirements zusammen besprochen haben, auch wichtig. Da waren Entwickler, Fachbereich und RE Team zusammen. Viele Entwickler entdecken eben mehr als nur wenige. Außerdem hat man dann die Fachlichkeit mitbekommen. Ich bin zwar nicht so der Experte, aber ich will dabei sein, wenn die Fachlichkeit besprochen ist. Ich muss die Fachlichkeit zwar nicht selbst bearbeiten, aber verstehen will ich es.
  Es ist schade, dass man das Gefühl hat, dass das Team im früheren Carving zu groß war. Es wäre gut, wenn wir Zeit dafür hätten. Es kostet wohl zu viel, wenn da alle dabei sind. Das kann ich schon verstehen. Wir haben eben oft über Themen gesprochen und uns dabei im Kreis gedreht. Da will man keine Zeit mehr verschwenden. Allerdings zeigen manche fachlichen Diskussionen auch, dass man zu dem besprochenen Thema noch nicht die ideale Lösung gefunden hat.

- Könntest du dir vorstellen, dass wir ein Event Storming für das Carving anwenden?

  Ja, ich schon. Es ist aber schon schiefgelaufen.

DDD

- Was ist für dich DDD? Wie würdest du in ein oder zwei Sätzen erklären was DDD ist?

  Die Fachlichkeit steht im Vordergrund, es gibt Bounded Contexte. Entwickler und Domänenexperten sprechen die gleiche Sprache.

- Leben wir das so im Projekt GSSN+?

  In den Carvings hat man das Gefühl, wir möchten die Fachlichkeit erarbeiten. Sonst nicht unbedingt. Viele Entwickler sind deswegen frustriert. Andere sind nicht so perfektionistisch.
  Zum Beispiel zum Thema Namenszusatz. Manche denken die Inhalte des Feldes sollten einfach übernommen und nach GSSN+ migriert werden. Andere setzen dem entgegen, dass wir so nur Datenmüll übernehmen, weil dort im Feld ja alles Mögliche abgespeichert wird und man sich so keine Mühe macht die Fachlichkeit umzusetzen. Ich sehe es nicht so dogmatisch. Es ist besser, etwas mehr Fachlichkeit hinein zu bekommen, keinen neuen Datenmüll.

- Wo siehst du Vorteile bei der Anwendung von DDD im aktuellen Projekt GSSN+?

Event Sourcing finde ich gut, also wie die Software entwickelt wird. Kleine Bounded Contexte und kleine Aggregate finde ich gut, mit zu großen ist es schwierig. Mit kleinen Aufteilungen gibt es nicht mehr so viel Exceptions die aufgrund der Abhängigkeiten durch die großen Aggregate entstanden waren, und wir haben eine lose Kopplung und somit weniger Abhängigkeiten. Wir haben durch das Event Sourcing einen sehr übersichtlichen Source Code und somit wenig Maintenance. Aufwand und Refactoring ist relativ einfach möglich. Außerdem können wir den aktuellen Zustand des Systems jederzeit aus den Events wiederherstellen.

- Welche Probleme gibt es derzeit in der Anwendung von DDD im aktuellen Projekt?
  Wie könnten diese gelöst werden?

  Die gemeinsame Sprache könnte besser sein. Beiden Seiten bzw. drei Seiten sollte es wichtig sein, dass man vom Gleichen spricht. Wir Entwickler sollten aber nicht so dogmatisch sein und die Fachlichkeit vorgeben wollen bei der Begriffsdefinition. Auf allen Seiten sollte aber das Bewusstsein für eine gemeinsame Sprache vorhanden sein, das fehlt immer mal wieder.

- Werden deiner Meinung nach fachliche Themen zwischen Fachbereich und Entwicklern ausreichend diskutiert sodass sich ein gemeinsames Verständnis entwickeln kann?

  Es ist schade, dass beim Carving nicht immer alle dabei sein können. Ein gemeinsames Event Storming wäre gut. Das darf aber nicht ziellos sein, sondern nur für ein neues Thema, das gerade aktuell ist.

- In DDD arbeitet der Domänenexperte direkt mit dem Entwicklerteam zusammen.
  Wer übernimmt deiner Meinung nach im aktuellen Projekt die Rolle des Domänenexperten?

  Eigentlich der Sasa, vielleicht auch Jörg. Wir als Software Engineers nicht.

- Hat dieser/diese deiner Meinung nach ausreichendes Domänenwissen, um das Projekt voranzubringen?

  Ja

- Findet eine direkte Zusammenarbeit zwischen Domänenexperten und Entwicklungsteam statt?

  Ja. Über die Carvings. Auch sonst. Auch außerhalb der Carvings gibt es immer wieder Gespräche, die man nicht unbedingt alle mitbekommt. Noch besser wäre, wenn das RE Team und der PO besser spontan erreichbar wären. Dazu müssten sie physisch anwesend sein, was eben oft nicht der Fall ist.

RE

- Welche Rolle spielt die Funktion RE im aktuellen Vorgehen für dich?
  Welche Aufgaben und Verantwortlichkeiten übernimmt das RE deiner Meinung nach im aktuellen Vorgehen?

  Sie müssen die Fachlichkeit gut kennen und wissen, was umgesetzt werden soll und dies in die Storys gießen. Die Storys könnten besser sein.

- Welche Verbesserungsvorschläge fallen dir ein?

  Ist es deiner Meinung nach notwendig, dass im aktuellen Vorgehen ein RE Team Teil des Vorgehens ist oder könnten die Aufgaben und Funktionen auch von anderen Rollen aus dem Team übernommen werden?

Für die Entwicklung wäre das zu viel. Es ist sinnvoll, dass es das RE Team gibt. Es wäre schwierig, wenn wir die Storys auch noch erstellen würden. Es ist hilfreich, wenn das vom RE Team gemacht wird. Auf der anderen Seite sind wir zu viele einzelne Teams. Da gibt es viel Abstimmungsbedarf.

- Könnte das RE deiner Meinung nach weitere Aufgaben und Verantwortlichkeiten übernehmen, um das aktuelle Vorgehen zu unterstützen?

  Eventuell das Event Storming.

- Ist das RE im aktuellen Vorgehen hilfreich, um in fachlichen Diskussionen zwischen Entwicklerteam und Fachbereich zu vermitteln?

  Ja, Jörg hilft zu vermitteln, weil er schon so lange in GSSN gearbeitet hat.

- Werden Anforderungen sind verständlich und ausreichend dokumentiert?

  Mir reichen die Storys.

Carving

- Wie beurteilst du die Carving-Termine in Bezug auf:

  Häufigkeit

  Einmal pro Woche reicht aus.

  Dauer

  o.k.

  Anzahl Teilnehmer

  Art der Durchführung (Methodik)

  Wenn es um die Requirements geht, müssen keine Storys besprochen werden. Die fachliche Diskussion ist wichtiger.

  Diskussionskultur

  Es ist gut, dass man sich traut; niemand sollte denken, dass er es besser weiß als andere. Alle Seiten haben etwas beizutragen.

- Fließen deine Anmerkungen, Lösungsvorschläge und Kritiken in die Lösungsfindung mit ein?

  Kommt darauf an.
  Ja

Context Map

- Bis vor wenigen Wochen war das RE Team dafür zuständig, die Context Map zu erstellen. Aktuell liegt dies in der Verantwortung der Entwickler.
  Welche Vor- und Nachteile haben die jeweiligen Vorgehensweisen?

  Es ist besser, wenn wir das machen. Erster Grund: Wir sind freier darin, die Domäne zu gestalten. Es ist sinnvoll, um das zu dokumentieren, was wir gemacht haben. Früher war nicht klar, ob es die Anforderung

war, die dort dokumentiert war oder die Dokumentation von dem, was schon umgesetzt wurde. Es ist sinnvoll. Ich weiß nicht, ob es den Abnehmer interessiert was RE da immer sehr genau beschrieben hat. Meiner Meinung war die Context Map zu genau formuliert.

- Nach der Literatur zu DDD entsteht die Context Map in Zusammenarbeit zwischen Domänenexperten und dem Entwicklungsteam.
  Könntest du dir dieses Vorgehen auch für das GSSN+ Projekt vorstellen?

  Ich denke, das wäre sinnvoll. So kann man gemeinsam über die Fachlichkeit diskutieren.

- Werden deiner Meinung nach die fachlichen Zusammenhänge und Abhängigkeiten des Systems GSSN+ über die Context Map klar?

  Ich weiß nicht. Ich beziehe die Informationen aus anderen Quellen.

- Welchen Quellen?

  Ich bekomme es irgendwie mit: Aus Terminen und Gesprächen, aus Pullrequests bzw. aus dem Source Code. Ich bin nicht so der Doku-Lese-Typ.

- Ist die Context Map deiner Meinung nach ausreichend modelliert und beschrieben, um als Systemdokumentation den Abnehmersystemen zur Verfügung gestellt zu werden?

  Als Abnehmersystem würde ich mir zuerst die Swagger UI anschauen. Die Context Map ist sehr detailliert. Dass ich die Länge der Felder ablesen kann, ist o.k. Wenn man gezielt etwas sucht, können die Übersichten mit den ganzen Events interessant sein und Zusammenhänge aufzeigen. In meiner täglichen Arbeit brauche ich die Übersichten mit den Events nicht, allerdings könnten sie für Abnehmersysteme ggf. hilfreich sein. Ich finde es sinnvoll und wichtig, dass die Fachlichkeit in der Kontext Map dokumentiert ist.

User Storys

- Wie beurteilst du die Qualität der User-Storys in Bezug auf:

  Schnitt

  In letzter Zeit sind die Storys öfter in Front- und Backend aufgeteilt. Es wäre besser, wenn das zusammen in einer Story beschrieben und nach Fachlichkeit aufgeteilt wäre.
  POST und GET sollten immer zusammen in der Story beschrieben sein, weil sonst die Testbarkeit nicht gegeben ist; wenn kein GET dabei ist, macht das keinen Sinn. Es macht keinen Sinn, Länder anlegen zu können, wenn man sie nicht abfragen kann.

