Masterarbeit

REQUIREMENTS-MANAGEMENT BEI DER PRODUKT-ENTWICKLUNG IM INDUSTRIELLEN UMFELD

ausgeführt am

Fachhochschul-Masterstudiengang
Automatisierungstechnik-Wirtschaft

von
Ing. Bernhard Grezl, BSc
1610322007

betreut und begutachtet von
Dipl.-Ing. Johannes Fritz, BSc

Graz, im Jänner 2018

.....
Unterschrift

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

Meiner Lebensgefährten, Mag. Eva Krinner, möchte ich für ihre Unterstützung und Motivation durch das gesamte Studium danken. Vor allem, dass sie mir die Zeit gegeben hat immer an meiner Masterarbeit zu arbeiten.

Ein weiterer Dank geht an meinen Betreuer Dipl.-Ing. Johannes Fritz, BSc, der sich immer Zeit für Fragen genommen und während des Erstellens der Masterarbeit sehr gutes Feedback auf meine Arbeit gegeben hat.

KURZFASSUNG

Diese Masterarbeit beschäftigt sich mit dem Requirements-Management bei mechatronischen Produktentwicklungen der Firma Anton Paar GmbH. Ziel dieser Arbeit ist es eine Möglichkeit zu finden, wie Requirements-Management bei einer Produktentwicklung in der Firma Anton Paar GmbH eingesetzt werden kann und die Definition, der dafür benötigten Werkzeuge und Diagramme festzulegen. Die theoretische Grundlage für eine strukturierte Anforderungsermittlung respektive -verwaltung ergibt sich aus den Themengebieten Systems Engineering, Requirements-Engineering und Requirements-Management. Des Weiteren sind die dazugehörigen Methoden, Modelle, Diagramme bis hin zum klassischen V-Modell Thema dieser Arbeit. Nach einer Erhebung des Ist-Standes des aktuellen Produktentwicklungsprozesses in Bezug auf das Requirements-Management wurde ein Requirements-Management-Leitfaden inklusive eines Requirements V-Modells entwickelt, welches über den ganzen Prozess hinweg eine Einteilung und Hierarchie aller benötigten Anforderungen samt dem Testaufwand abbildet. Nach diesem Modell sind in einem laufenden Entwicklungsprojekt unter Zuhilfenahme von Softwareprogrammen, Modelle und Diagramme erstellt und Anforderungen ermittelt worden. Der Einsatz des Leitfadens im Praxisbeispiel zeigt auf, wie das Requirements-Management in einer mechatronischen Produktentwicklung der Firma Anton Paar GmbH implementiert werden kann. Schlussendlich erfolgt eine wirtschaftliche Gegenüberstellung der zusätzlich anfallenden Kosten beim Einsatz des Leitfadens mit den Kosten des Aufwands bei einer nötigen Neukonstruktion im Falle einer übersehenen Anforderung.

ABSTRACT

This master thesis deals with the requirements management of a mechatronic product development in the company Anton Paar GmbH. The aim of this master thesis is to develop a possibility of the usage of the requirements management at the company Anton Paar GmbH for the product development and to define all necessary tools and diagrams. Since the structured requirements determination as well as the requirements management are theoretically based on systems engineering, requirements engineering and requirements management, they are discussed in detail. Additionally, methods such as, models, diagrams up to the V-Model are subjects of this thesis. The current state of the product development process with respect to requirements management is evaluated. Moreover, a requirements management guideline with a requirements V-Model has been developed to classify the requirements and illustrate a hierarchy of all demanded requirements and tests. With regard to this V-Model, models, diagrams as well as the requirements will be identified in a current development project. Furthermore, the usage of the guideline shows how requirements management can be implemented at a specific mechatronic product development at the company Anton Paar GmbH. Finally, the whole process will be reviewed with respect to an economic perspective comparing the costs of related to the usage of the requirements management guideline with those resulting from rectifying ignored requirements.

INHALTSVERZEICHNIS

1 EINLEITUNG

Die Firma Anton Paar GmbH mit Hauptsitz in Graz entwickelt, assembliert und vertreibt Labormessgeräte. Bei einer Entwicklung eines neuen Produkts wird ein Team benötigt, welches diese durchführt. Um bei einem Entwicklungsprojekt die Sicht der Firma und des Kunden einzubringen, besteht das Team aus dem Produktmanagement und dem Steering-Board (Projektauftraggeber und Führungskräfte). Zusätzlich sind Entwicklungsspezialisten aus den Bereichen Mechanik, Elektronik und Soft-/Firmware tätig, deren Tätigkeiten vom Projektleiter koordiniert werden. Zusätzlich kommen Produktspezialisten, Physiker und Techniker aus dem Product-Engineering hinzu. Da die Firma ISO 9001 zertifiziert ist, ist die Produktentwicklung auch standardisiert und der Produktentwicklungsprozess (PEP) in Phasen eingeteilt. Derzeit wird in diesen verschiedenen Phasen (Projektstarts-, Projektauftrags-, Spezifikations-, Entwicklungs-, Überleitungs- und Marktbeobachtungsphase) anhand verschiedener Dokumente das Produkt beschrieben, spezifiziert, technisch abgeklärt und entwickelt bis hin zur Überleitungsphase, in der das Produkt in ein Serienprodukt umgesetzt wird, in der das finale Testen stattfindet.