  Lösungsneutralität

  Gut wäre es, wenn dort beschrieben wäre »Was« passieren soll und weniger »Wie«. Auf jeden Fall, solange die Lösung noch offen ist.

  Verständlichkeit

  Oft stehen zu viele Infos in den Storys, die nicht gebraucht werden.

- Warst du bei den Event Storming Terminen dabei?

  Nein.

Interview E

Interview zum Thema „Die Rolle des Requirements Engineerings (RE) in Domain-Driven Design (DDD)“

Kennung: E
Berufserfahrung: 10 Jahre Software Engineer

Einleitung

- Was ist deine Rolle im aktuellen Projekt GSSN+?

  Entwickler-Technical Lead

- Wieviel Spaß macht dir deine Arbeit im Projekt GSSN+ aktuell?

  Phasenweise gemischt. Mal sehr viel, mal weniger.

- Wie viel Erfahrung hast du bereits praktisch und/oder theoretisch mit DDD sammeln können?

  Ich habe zwei Bücher gelesen, 2 Konferenzen besucht und eine Schulung gehabt. Praktisch habe ich vor GSSN+ bei ForRetail mit DDD herum-experimentiert. Da haben wir viele Anfängerfehler gemacht. Wir haben vor allem die Patterns angewendet und die Bounded Contexts nur stiefmütterlich betrachtet. Die Bounded Contexts gehören aber in den Vordergrund, und dann kommen die Pattern.

- Wieviel Erfahrung hast du mit Requirements Engineering, sowohl praktisch als auch theoretisch?

  Theorie zu RE habe ich an der Uni gelernt. Ich habe auch mal ein Lasten- und Pflichtenheft geschrieben. Für die Spezifikation in Classic hatten wir einen RE-ler von der TSS. Da hatte ich dann auch mit der Spezifikation zu tun.

- Sind dir die Projektziele und Iterationsziele des Projekts GSSN+ stets klar?

  Grob sind mir die Iterationsziele klar. Es wäre aber wichtig, dass diese klar sind. Warum? Weil wir öfters diskutieren, ob ein Thema jetzt für Iteration 1 oder 2 ist. In der Vergangenheit war das oft nicht klar. Die Frage ist auch, wie perfekt die UI sein muss. Das ist mir noch unklar. In der Story Map, die auch die Iterationszeile enthält, ist nur das Fachliche abgebildet. Gehört die UI da auch mit rein?
  Das Projektziel ist mir klar. Ein klarer roter Faden fehlt aber. Ich kann es aber gar nicht an bestimmten Dingen festmachen. Es ist mein Bauchgefühl.

Zusammenarbeit im aktuellen Vorgehen

- Wie bewertest du die Zusammenarbeit im ganzen Team oder zwischen den einzelnen Teams?
  Welche Aspekte der Zusammenarbeit gefallen dir gut?
  Welche Aspekte gefallen dir nicht so gut?
  Was könnte man verändern damit sich die Zusammenarbeit im gesamten Team oder zwischen den einzelnen Funktonen sich verbessert?

  Mein Wunsch: Wir bekommen die Anforderungen auf Epic-Ebene und diskutieren dann die Fachlichkeit gemeinsam mit Event Storming. Den Mehrwert, der so ein Vorgehen erzielt, konnten wir noch nicht ermitteln. Es hat aber einen großen Mehrwert, sowohl für das RE als auch den Fachbereich, nicht nur für die Entwicklung. Wenn die Fachlichkeit da ist, kann man auch die Storys richtig schneiden. Über die Epics definieren wir

dann die groben Storys, bei den konkreten AKs müssen wir Entwickler nicht unbedingt dabei sein.
Gut finde ich, was wir aktuell in den Carvings versuchen. Die Entwickler schauen nochmal über die Storys. So können wir vermeiden, dass eine bestimmte Lösung in den Storys vorgegeben wird. Mir würden reichen, dass in der Story das Problem beschrieben steht, also grob, was passieren soll. Das würde mir persönlich reichen.
Ich sehe in DDD einen riesen Vorteil. Mir gefällt nicht, dass wir von der Oberfläche her diskutieren. Das kann man sich so halt besser vorstellen über die UI, aber es müsste erst das Modell und dann die UI diskutiert werden. Anders führt es nur zur Verwirrungen. Wie zum Beispiel mit »Tag« und »Label«. Da diskutieren wir dann über das Oberflächenelement. Leider haben wir es nicht geschafft, die Sinnhaftigkeit von DDD zu vermitteln und zu motivieren.

DDD

- Was ist für dich DDD? Wie würdest du in ein oder zwei Sätzen erklären was DDD ist?

  Ein Modellierungsansatz bei dem die Fachlichkeit im Mittelpunkt steht.

- Wo siehst du Vorteile der Anwendung von DDD im aktuellen Projekt GSSN+?

  Die Anwendung von Event Storming wäre ein Vorteil. DDD ist eine gute Sache, um komplexe Sachverhalte in kleine Portionen zu schneiden. Man hat abgeschlossene Domänen, die in Verbindung sind. Man hat eine lose Verbindung. Die Patterns sind nett, aber der Hauptvorteil ist, dass man GSSN in kleine Pakete unterteilen kann. Man hat eine lose Kopplung. Die Kernfunktionalität lassen sich von Randthemen trennen. Das hilft dem Verständnis, wenn man ein loses gekoppeltes System baut. So kommen wir nicht auf die Classic Schiene bei der alles miteinander verzahnt ist.

- Macht ihr eigentlich Context Mapping?

  Machen wir.

- Wo siehst du Schwächen in der Anwendung von DDD im aktuellen Projekt GSSN+?

  Auf den ersten Blick scheint es mehr Komplexität zu verursachen. Das hat aber nichts mit DDD zu tun, sondern mit den Microservices. Auf den ersten Blick scheint es einfacher zu sein, alles zusammenzuschmeißen. Aber im gesamten hat DDD nur Vorteile.

- Welche Probleme gibt es aktuell in der Anwendung von DDD im aktuellen Projekt?

  Die gemeinsame Sprache ist nicht so optimal. Das sieht man an der Frage was ein Outlet ist. Ich glaube nicht, dass die aktuelle Definition die richtige ist. Ein Outlet wird geöffnet und geschlossen. Wir sagen aber in GSSN+, wir erstellen und löschen ein Outlet. Wir benutzen also Begriffe, die ein Händler nicht benutzen würde. Das wäre aber eine natürliche Sprache. Aber wir haben uns davon verabschiedet und die Diskussionen darüber gelassen. Das bekommen wir nicht mehr weg. Jetzt haben wir einen Button »Create« und »Delete«. Die Diskussionen darüber waren zu lang.

- In DDD arbeitet der Domänenexperte direkt mit dem Entwicklerteam zusammen.
  Wer übernimmt deiner Meinung nach im aktuellen Projekt die Rolle des Domänenexperten?

  Sasa ist der Domänenexperte, aber das passt nicht ganz genau. Wir müssten eigentlich mit dem Händler reden. Er weiß, wie er die Verträge managed. Die Schwierigkeit dabei wäre, dass wir dann mit 50 Händlern

sprechen müssten. Jetzt wird es über Sasa zentralisiert. Da ist er schon der richtige Ansprechpartner. Eigentlich ist der Domänenexperte jemand, der täglich mit dem Geschäft zu tun hat. Bei uns geht es aber nicht anders. Sasa geht oberflächen-getrieben vor, er hat kein Interesse an DDD. Das liegt in seinem Naturell. Den Mehrwert, den das DDD bietet, müssten wir für ihn rausstellen.

- Muss der Domänenexperte sich auskennen mit DDD?

  Es ist nicht so wild, was der liefern muss. Wir müssen da gar nicht auf Kommando-Ebene mit ihm diskutieren. Die Ereigniskette ist interessant. Zum Beispiel die Information, dass die Dienstleistung gekündigt wird, wenn der Vertrag ausläuft. Sasa hat uns zum Beispiel mal erklärt, wir eine Firmenöffnung abläuft. Das sind dann die Events von selbst rausgelaufen. Man muss den Domänenexperten nur interviewen und daraus werden dann die Events generiert. Er muss nicht wissen was ein Aggregat oder eine Entität ist. Damit muss er sich ja auch gar nicht beschäftigen. Die Einteilung kommt aus dem Entwicklungsteam. Danach verifiziert man das mit dem Domänen-Experten. Mit dem Domänen-Experten sollte man nur auf Ebene von Fachobjekten und Ereignissen diskutieren. Was passiert, wenn die Business Site geschlossen wird?
  Die Kommandos im Modell würde ich auch heute weglassen. Kommando und Event sind eh gleich. Das sind auch keine Begriffe, die der Domänen-Experte sagen würde. Auf natürlich-sprachigem Weg ist das möglich. Viele Sachen kann man in der Diskussion weglassen. Ich stelle mir ein großes Poster vor, auf dem die Fachobjekte und die Events abgebildet sind. Damit kann der domänen-Experte etwas anfangen. Die Begriffe wie „Aggregat" oder „Value Objekt" sind nur Implementierungspattern. Am Anfang hatte ich gedacht, dass wir auf Aggregat-Ebene diskutieren müssten.

RE

- Welche Rolle spielt die Funktion RE im aktuellen Vorgehen für dich?
  Welche Aufgaben und Verantwortlichkeiten übernimmt das RE deiner Meinung nach im aktuellen Vorgehen?

  Ein Stück fachliche Erarbeitung des Modells. RE diskutiert mit dem PO über die Fachlichkeit, schreibt die Storys. RE ist zuständig für die Iterationsplanung, die Story Map und die Vorbereitung der Carving- und Refinement-Termine.

- Welche Verbesserungsvorschläge fallen dir ein?

  Vieles wurde schon besprochen. Zum Beispiel, dass die Entwicklung vorher auf die Strorys drauf schauen sollte bevor wir diskutieren. Ich persönlich finde die Story zu lösungsgetrieben. Was will der Kunde sollte dastehen, nicht wie.
  Die Aufteilung der Storys in Backend und Frontend ist nicht gut. Das sollte zusammen sein.
  Gefühlt haben wir drei Subteams. RE, UX und Entwicklung. Das ist weit weg von meinem Wunsch. Wir sollten enger zusammenarbeiten und detaillierter über das Fachmodell diskutieren.

- Zum Thema engere Zusammenarbeit.
  Mit wurde mal gesagt, dass es gar nicht gewünscht ist, dass das RE Team öfter in Ulm ist, weil ihr Entwickler dann von uns gestört werden würden.
  Ist das so?

  Ja, durch die vielen Termine die wir dienstags und mittwochs haben sind zwei Vormittage für konzentriertes Arbeiten schon im Eimer. Nicht die Anzahl der Tage ist ausschlaggebend, sondern die Zusammenarbeit an den zwei Tagen, wenn RE, Fachbereich und Entwicklung zusammen sind. Wir sollten in kleiner Runde

tiefer diskutieren.
Zu oft scheint es, rennen wir hinterher damit der Sprint geplant werden kann. In den Refinements wird dann aber zu viel diskutiert. Es wäre besser, wir gehen mit der Idee in ein Carving und schauen gemeinsam, was wir damit machen. Dann würden wir schneller durch die Refinements kommen.

- Ist es deiner Meinung nach notwendig, dass im aktuellen Vorgehen ein RE Team Teil des Vorgehens ist oder könnten die Aufgaben und Funktionen auch von anderen Rollen aus dem Team übernommen werden?
  Irgendjemand muss die Storys schreiben.

  Wir sind agil, aber man braucht Leitplanken. Das Aufbrechen von Epic zu Storys muss von jemandem gemacht werden. Die Entwicklung könnte anfangen Storys zu schreiben. Der PO hat keine Zeit und keine Lust. Auf der anderen Seite bekommen wir mit dem RE Team die Informationen über einen Mittelsmann rein. Das führt zu Dreiecksdiskussionen.

- Könnte das RE deiner Meinung nach weitere Aufgaben und Verantwortlichkeiten übernehmen, um das aktuelle Vorgehen zu unterstützen?

  Das RE Team könnte aufzeigen wo wir in der Iteration stehen. Da könnte man z.B. im Review gemeinsam darauf schauen. Was sind gerade die aktuellen Themen.

- So eine Art Roadmap?

  Ja.

- Ist das RE im aktuellen Vorgehen hilfreich, um in fachlichen Diskussionen zwischen Entwicklerteam und Fachbereich zu vermitteln?

  Das RE ist hilfreich, weil wir schon mit dem PO und UX diskutiert. RE bringt nochmal einen anderen Blickwinkel mit in die Diskussion.

- Werden Anforderungen verständlich und ausreichend dokumentiert?

  Die Storys sollten noch grober geschrieben sein.

Carving

- Wie beurteilst du die Carving-Termine in Bezug auf:

  Häufigkeit

  So wie es aktuell ist, ist es gut. Es wäre aber nicht schlecht einen längeren Zeitrahmen zu haben, wenn wir über größere fachliche Themen diskutieren würden. Zum Beispiel Verträge. Da würde es sich anbieten, wir würden uns für ein oder zwei Tage einschließen und ein Event Storming Workshop durchführen. Einen einmaligen Workshop also.

  Dauer

  Das kommt wie gesagt auf das Thema an

  Anzahl Teilnehmer

  Deutlich besser, wenn es so wie aktuell nur zwei Entwickler sind.

Art der Durchführung (Methodik)

Wir sind immer sehr konkret im Carving, ein andres Format wäre besser.

Diskussionskultur

Kreativität der Ideenfindung.

- Fließen deine Anmerkungen, Lösungsvorschläge und Kritiken in die Lösungsfindung mit ein?

  Kommt darauf an.
  Ja

Context Map

- Bis vor wenigen Wochen war das RE Team dafür zuständig, die Context Map zu erstellen. Aktuell liegt dies in der Verantwortung der Entwickler.
  Welche Vor- und Nachteile haben die jeweiligen Vorgehensweisen?

  Das aktuelle Vorgehen ist besser, weil wir in der Context Map sehr detailliert unterwegs sind, auf Pattern-Ebene. Auf diesem Detaillierungsgrad sind wir auf der Implementierungsebene, und da ist es besser, wenn wir da mitreden können. Mit dem alten Vorgehen war es problematisch, weil RE da nach bestem Gewissen modelliert hat und wir uns dann hinterher dazu abgestimmt haben.

- Werden deiner Meinung nach die fachlichen Zusammenhänge und Abhängigkeiten des Systems GSSN+ über die Context Map klar?

  Grundsätzlich ja. Das Wiki ist aber nicht das richtige Medium. Es ist schwierig das Gesamtbild zu bekommen. Wir bräuchten ein großes Poster, in das man hineinzoomen können sollte. So würden die Abhängigkeiten klar werden. Wir bräuchten ein Tool, in dem man Filtern kann und mir so zum Beispiel nur die Fachobjekte angezeigt werden, ohne die Information, ob es sich dabei um ein Value Objekt oder eine Entity handelt. Das Wiki ist gut für die Pflege, aber nicht zur Anzeige des Modells.

- Ist die Context Map deiner Meinung nach ausreichend modelliert und beschrieben, um als Systemdokumentation den Abnehmersystemen zur Verfügung gestellt zu werden?

  Momentan ist es schwierig. Für die Abnehmersysteme ist das weniger relevant. Die Schnittstelle ist relevanter. Das Ding muss funktionieren. Für den Fachbereich ist es relevant und für uns als Entwickler.

- Warst du bei den Event Storming Terminen dabei?

  Ja

- Was hat dir da gut gefallen?
  Was nicht so?

  Wir sind ohne Vorgabe eines Themas in die Diskussion. Das war nicht so gut.

- Was müsste verändert werden, damit es funktioniert?

  Wir bräuchten mehr Führung. Einer moderiert, es werden Fragen gestellt, und einer klebt die Events auf. Nicht alle auf einmal. Wir sollten das mit einem weniger kontroversen Thema versuchen, damit wird nicht im Kreis diskutieren.

Interview F

Interview zum Thema „Die Rolle des Requirements Engineerings (RE) in Domain-Driven Design (DDD)“

Kennung: F
Berufserfahrung: 5 Jahre

Einleitung

- Was ist deine Rolle im aktuellen Projekt GSSN+?

  Software Engineer

- Wieviel Spaß macht dir deine Arbeit im Projekt GSSN+ aktuell?

  Es macht mir viel Spaß.

- Wie viel Erfahrung hast du bereits praktisch und/oder theoretisch mit DDD sammeln können?

  In Retail haben wir auch schon DDD angewendet, aber da haben wir das eher ausprobiert und waren nicht so konsequent in der Anwendung. Erfahrung habe ich durch meine Arbeit im aktuellen Projekt, auch bekomme ich von den Kollegen einiges mit.

- Wieviel Erfahrung hast du mit Requirements Engineering, sowohl praktisch als auch theoretisch?

  Im Studium hatte ich Vorlesungen dazu; in kleineren Projekten, in denen es keine dedizierte Rolle RE gab, haben wir das als Entwickler dann auch selbst übernommen.

- Hast du selbst schon einmal Methoden, Techniken oder Vorgehensweisen aus dem RE angewendet?
  Sind dir die Projektziele und Iterationsziele des Projekts GSSN+ stets klar?

  Kommt drauf an. Die Richtung, wo es hingehen soll fehlt. Wir Entwickler haben unser eigenes Ziel definiert, also die Ablösung von Classic. Das Ziel ist nicht richtig klar.

- Wurde die Vision vorgestellt?

  Ja, immer mal wieder. Da hat der PO dann in einem Meeting die Vision vorgestellt. Das hatte geholfen, allerdings ändert sich das Ziel auch immer mal wieder, und damals stand z.B. die Navigation im Vordergrund. Die Iterationsziele sind weniger klar. Zum Beispiel wussten wir gar nicht, dass wir Iteration 1 schon abgeschlossen hatten. Die Iterationsziele sind ja in der Story Map, also transparent sind sie eigentlich schon, aber es gibt keine direkte Kommunikation der Iterationsziele.

Zusammenarbeit im aktuellen Vorgehen

- Wie bewertest du die Zusammenarbeit im ganzen Team oder zwischen den einzelnen Teams?
  Welche Aspekte der Zusammenarbeit gefallen dir gut?
  Welche Aspekte gefallen dir nicht so gut?
  Was könnte man verändern damit sich die Zusammenarbeit im gesamten Team oder zwischen den einzelnen Funktonen verbessert?

UX ist mittlerweile raus. Da bekommen wir nicht viel mit, was da passiert.
Die Story-Inhalte passen. Man weiß, was damit gemeint ist.

- Was hältst du von Event Storming?

  Das alte Fass mit dem Event Storming würde ich nicht mehr aufmachen. An unserem aktuellen Vorgehen würde ich nicht mehr rütteln. Es funktioniert so, wie es gerade läuft.
  Ein anderes Vorgehen wäre richtiger, aber wir sind jetzt durch mit dem Ausprobieren.

- Warum hat das Event Storming nicht funktioniert?

  Wir hatten keine Erfahrung und haben das erste Mal ein Event Storming Workshop durchgeführt. Wir haben es nicht geschafft, die Vorteile zu vermitteln, die man sich erhofft hatte. Umgekehrt war die Bereitschaft nicht da sich von den Oberflächen zu lösen und von dem Vorgehen, wie man es kennt. Wir hatten viele Chancen, aber es hat nicht gezündet.
  Wir hatten uns überlegt, von extern eine Moderation zu holen, jemand der weiß, wie man so einen Workshop durchführt, aber es war dann zu spät.
  Dann hat der Fachbereich sich eine Alternative überlegt.

- Was wäre denn ein richtiges Vorgehen?

  Eine fachliche Diskussion fernab der Oberfläche. Die Problemstellungen werden dann in Storys gepackt, ohne detaillierte Lösungsbeschreibung. Meiner Meinung nach bräuchten wir crossfunktionale Entwicklungsteams, die sich auch um die UI und das UX kümmern. Das haben wir aber nicht erprobt.

- Frustriert dich, dass wir jetzt anders vorgehen?

  Das hat mich frustriert, aber ich habe damit abgeschlossen.

DDD

- Was ist für dich DDD? Wie würdest du in ein oder zwei Sätzen erklären, was DDD ist?

  Die Fachlichkeit steht im Vordergrund und diktiert alles. Es gibt keine technischen Begriffsdefinitionen. Eine fachliche Sprache wird verwendet.

- Haben wir diese Sprache?

  Ja eigentlich schon. Wir haben einen Kompromiss.

- Schaust du manchmal ins Glossar?

  Nein, am Anfang habe ich das mal.

- Achten wir auf die fachliche Sprache?

  Nicht so richtig. Am Anfang waren wir da aber auch zu akademisch mit der Definition. Wir dachten, dass es heißen sollte »Outlet gründen« anstatt »Outlet erstellen«, weil das der natürlichen Sprache entspricht; aber dass der Anwender des Systems wahrscheinlich auch »Outlet erstellen« sagt und nicht »Outlet gründen«, haben wir in dieser Diskussion nicht berücksichtigt. Der MPC legt es eben auch nur an. Es gibt aber auch auf Fachbereich Seite keine Offenheit, die Begriffe zu hinterfragen.
  Das Problem ist, dass wir nicht an den richtigen Domänenexperten rankommen. Wir haben aber niemanden, der geeigneter wäre als den PO, also passt das schon.

- Wo siehst du Vorteile der Anwendung von DDD im aktuellen Projekt GSSN+?

  Das Team ist im gleichen Problem Space unterwegs. Man verwendet eine gemeinsame Sprache. Es wird nicht in der Oberfläche gedacht und auf der anderen Seite in Objektmodellen. Die Entwicklung kann man so beschleunigen, weil man sich auf die Fachlichkeit konzentriert und nicht darauf, wie sich das in der UI äußert.

- Welche Probleme gibt es aktuell in der Anwendung von DDD im aktuellen Projekt?

  Aus Entwicklersicht betreiben wir DDD. Auf Seite des Fachbereichs nicht.
  Ich weiß aber nicht ob es falsch ist. Es funktioniert. Wir wissen nicht, ob es anders besser funktionieren würde.

- Werden deiner Meinung nach fachliche Themen zwischen Fachbereich und Entwicklern ausreichend diskutiert, sodass sich gemeinsames Verständnis entwickeln kann?

  Ja. Doch. Wir diskutieren ja überall. Im Refinement, im Planning. Es mangelt nicht an Diskussionen.

RE

- Welche Aufgaben und Verantwortlichkeiten übernimmt das RE deiner Meinung nach im aktuellen Vorgehen?

  Anforderungen sammeln, in Storys gießen, Lösungen mit UI/UX abstimmen und mit in Storys einbeziehen.

- Welche Verbesserungsvorschläge fallen dir ein?

  Ich habe keine großen Anforderungen an das RE.

- Was sind deine Anforderungen?

  Den PO zu vertreten. Das RE Team als Entlastung und Puffer für den PO. UX und UI fährt parallel. Ich weiß auch nicht, wieviel Abstimmung tatsächlich zwischen dem UX Team und RE stattfindet.

- Antwort Interviewer:
  Wir bekommen die fertigen Oberflächenkonzepte und schreiben die Storys dazu.

  Das habe ich mir schon gedacht.

- Ist es deiner Meinung nach notwendig, dass im aktuellen Vorgehen ein RE Team Teil des Vorgehens ist oder könnten die Aufgaben und Funktionen auch von anderen Rollen aus dem Team übernommen werden?

  Aus Entwicklersicht ist kein RE nötig. Das könnte auch von Entwicklern übernommen werden.

- Wäre das besser?

  Ja, aber der PO müsste da mitspielen.

- Warum wäre das besser?

  Je direkter die Kommunikation, desto besser. Warum über einen Mittelsmann kommunizieren, wenn die Kommunikation auch direkt laufen könnte. Aus PO Sicht sehe ich aber einen Grund. Das RE Team puffert die Themen zum PO hin.

- Das Ziel meiner Arbeit ist es zu untersuchen, wo ein RE Team das Vorgehen in DDD unterstützen kann. Fällt dir dazu etwas ein?

  Da sehe ich die Rolle eher moderierend. Im Event Storming zum Beispiel.

- Könnte das RE auch nützlich sein, um dem Fachbereich DDD als Vorgehensweise näher zu bringen und den Mehrwehrt zu vermitteln?

  Ja, das könnte ich mir vorstellen.

- Könnte das RE deiner Meinung nach weitere Aufgaben und Verantwortlichkeiten übernehmen um das aktuelle Vorgehen zu unterstützen?

  Das RE könnte mehr Richtung UI gehen. Es verschwimmt alles.
  Aus Scrum Sicht bleibt nicht mehr so viel übrig für das RE.
  Das RE könnte auch die Nutzerperspektive vertreten und in Kommunikation mit dem Endanwender treten.

- Ist das RE im aktuellen Vorgehen hilfreich, um in fachlichen Diskussionen zwischen Entwicklerteam und Fachbereich zu vermitteln?

  Vermitteln tut das RE Team nicht. Es fängt eher die Sachen vor dem PO ab.

Carving

- Wie beurteilst du die Carving-Termine in Bezug auf:

  Häufigkeit

  Dauer

  Anzahl Teilnehmer

  Art der Durchführung (Methodik)

  Diskussionskultur

  Kreativität der Ideenfindung

  Die Reduzierung der Teilnehmer in den Carving Terminen ist okay. Es ist die Krücke, die dazu dient, den Team-Split nicht angehen zu müssen. Eigentlich sollten aber nicht nur die Vertreter der Teams hingehen, sondern das ganze Team. Für mich verschwimmen die Termine Carving und Refinement. Vieles hängt an der Teamgröße. Das Carving ist dazu da, das Gleichgewicht zwischen Entwicklern und Fachbereich herzustellen. Wenn nur Team Repräsentanten zu den Carving Terminen hingehen, führt das zu einem Wissensungleichgewicht. Das ist gefährlich, wenn wir in den Meetings rotieren, dann hängt zu viel Wissen bei einzelnen Leuten.
  Das Ideale für ein Carving wäre ein Event Storming Workshop.

Context Map

- Bis vor wenigen Wochen war das RE Team dafür zuständig, die Context Map zu erstellen. Aktuell liegt dies in der Verantwortung der Entwickler.
  Welche Vor- und Nachteile haben die jeweiligen Vorgehensweisen?

Beides ist suboptimal. Jetzt ist die Context Map eine Dokumentation. Wer interessiert sich aber dafür? Das RE vielleicht. Das Modell wird eigentlich gemeinsam bei einem Event Storming erarbeitet und dient als Diskussionsgrundlage. Wenn man es dazu nicht nutzt, ist es eben nur eine Dokumentation.

- Nach der Literatur zu DDD entsteht die Context Map in Zusammenarbeit zwischen Domänenexperten und dem Entwicklungsteam.
  Könntest du dir dieses Vorgehen auch für das GSSN+ Projekt vorstellen?

  Ja, das wäre schön

- Werden deiner Meinung nach die fachlichen Zusammenhänge und Abhängigkeiten des Systems GSSN+ über die Context Map klar?

  Ja, wenn man es richtigmacht. Bei der Pflege haben wir technische Einschränkungen. Man kann nicht rein oder rauszoomen. Am Anfang hatten wir das Modell auf Papier. Wenn der Wind kam, war alles weg. Das ist eine technische Hürde. Es sollte ja eine Diskussionsgrundlage sein. Es sollte keine Spezifikation sein. Die Nutzung als Dokumentation ist nur ein Teilaspekt. Mich würde mal interessieren, wie eine gelebte Context Map aussieht.

User Storys

Wir diskutieren da zu viel. Die Story sollte eine grobe textuelle Beschreibung sein. Von der Beschreibung passen mir die Storys wie sie aktuell sind.

Interview PO

Interview zum Thema „Die Rolle des Requirements Engineerings (RE) in Domain-Driven Design (DDD)"

- Was ist eure Rolle im aktuellen Projekt GSSN+?

  Product Owner

- Seit wann arbeitet ihr im GSSN Umfeld?

  T: Seid Oktober 2015

  S: Seid Dezember 2008

- In welcher Rolle, oder in welchen Rollen ward ihr bereits tätig?

  S: Schnittstellen, Entwicklung und Märkte

  T: Projekt Manager Muschi und Anforderungsmanager

- Was ist das Projektziel von GSSN+?

  T: Ein geiles System zu entwickeln

  S: Wir wollen für jeden Nutzer des Systems Mehrwert generieren. Das System soll Spaß machen. Die Benutzer sollen Bock auf das System haben. Es soll so aufgebaut sein, dass jeder unbewusst gern mit dem System arbeitet.

T: Sasa, du weißt ja am besten wie Kacke das alte System ist. Du bekommst ja auch die ganzen Supportanfragen.

S: Bestimmte Anfragen sind heute nur über eine Datenbankabfrage möglich. Die Daten haben großes Potential. Wir wissen ja selbst nicht was man damit alles machen kann. Das muss jeder selber wissen, und so sollten die Benutzer die Möglichkeit haben die Fülle an Daten für sich zu nutzen.

- Gibt es ein Vision Statement?

S: Das habe ich eben ja schon gesagt. Das ist die Vision des Systems.

- Wurde das auch Kommuniziert an das Team?

Ja, das haben wir gemacht, das wurde auch vom Entwicklungsteam auf Video aufgenommen.

- Was ist für euch DDD?

T: DDD bedeutet eigentlich nur, dass die Entwickler die Fachlichkeit besser verstehen. So bekommt der Developer die Fachlichkeit besser umgesetzt und es gibt weniger Rückfragen. S: Es ist einfach nur ein weiterer Ansatz das System zu entwickeln. In drei Jahren heißt es anders. Das ist wieder mal der Versuch etwas Besonderes zu machen. Ich frage mich, warum macht man nicht einfach? Das Problem ist, dass die Erfahrung nicht da ist. Da versucht man wieder was aus der Lehrer zu übernehmen. DDD ist ein schöner Ansatz.

- Wie wurde kommuniziert, dass mit DDD vorgegangen werden soll?

T: Simon hat das vorgestellt, wegen der Microservices. Dafür ist es bestens geeignet.
Da gibt es ein Buch von Brandolini. Ich weiß nicht ob das jetzt schon fertig ist. Da gibt es ein Zitat, das fand ich gut: „The bullshit asymmetry: the amount of energy needed to refute bullshit is an order of magnitude bigger than to produce it".
Das Problem ist, dass die Leute immer mitreden und mitdiskutieren wollen. Und am Ende behaupten sie, dass die Erde flach sei.

S: Die Entwickler maßen sich an mehr wissen zu wollen als der Enduser. Jeder hat seinen Bereich und seine Aufgaben. Sie müssen nicht mit dem Enduser sprechen.

- Haben sie sich das gewünscht?

S: Ja, das wollten sie. Als wir dann aber in Korea die Usertests durchgeführt haben um direktes Feedback von den Endanwendern einzuholen haben sie gefragt warum wir das machen. Man legt es sich wie man es braucht. Wenn heute alle Veganer sind, dann isst man Fleisch, und wenn alle Fleisch essen, dann wird man Veganer. Wenn ich am Bau eines Autos beteiligt bin, dann bin ich ein Rädchen in der Entwicklung, dann muss ich doch nicht verstehen wie die Felgen gebaut werden.

- Warum wurde die Abteilung RE mit Jörg beauftragt?

T: Weil Jörg eine Waffe ist. Den wollten wir in der Entwicklung dabeihaben. Am Anfang hat das RE jemand von der TSS übernommen. Aber das gab Probleme und dann hatten wir die Idee, dass wir Jörg dafür beauftragen können. Wir haben ihn jetzt da installiert wo es Sinn macht.

- Welche Probleme haben wir aktuell in der Entwicklung?

S: Das Rollenverständnis. Wenn du deine Rolle nicht akzeptiert wird alle diskutiert. Das ist nicht effizient. Der Anspruch an die Software ist, dass der Anwender damit etwas anfangen kann. Bei uns spielt der aber oft keine Rolle.
Du meinst bei den Entwicklern spielt das keine Rolle?
Ja auch, aber auch auf anderen Ebenen. Das ist ein Großkonzernproblem. Da geht es nur um Budget und Termine. Das Produkt an sich ist dabei leider oft zweitrangig.

T: Mir passt nicht, dass die TSS nicht offen für den Wettbewerb sein muss, weil sie eine Daimlertochter sind. Die haben ein ganz anderes Selbstverständnis. Mit anderen Unternehmen würde das anders laufen.

Extraktions- und Auswertungstabelle

Kategorie **Stimmung im Team**

Subkategorie	Interview	Rolle	Sachverhalt	Ursache	Wirkung
Frust	A	RE	Es gibt Abhängigkeiten zum PO oder UX weswegen man warten muss. Das ist nicht toll.	Abhängigkeiten zu PO und UX bestehen	Frustration
Spaß	A	RE	Es gibt immer wieder interessante Themen bei denen die Arbeit Spaß macht.	Interessante Themen fördern die Motivation und die Stimmung	Spaß bei der Arbeit
Spaß	C	ENT	Mittlerweile bin ich schmerzbefreit und kann entspannen und Spaß haben. Die Dinge laufen wie sie laufen, man könnte sich aufregen aber das bringt nichts. Es macht Spaß. Das hängt ja auch von vielen Faktoren ab. Nicht nur das Projekt, sondern z.B. auch die Kollegen mit denen man zusammen arbeitet.		
Spaß	B		Sehr viel Spaß		
Spaß	D		Sehr viel Spaß		
Spaß	E		Phasenweise gemischt. Mal sehr viel, mal weniger.		
Frust	D		In den Carvings hat man das Gefühl wir möchten die Fachlichkeit erarbeiten. Sonst nicht unbedingt. Viele Entwickler sind deswegen frustriert. Andere sind nicht so perfektionistisch.	Viele Entwickler sind frustriert, weil die Fachlichkeit nicht erarbeitet wird.	Frust im Team
Spaß	F		Es macht mir viel Spaß		
Frust	F		1. Was wäre denn ein richtiges Vorgehen? Eine fachliche Diskussion fernab der Oberfläche. 2. Frustriert dich, dass wir jetzt aber anders vorgehen? Das hat mich frustriert, aber ich habe damit abgeschlossen.	Die Diskussion der Fachlichkeit wird über die Oberfläche diskutiert	Frustration

Kategorie **Ziele**

Subkategorie	Dok.	Zeile	Sachverhalt	Ursache	Wirkung
Projekt- und Iterationsziel	A		Projektziel ist klar: Ablösung von GSSN Classic. Die Iterationsziele sind nicht alle klar. Die Iterationsziele ergeben sich im Laufe der Iteration.	Die Ziele sind nicht klar	

Subkategorie	Dok.	Zeile	Sachverhalt	Ursache	Wirkung
Projekt- und Iterationsziel	B		Im Groben ja, im Detail gibt es Abstimmungsbedarf. Das Ziel ist die Ablösung von GSSN Classic. Das neue System soll ein modernes Bedienkonzept haben. Auch das Thema Netzwerkplanung soll mit GSSN+ möglich werden. Das Ziel ist die Übersetzung der Vision in die Domänen. Die Ziele sind im HLD dokumentiert.	Die Ziele sind im HLD dokumentiert. Das HLD wird im Projekt als „überholt“ angesehen	Es gibt Abstimmungsbedarf
Projekt- und Iterationsziel	C		Der Gesamtüberblick in GSSN+ fehlt. Die Iterationen sind mal festgelegt worden, mir ist aber nicht klar ob das noch aktuell ist. Die Ziele sind schwammig definiert.	Die Use Cases die umgesetzt werden sollen sind nicht bekannt. Die Top-Five Feature fehlen. Die Vision sollte sich in den Anwendungsszenarien wiederfinden. Die Themes fehlen. Der Entwickler hat den Anspruch das Problem zu verstehen. Die Technik ist nur das Mittel zum Zweck.	Unsicherheit und Diskussionen über die Relevanz und Priorisierung von Themen
Projekt- und Iterationsziel	D		Die Iterationsziele sind nicht klar. In der Story Map stehen die Iterationsziele. Das Projektziel ist klar: Die Ablösung von Classic und der Lifegang.		
Projekt- und Iterationsziel	E		Die Iterationsziele sind grob klar. Das Projektziel ist klar.		Ein roter Faden durch das Projekt fehlt. Es wird oft diskutiert ob ein Thema Iteration 1 oder 2 relevant ist.
Projekt- und Iterationsziel	F		Die Iterationsziele sind nicht klar.	Es gibt keine direkte Kommunikation der Iterationsziele.	Die Richtung der Entwicklung ist nicht klar. Die Entwickler haben sich ein eigenes Ziel definiert.
Projekt- und Iterationsziel	PO		Es soll ein geiles System entwickelt werden, das Spaß macht. Der Nutzer soll dabei die Möglichkeit haben die Fülle an Daten für sich zu nutzen. Die Vision wurde von uns an das Entwicklerteam kommuniziert. Es wurde auf Video aufgenommen		

Kategorie **Aktuelles Vorgehen**

Code	Dok.	Zeile	Sachverhalt	Ursache	Wirkung
Fachliche Diskussion	A		Über Diskussionen kommt man zu Lösungen. Einwände zu Entscheidungen des POs sind ok, solange alternative Lösungsvorschläge gemacht werden. Entscheidungen vom PO sollten akzeptiert werden.		
Bewertung des Vorgehens	B		Die Zusammenarbeit im ganzen Team funktioniert. Es ist kein Best Practice. Es ist das beste was wir hinbekommen haben. Es ist eine Kompromisslösung.	Die Zusammenarbeit mit dem PO ist auf Domänenebene nicht mehr vorhanden. Der PO lässt sich nicht auf die Domänen ein. Der Wille sich damit auseinanderzusetzen fehlt. Er sollte sich aber auf die Denkweise einlassen. Das Vermögen abstrakt zu denken fehlt da.	Wir diskutieren von der Oberfläche her
Bewertung des Vorgehens	C		Der Faden durch das System fehlt. Die Grundvoraussetzung fehlt. Der Backlog ist z.B. nicht richtig priorisiert. Dem Kunden ist das egal. Hauptsache der Sprint ist voll. Der Faden fehlt. Wir schätzen häufig Stories, die in den nächsten Sprint keine Themen sind.	Die Reihenfolge der Themen fehlt. Nur Themen sollten im Carving besprochen werden die auch Thema des nächsten Sprints sein werden.	Damit wird dann das Team belastet. Das Team wird aus dem Kontext gerissen und wenn es soweit ist hat man das Thema wieder vergessen.
Bewertung des Vorgehens	E		Anforderungen sollten auf Epic-Ebene im Event Storming diskutiert werden. Es ist nicht gut, dass von der Oberfläche her diskutiert wird. Erst sollte das Modell und dann die Oberfläche diskutiert werden.		
Fachliche Diskussion	F		Eine fachliche Diskussion fernab der Oberfläche. Die Problemstellungen werden dann in Stories gepackt, ohne detaillierte Lösungsbeschreibung.		

Kategorie **DDD**

Subkategorie	Dok.	Zeile	Sachverhalt	Ursache	Wirkung
Definition DDD	A		Man hat eine Context Map. Die Domänen werden unabhängig voneinander betrachtet, jeder Bereich hat seine Funktion in der Domäne.		
Definition DDD	B		DDD ist die Weiterentwicklung eines fachlichen Datenmodells mit Eigenständigkeit der Domänen.		

Subkategorie	Dok.	Zeile	Sachverhalt	Ursache	Wirkung
Definition DDD	C		In er ersten Instanz ist es eine Methode zur Analyse von fachlichen Problemstellung für Transparenz und einem gemeinsamen fachlichen Verständnis. In GSSN+: ich möchte Händler Stammdaten verwalten.		
Definition DDD	D		In DDD steht die Fachlichkeit im Vordergrund. Es gibt Bounded Contexts und Entwickler und Domänenexperten sprechen die gleiche Sprache.		
Definition DDD	E		Modellierungsansatz bei dem die Fachlichkeit im Mittelpunkt steht.		
Definition DDD	F		Die Fachlichkeit steht im Vordergrund, und diktiert alles. Es gibt keine technischen Begriffs Definitionen. Eine Fachliche Sprache wird verwendet.		
Definition DDD	PO		T: DDD bedeutet eigentlich nur, dass die Entwickler die Fachlichkeit besser verstehen. So bekommt der Developer die Fachlichkeit besser umgesetzt und es gibt weniger Rückfragen. S: Es ist einfach nur ein weiterer Ansatz das System zu entwickeln. In drei Jahren heißt es anders. Das ist wieder mal der Versuch etwas Besonderes zu machen. Ich frage mich, warum macht man nicht einfach? Das Problem ist, dass die Erfahrung nicht da ist. Da versucht man wieder was aus der Lehrer zu übernehmen. DDD ist ein schöner Ansatz.		
Subkategorie	**Interview**	**Zeile**	**Sachverhalt**	**Ursache**	**Wirkung**
Erfahrung	A		Keine Erfahrung vor dem aktuellen Projekt vorhanden. Theoretische Grundlagen wurden über Bücher und dem Internet angelesen.		
Erfahrung	B		Die Theorie habe ich aus Büchern, GSSN+ ist das erste praktische Projekt das ich mache		
Erfahrung	C		Ich hatte davon gehört, aber ich hatte keine Erfahrung vor dem Projekt. Literatur wurde rumgereicht. Jemand hat das moderiert und gecoacht.		

Subkategorie	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
Erfahrung	D		Im Audit Tool Projekt haben wir DDD mal ausprobiert, das ging einige Jahre. Dazu kommt meine praktische Erfahrung in GSSN+. Da lerne ich viel „On the Job“. Da bekomme ich einiges mit. Außerdem habe ich ein Buch gelesen und an einem Workshop auf einer Software Konferenz teilgenommen.		
Erfahrung	E		Ich habe zwei Bücher gelesen, 2 Konferenzen besucht und eine Schulung gehabt. Praktisch habe ich vor GSSN+ bei For-Retail mit DDD rumexperimentiert. Da haben wir viele Anfängerfehler gemacht. Wir haben vor allem die Patterns angewendet und die Bounded Contexts nur stiefmütterlich betrachtet. Die Bounded Contexte gehören aber in den Vordergrund, und dann kommen die Pattern.		
Erfahrung	F		In 4Retail haben wir auch schon DDD angewendet, aber da haben wir das eher ausprobiert und waren nicht so konsequent in der Anwendung. Erfahrung habe ich durch meine Arbeit im aktuellen Projekt und bekomme auch von den Kollegen einiges mit.		

Kategorie **DDD**

Code	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
Vorteile DDD	A		Ich sehe keine Vorteile, aber auch keine Nachteile. In unserem Projekt unterstützt es die Anforderungsanalyse durch die gegebene Architektur mit vielen Services.		
Vorteile DDD	B		Die Dokumentationsfähigkeit der Geschäftsregeln. Der fachlichere Ansatz für eine Microservice Architektur. Die gemeinsame Spracheben		
Vorteile DDD	C		Es war zum Scheitern verurteilt. Zuviel Betriebsblindheit.	Von den gleichen Leuten die das Alt-System betreuen und betreiben kannst du GSSN+ nicht neu bauen lassen. Das war eine Fehleinschätzung	Man landet immer bei der gleichen Lösung.

Code	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
Vorteile DDD	D		Die Aufteilung in Bounded Contexts gewährt eine lose Kopplung.		Es gibt weniger Abhängigkeiten. Der Source Code ist übersichtlicher und die Wartung ist dadurch vereinfacht.
Vorteile DDD	F		Das Team ist im gleichen Problem Space unterwegs. Man verwendet eine gemeinsame Sprache. Es wird nicht in der Oberfläche gedacht und auf der anderen Seite in Objektmodellen.		Die Entwicklung kann man so beschleunigen, weil man sich auf die Fachlichkeit konzentriert und nicht darauf wie sich das in der UI äußert.
Vorteile DDD	PO		DDD bedeutet, dass die Entwickler die Fachlichkeit besser verstehen.		dadurch gibt es weniger Rückfragen gibt.

Kategorie **DDD**

Subkategorie	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
Probleme DDD	A		Die Diskussion über die Context Map fehlt. Das weiß ich gehört eigentlich zu DDD. Es gibt Situationen in denen die Bedeutung der Context Map für mich (als RE) nicht klar ist bzw. ich auch ohne auskommen könnte.		
Probleme DDD	B		Es verlangt in einem agilen Projekt, dass sich der PO mit dem Thema DDD auseinandersetzt. Der PO muss sich auf die Kontext Map einlassen. Die Akzeptanz beim Kunden muss vorhanden sein, dass das funktioniert.	Das Denken in Domänen ist komplizierter als im Objektorientierten.	
Probleme DDD	C		Man bräuchte Leute die die Methodik verstanden haben. Die wissen wie sie technisch funktioniert in der Umsetzung, und die das auch wertschätzen. Das sollte nicht von Extern motiviert sein. Das Commitment von allen ist nötig. Dafür müssen es aber alle verstanden. Und das nicht während der Projektlaufzeit, sondern schon im Vorprojekt.	Nicht alle Leute haben die Methodik verstanden.	Bis heute sind einige nicht motiviert. Die Context Map wird z.B. nicht von allen Beteiligten genutzt
Probleme DDD	D		In den Carvings hat man das Gefühl wir möchten die Fachlichkeit erarbeiten. Sonst nicht unbedingt. Viele Entwickler sind deswegen frustriert. Andere sind nicht so perfektionistisch.	Fehlende Diskussion der Fachlichkeit	Frustration

Subkategorie	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
Probleme DDD			Die gemeinsame Sprache könnte besser sein. Beiden Seiten (Drei Seiten) sollte es wichtig sein, dass man vom gleichen spricht. Wir Entwickler sollten aber nicht so dogmatisch sein und die Fachlichkeit vorgeben wollen bei der Begriffsdefinition. Auf allen Seiten sollte aber das Bewusstsein für eine gemeinsame Sprache da sein, das fehlt immer mal wieder.		
Probleme DDD	F		Aus Entwicklersicht betreiben wir DDD. Auf Seite des Fachbereichs nicht.		
Probleme DDD	PO		Ein Problem ist, dass die Entwickler mitreden und mitdiskutieren wollen. Es ist eine Anmaßung, dass der Entwickler mehr verstehen will als der Enduser. Der Entwickler muss nicht mit dem Enduser sprechen.		

Kategorie **Fachliche Diskussion**

Subkategorie	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
Fachliche Diskussion	B		Fachliche Themen werden nicht ausreichend diskutiert.	Es fehlt die Akzeptanz auf Seiten des Fachbereichs mit DDD zu arbeiten. Der PO lässt sich nicht auf die gemeinsame Sprache ein.	
Fachliche Diskussion	A		Dinge die vom PO vorgegeben sind, müssen nicht diskutiert werden. Fachliche Themen werden zwischen RE Team und Fachbereich besprochen und dann dem Entwicklungsteam in Form von User Stories vorgestellt. Eine Besprechung der Themen im Team würde helfen.	Fachliche Diskussionen fallen weg, weil das Carving oft ausfällt.	Diese fachliche Diskussion vorab fehlt den Entwicklern. Fachliche Themen werden zwischen PO und UX diskutiert und dann als Lösung vorgestellt.
Fachliche Diskussion	D		In den Carvings hat man das Gefühl wir möchten die Fachlichkeit erarbeiten. Sonst nicht unbedingt.		
Fachliche Diskussion					
Fachliche Diskussion					
Fachliche Diskussion					

Kategorie **Sprache**

Subkategorie	Interview	Zeile	Sachverhalt	Ursache	Wirkung
Sprache	B		Es fehlt die Akzeptanz auf Seiten des Fachbereichs mit DDD zu arbeiten. Der PO lässt sich nicht auf die gemeinsame Sprache ein.		
Sprache	C		Man tut sich schwer die Fachlichkeit auf der Sprachebene zu vermitteln.		
Sprache	D		Die gemeinsame Sprache könnte besser sein. Beiden Seiten (Drei Seiten) sollte es wichtig sein, dass man vom gleichen spricht. Auf allen Seiten sollte aber das Bewusstsein für eine gemeinsame Sprache da sein, das fehlt immer mal wieder.		
Sprache	E		Die gemeinsame Sprache ist nicht so optimal.	Die Diskussionen darüber waren zu lang.	Aber wir haben uns davon verabschiedet und die Diskussionen darüber gelassen.
Sprache	F		Wir achten nicht so richtig auf eine fachliche Sprache. Auf Seiten des Fachbereichs gibt es keine Offenheit die Begriffe zu hinterfragen.	Das Problem ist, dass wir nicht an die richtigen Domänenexperten ran kommen.	

Kategorie **Domänenexperte**

Subkategorie	Interview	Zeile	Sachverhalt	Ursache	Wirkung
Domänenexperte	A		Ein RE Team ist ein Domänenexperte auf kleinem Gebiet. Die richtigen Domänenexperten Sasa und Tim wissen was sie wollen. Haben einen Plan was das System können soll. à bringt das Projekt voran		
Domänenexperte	B		Der PO ist der Domänenexperte. RE nimmt diese Rolle aber auch ein, weil der PO auf dieser Ebene nicht diskutieren möchte. Das Abstraktionsvermögen fehlt da.		
Domänenexperte	C		Sasa ist nicht der Domänenexperte. Der weiß wie es momentan läuft. Aber er ist nicht der Experte für die Domäne. Der Domänenexperte darf nicht PO sein. Das System nähert sich immer mehr Classic an.		

Subkategorie	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
Domänenexperte	E		Sasa ist der Domänenexperte, aber das passt nicht ganz genau. Der Händler ist der Domänenexperte. Mit dem Domänenexperten muss man nicht auf Kommando-Ebene diskutieren. Die Ereigniskette ist interessant. Er muss nicht wissen was ein Aggregat oder eine Entität ist.		

Kategorie **RE**

Subkategorie	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
Rolle RE	A		RE ist notwendig.	Die POs können nicht alle Stories selber schreiben können. Fehlendes Technische Verständnis auf Seiten der POs wird durch uns übersetzt. RE als Schnittstellenfunktion also.	RE nimmt dem PO viel Arbeit ab.
Mögliche Rolle RE	A		Das RE könne den Prozess zur Lösungsfindung so unterstützen, damit nicht ständig um-entschieden wird.	Die Entscheidungen sind oft sprunghaft	da könnte das RE besser unterstützen
Rolle RE	B		RE übersetzt die Anforderungen für die Entwickler. RE bespricht die Domänen mit dem Entwicklungsteam. Technisches Verständnis ist wichtig zur Diskussion mit den Entwicklern.		
Rolle RE	B		Das RE ist der verlängerte Arm des POs. RE ist Anwalt des POs und nicht des Entwicklungsteams.	Es ist notwendig, dass da jemand vermittelt zwischen Fachbereich und Entwicklung.	Der Abstimmungsaufwand steigt durch die Rolle des RE zwischen beiden Parteien.
Rolle RE	C		Die Aufgabe des RE ist es Struktur in den Entwicklungsprozess zu bekommen. Zuerst sollte die Fachlichkeit geklärt werden und dann die UI. Ein Kompass durch das System ist wünschenswert.	Der PO kann nicht in Abläufen denken.	Der Weg im Projekt wird über die Oberfläche gegangen.
Rolle RE	D		Ein RE Team ist sinnvoll.	Für das Entwicklungsteam wäre es schwierig RE Tätigkeiten zu übernehmen	
Mögliche Rolle RE	D		Das RE kann das Event Storming moderieren.		

Subkategorie	Interview	Zeile	Sachverhalt	Ursache	Wirkung
Rolle RE	E		Die Stories müssen von jemandem geschrieben werden wenn der PO keine Lust oder keine Zeit hat.	Informationen kommen über das RE durch einen Mittelsmann rein.	Das führt zu Dreiecksdiskussion. Das RE bringt manchmal einen neuen Blickwinkel auf einer Diskussion.
Rolle RE	E		Das RE könnte im Review mit einer Roadmap aufzeigen, was die aktuellen Themen sind und wo wir in der Entwicklung gerade stehen.		
Rolle RE	F		Das RE könnte auch von den Entwicklern übernommen werden.		Das wäre besser, weil die Kommunikation nicht über einen Mittelsmann läuft.
Rolle RE	F		RE kann bei einem Event-Storming moderieren. Das RE könnte den Nutzer des Systems vertreten.		

Kategorie **Carving**

Subkategorie	Interview	Zeile	Sachverhalt	Ursache	Wirkung
Carving	A		Es macht Sinn neue Themen im Carving zu besprechen.		Werden Themen vor der Schätzung nicht im Carving besprochen führt das zu Diskussionen im Schätzmeeting.
Carving	A		Es wäre wichtig, dass die Entwickler im Carving dabei sind, die sich mit dem zu besprechenden Thema auskennen	damit sich das Team nicht erst dazu besprechen und abstimmen muss.	Manchmal waren welche mit im Carving die sich bei einem bestimmten Thema nicht so gut auskannten. Da haben wir etwas besprochen, und eine Woche später hieß es dann, das können wir nicht so machen.
Carving	B		Wir sollten und den Themen über die Domänen annähern und nicht über die Oberfläche. Erst diskutieren wir über die Oberfläche und dann überlegen wir wo es in den Domänen umgesetzt werden soll.	Sasa hat unterbunden, dass wir über Domänen diskutieren. Das ist nicht gut.	
Carving	B		Wir haben eine abgewürgte Diskussionskultur. Alle Parteien müssen gewillt sein ergebnisoffen zu diskutieren.	Ursachen liegen in schlechter Erfahrung mit dem Event Storming.	
Carving	C		Das Carving ist nützlich um sich über fachliche Themen Gedanken zu machen.		Über das Carving werden fachliche Themen wiederholt besprochen.

Subkategorie	Interview	Zeile	Sachverhalt	Ursache	Wirkung
Carving	C		Im Carving sollten wir einen Schritt zurück und auf Epic Ebene diskutieren. Die Aufgabe des RE ist es das als Epic zu erkennen. In den Epic sollten die Anforderungen aufgeschrieben werden, dann schauen wir gemeinsam was die Stories dazu sein könnten.	Viele Stories sind eigentlich Epics. Die müssten runter gebrochen werden.	Dann bekommt man das in den Griff, dass sie Stories oft falsch geschnitten sind.
Carving	C		Einen Überblick bekommt man über eine lange Wand auf der alle Aggregate und Kommandos enthalten sind. Zuerst sollten die Domänen und Bounded Contexts definiert werden. Danach geht man eine Ebene tiefer.	Nicht jedes Tool kann das darstellen.	
Carving	D		Die Carving Termine mit allen Entwicklern, dem Fachbereich und RE Team waren wichtig. Es ist schade, dass man das Gefühl hat, dass das Team im früheren Carving zu groß war.		Viele Entwickler entdecken eben mehr als nur wenige. Außerdem hat man dann die Fachlichkeit mitbekommen.
Carving	D		Ich bin zwar nicht so der Experte aber ich will dabei sein, wenn die Fachlichkeit besprochen ist.	Ich muss die Fachlichkeit zwar nicht selbst bearbeiten, aber verstehen will ich es.	
Carving	E		Es ist gut, dass die Stories im Carving besprochen werden.		So kann vermieden werden, dass eine bestimmte Lösung vorgegeben wird.
Carving	E		Ein großes Poster mit den Fachobjekten und Events ist eine gute Methode um mit dem Domänenexperten zu diskutieren.		
Carving	F		Ideal wäre für das Carving ein Event Storming Workshop		

Kategorie **Context Map**

Subkategorie	Interview	Zeile	Sachverhalt	Ursache	Wirkung
Context Map	A		Wenn RE die Context Map Modellierung übernimmt, kann es sein Verständnis aufbauen.		
Context Map	A		Wenn wir anhand der User-Stories über die Funktionen diskutieren, modellieren wir in der Vorstufe die Context Map		

Subkategorie	Interview	Zeile	Sachverhalt	Ursache	Wirkung
Context Map	B		RE übernimmt die Erstellung der Context Map		Führt zu hohem Abstimmungsaufwand mit den Entwicklern. Die Freiheitsgrade in der Lösungsfindung werden eingeschränkt. Context Map musste nachdokumentiert werden.
Context Map	B		Entwicklung übernimmt die Erstellung der Context Map		wir haben mehr Aufwand das Verständnis zu schaffen.
Context Map	D		Es ist besser, wenn das Entwicklungsteam die Context Map erstellt.	Context Map war vom RE Team zu genau beschrieben	Dadurch sind wir freier die Domäne zu gestalten.
Context Map	E		Es ist besser, dass das Entwicklungsteam die Context Map entwickelt da wir es sind die auf der Implementierungsebene mitreden sollten.		
Context Map	F		Die Context Map sollte als Diskussionsgrundlage dienen und nicht zur Dokumentation. Es ist unklar wer sich für die Context Map interessiert.		
Context Map	F		Es wäre schön, wenn die Domänenexperten und die Entwickler das Modell gemeinsam erstellen.		

Kategorie **User Stories**

Subkategorie	Interview	Zeile	Sachverhalt	Ursache	Wirkung
User Stories	A		Das RE Team stimmt sich zur Erstellung der User Stories ab. Sind die Entwickler nicht zufrieden können sie die Stories zusammenfassen oder dem RE Team eine Info dazu geben, damit das RE Team das anpasst		
User Stories	A		Wenn man weiß, dass etwas nicht anders umgesetzt werden kann, kann man die Lösung in der Story beschreiben.		
User Stories	C		Die Stories oft falsch geschnitten sind.		
User Stories	D		In den Stories stehen oft überflüssige Informationen.		
User Stories	D		Es ist besser, wenn Front- und Backend in einer Story umgesetzt wird.		

Subkategorie	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
User Stories	D		In der Story soll beschrieben sein "was" passiert und nicht "wie"		
User Stories	E		Die User-Stories sind zu lösungsgetrieben. "Was" nicht "Wie". Die User Stories sollten nicht in Backe. Und Frontend aufgeteilt sein.		

Kategorie **Event Storming**

Subkategorie	Inter-view	Zeile	Sachverhalt	Ursache	Wirkung
Event Storming	B		Wir haben schlechte Erfahrung mit dem Event Storming gemacht. Viel Zeit und Geld wurde da verbrannt.		
Event Storming	D		Das Event Storming ist schiefgelaufen. Das Event Storming darf nicht ziellos sein. Es sollte für ein aktuelles Thema genutzt werden.		
Event Storming	E		Wir bräuchten mehr Führung. Einer moderiert, es werden Fragen gestellt und einer klebt die Events auf. Nicht alle auf einmal.		
Event Storming	E		Wir sollten das mit einem weniger kontroversen Thema versuchen damit wird nicht im Kreis diskutieren.		
Event Storming	F		Wir haben es nicht geschafft die Vorteile eines Event Storming zu vermitteln.	Die Bereitschaft sich von der Oberfläche zu lösen hat gefehlt. Eine externe Moderation hat gefehlt	

Ergebnisse der teilnehmenden Beobachtung

September 2017

1. Wir starten jetzt die Context Map zu modellieren. Eigentlich sollte das Entwicklungsteam das machen, bis her haben sie aber noch nichts gemacht. Jörg entscheidet nun, dass wir das im RE Team übernehmen.
2. Jörg erzählt, dass das Entwicklungsteam gern die Nutzer des Systems in die Analyse miteinbinden möchte. Als Argument dagegen sagt er, dass es sich bei GSSN+ nur um ein Stammdatensystem handele, in dem Daten eingetragen werden. Die Benutzer dazu zu befragen sein sinnlos.

18.0A017

3. Im Carving Meeting wird über die Verwendung eines bestimmten Begriffs diskutiert. Ein Entwickler ist offensichtlich unzufrieden mit der Begriffsdefinition. Es wird ein großer Teil des Meetings darauf verwendet. Am Ende ist der Entwickler frustriert und der Fachbereich genervt von der Diskussion.

Oktober 2017

05.10.2017

4. Das Glossar enthält nur wenige Begriffe. Das Team verwendet das Glossar nicht.
5. Der PO entscheidet welches Thema jetzt aktuell ist für die Entwicklung. Wir fangen daraufhin an die Context Map zu modellieren und die Stories dazu zu schreiben. Im RE Team diskutieren wir über Value Objects und Aggregate. Wir modellieren da aber technische Lösungen, die wir dann dem Entwicklungsteam vorwegnehmen. Woher sollen wir wissen ob die Modellierung Sinn macht? Es stellt sich heraus, dass die Entwickler teilweise selbst erst während der Implementierung wissen was ein „Value Object“ und was besser als Entity umgesetzt werden soll. So stimmt dann unser Modell nicht mit der Implementierung überein.

Januar 2018

6. Den Entwicklern gefällt nicht, dass wir das Modell mit den DDD Patterns modellieren. Wir entschließen uns das Modell nur über Kommandos und Events zu modellieren.

08.01.2018

7. Vom Fachbereich gibt es häufig Beschwerden über das Entwicklungsteam. Sie würden zu viel Wissen wollen. Dadurch entstünden zu viele Diskussionen. Eine Aussage des POs lautet: „Wenn zu viel diskutiert wird, machen wir eben kein Domain-Driven Design mehr.

31.01.2018

8. Im Schätzmeeting diskutieren wir im Team wieviel Details in der Story beschrieben sein müssen. Einigen Entwicklern würde es reichen, wenn wir nur schreiben, welches Verhalten gewünscht ist. Andere wiederum wollen ganz genau beschrieben habe wie sich jedes Oberflächenelement verhält. Es ist schwierig es allen recht zu machen.

15.01.2018

9. Beim Mittagessen spreche ich mit den Entwicklern über die Storys. Einige Entwickler sagen, dass die Storys häufig zu lösungsgetrieben sind, und das RE-Team dadurch zu viel Lösungsdesign vorwegnimmt. In der Story sollte nur stehen „Was“ und nicht „Wie“.

Februar 2018

10. Das Entwicklungsteam übernimmt die Modellierung der Context Map.

26.03.2018

11. Im RE Team gibt es häufig Diskussionen darüber wie eine User Story geschrieben werden sollte. Die RE Teamleitung möchte die Stories nicht zu offen formulieren, damit die Entwickler „nicht einfach machen was sie wollen“. Nach Gesprächen mit den Kollegen aus der Entwicklung stellt sich heraus, dass es zwei Lager im Team gibt. Das eine möchte die User Stories nur ganz grob beschrieben haben, damit möglichst wenig Lösungsdetails vorgegeben werden, das andere Lager hätte gern detaillierter beschrieben.

04.04.2018

12. Beim Mittagessen wird über eine User-Story gesprochen. Diese war sehr detailliert beschrieben und aus Entwicklersicht zu sehr lösungsorientiert beschrieben. Ein Entwickler sagt: „Genau aus diesem Grund sollte bei der Lösungsfindung immer ein Entwickler dabei sein. Der könnte dann seine technische Sicht miteinbringen.

Mai

13. Wo sind die Ziele in der Story Map zu finden? Ich sehe sie nicht.

ABKÜRZUNGSVERZEICHNIS

AK	Akzeptanzkriterium
BUC	Business Use Case
DDD	Domain-Driven Design
ID	Identifikationskennung
IREB	International Requirements Engineering Board
MPC	Market Performance Center
Kap.	Kapitel
PAM	Purpose, Advantage, Measure
RE	Requirements Engineering
UL	Ubiquitous Language

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

REFERENZEN

Beck, K., & et al. (2001). Prinzipien hinter dem agilen Manifest. Retrieved from http://agilemanifesto.org/iso/de/principles.html

Brandolini, A. (2017). *Introducing EventStorming: An act of Deliberate Collective Learning*: Leanpub.

Brooks, F. P. (1886). No Silver Bullet – Essence and Accidents of Software Engeneering. *International Federation of Information Processing (IFIP) Congress '86*, 1068–1076.

Broy, M., Geisberger, E., & Kazmeier, J. e. a. (2007). Requirements Engineering: Ein Requirements-Engineering-Refernzmodell. *Informatik Spektrum*. (30), 127. Retrieved from https://link.springer.com/article/10.1007%2Fs00287-007-0148-5

Daimler AG. (2016). *High Level Definition for GSSNneu*. Version 1.0.

Evans, E. (2003). *Domain-driven design: Tackling complexity in the heart of software*. Upper Saddle River, NJ: Addison-Wesley.

Gläser, J., & Laudel, G. (2010). *Experteninterviews und qualitative Inhaltsanalyse als Instrumente rekonstruierender Untersuchungen* (4. Auflage). *Lehrbuch*. Wiesbaden: VS Verlag.

Gloger, B. (2016). *Scrum: Produkte zuverlässig und schnell entwickeln* (5., überarbeitete Auflage). München: Hanser.

Göthlich, S. E. (2003). *Fallstudien als Forschungsmethode: Plädoyer für einen Methodenpluralismus in der deutschen betriebswirtschaftlichen Forschung*. Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 578.

Hofmann, H. F., & Lehner, F. (2001). Requirements Engineering as a Success Factor in Software Projects. *IEEE Software*. (18), 58–66.

Hruschka, P. (2014). *Business analysis und requirements engineering: Produkte und Prozesse nachhaltig verbessern* (1. Aufl.). München: Carl Hanser Fachbuchverlag.

Institution of Electrical Engineers. (1880). *Managing complexity in software engineering*. *IEE computing series: Vol. 17*. London: Peregrinus on behalf of the Institution of Electrical Engineers.

Kittlaus, H.-B., & Fricker, S. A. (2017). *Software Product Management: The ISPMA-Compliant Study Guide and Handbook*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Laudon, K. C., Laudon, J. P., & Schoder, D. (2010). *Wirtschaftsinformatik: Eine Einführung* (2., aktualisierte Aufl.). *IT*. München: Pearson Deutschland.

Leffingwell, D. (2011). *Agile software requirements: Lean requirements practices for teams, programs, and the enterprise*. *The Agile software development series*. Upper Saddle River, N.J: Addison-Wesley.

Millett, S., & Tune, N. (2015). *Patterns, principles, and practice of domain-driven design*. Indianapolis, IN: John Wiley & Sons.

Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: a roadmap. *Proceedings of the Conference on The Future of Software Engineering*. (11), 35–46.

Patton, J. (2014). *User story mapping: [discover the whole story, build the right product]*. Beijing: O'Reilly.

Pichler, R. (2013). *Scrum: Agiles Projektmanagement erfolgreich einsetzen* (1. Aufl.). Heidelberg: dpunkt.verlag.

Pohl, K., Hönninger, H., Achatz, R., & Broy, M. (2012). *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Berlin, Heidelberg: Springer.

Pohl, K., & Rupp, C. (2015). *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering: foundation level nach IREB-Standard* (4., überarbeitete Auflage). Heidelberg: dpunkt.

Preußig, J. (2018). *Agiles Projektmanagement: Agilität und Scrum im klassischen Projektumfeld* (1. Auflage). *Haufe Fachbuch: Vol. 10248*. Freiburg, München, Stuttgart: Haufe Gruppe.

Robertson, S., & Robertson, J. (2013). *Mastering the requirements process: Getting requirements right* (3rd ed.). Upper Saddle River, N.J: Addison-Wesley.

Runeson, P., & Höst, M. (2008). *Empirical Software Engineering: Guidelines for conducting and reporting case study research in software engineering*: Springer.

Rupp, C. (2005). Ran an die Kundenwünsche: Ermittlungstechniken als Schlüssel zum Projekterfolg. *Enterprise Architektur Magazin*, 68–71.

Rupp, C. (2014). *Requirements-Engineering und -Management: Aus der Praxis von klassisch bis agil* (6., aktual. u. erw. Aufl.). München: Hanser.

Schwaber, K., & Sutherland, J. (2013). *Der Scrum Guide: Der gültige Leitfaden für Scrum: Die Speilregeln*. Retrieved from https://www.scrumguides.org

The Standish Group. (2016). *Chaos Report.*

Vernon, V. (2015). *Implementing domain-driven design* (Fourth printing). Upper Saddle River, NJ: Addison-Wesley.

Vernon, V., Lilienthal, C., & Schwentner, H. (2017). *Domain-Driven Design kompakt* (1. Auflage). Heidelberg: dpunkt.verlag.

Weilkiens, T. (2014). *Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur* (3., überarb. u. aktualis. Aufl.). Heidelberg: dpunkt.

Wirdemann, R., & Mainusch, J. (2017). *Scrum mit User Stories* (3., erweiterte Auflage). München: Hanser.