1.1 Ausgangssituation

Requirements-Management (RM) respektive Requirements-Engineering (RE) sind anforderungsgetriebene Zugänge zur Produktentwicklung, welche strukturiert auf Anforderungen (Requirements) eines Produktes eingehen und die Ermittlung sowie auch die Kontrolle dieser Anforderungen über die ganze Produktentwicklung hinweg sicherstellen können. Im vorher genannten PEP der Anton Paar GmbH sind Grundzüge von RM enthalten. Diese Grundzüge der Anforderungsanalyse sind aber noch nicht strukturiert erfasst und zudem noch nicht in einen Kontext mit RM gebracht worden. Grundsätzlich soll es mit RM möglich sein, dass die zu entwickelnden Produkte maßgeschneidert auf die Anwendung und somit für den Kunden entwickelt werden.

Zusätzlich gibt es am Markt auch Software-Werkzeuge, die das RM unterstützen. Damit können Anforderungen verwaltet und zu späteren Zeitpunkten systematisch getestet werden. Die Firma Anton Paar GmbH hat sich für die Software *Polarion*[1] entschieden, für die es noch keine genauen Vorgaben gibt, aber schon bei manchen Projekten ihren Einsatz fand.

1.2 Ziel

Das aus der Softwareentwicklung stammende RE und RM ist bereits ausreichend bestimmt. Daher soll in dieser Masterarbeit herausgearbeitet werden, wie man das schon bekannte RM auf eine mechatronische Produktentwicklung umlegen kann. Anhand des Ist-Standes im PEP und einer Untersuchung der aktuellen Verwendung von Polarion soll ein Konzept ausgearbeitet werden, wie RM in einem mechatronischen Produktentwicklungsprojekt eingesetzt und wie die Software *Polarion* dafür besser verwendet werden kann. Dieses Konzept soll bei einem laufenden Entwicklungsprojekt zur Anwendung kommen. Zu erstellende Diagramme sollen mittels Modellierungssprachen den Entwicklungsprozess begleiten und unter-

[1] Siehe Siemens Industry Software GmbH (2017), Online-Quelle [06.10.2017].

stützend einwirken. Schlussendlich soll die Wirtschaftlichkeit des RM untersucht werden, um eine Aussage treffen zu können, ob RM bei einer mechatronischen Produktentwicklung wirtschaftlich vertretbar angewandt werden kann.

1.3 Aufbau der Arbeit

Die Masterarbeit ist in einen theoretischen und einen praktischen Teil gegliedert. In der Theorie werden die Bestandteile RE und RM beginnend mit dem Systems Engineering bis hin zu unterstützenden Modellen und Diagrammen beschrieben. Im Praxisteil wird anhand des Ist-Standes des RM im PEP ein Requirements-Management-Leitfaden entwickelt, der in einem mechatronischen Produktentwicklungsprojekt seine Tauglichkeit unter Beweis stellt. Mit der Untersuchung der auftretenden Kosten wird schlussendlich die Wirtschaftlichkeit überprüft.

1.4 Firmenbeschreibung

Die Anton Paar GmbH ist ein weltweit agierendes Industrieunternehmen, das Labormessinstrumente, Prozessinstrumente und Automatisierungslösungen entwickelt und vertreibt. Dabei hält das Unternehmen mit Hauptsitz in Graz die Marktführerschaft im Bereich Dichte- und Konzentrationsmessung, der Rheologie und CO_2-Messung.[2] In Abb. 1 ist zur Veranschaulichung ein Auszug aus der Produktpalette der Anton Paar GmbH abgebildet.

Abb. 1: Übersicht einiger Anton Paar Produkte, Quelle: Anton Paar GmbH (2018), Online-Quelle [15.01.2018].

Gegründet wurde das Unternehmen 1922 als Reparaturwerkstätte vom Schlossermeister Anton Paar. Über die Jahre hinweg führten die Nachkommen von Anton Paar das Unternehmen weiter. Derzeit leitet Dr. Friedrich Santner als Geschäftsführer das Unternehmen. Mittlerweile hat das Unternehmen 26 Toch-

[2] Vgl. Anton Paar GmbH (2017), Online-Quelle [21.08.2017].

tergesellschaften und arbeitet mit 60 Vertriebspartnern zusammen. Mit über 2500 Beschäftigten ist das Unternehmen in 110 Ländern aktiv. Im Jahr 2016 hat es einen Jahresumsatz von 282 Millionen Euro erwirtschaftet. Seit 1994 ist das Unternehmen ISO 9001 zertifiziert. Das Qualitätsmanagement ist über die ganze Welt im Anton Paar Konzern etabliert und wird durch Audits überprüft.[3]

Wie bereits kurz erwähnt, ist ein Teilbereich der Anton Paar GmbH die Rheologie. In diesem Bereich stellt die Anton Paar GmbH selbst Rheometer her, die zur Bestimmung von Fließeigenschaften von Flüssigkeiten verwendet werden. Mit großer Erfahrung in verschiedenen Technologien schafft es die Anton Paar GmbH die Komponenten des Rheometers wie den luftgelagerten Electronically Commutated Motor (EC-Motor), spezielle Heizungen, die Anwendersoftware und bis hin zur vollautomatisierten Messplatzlösung selbst zu entwickeln und auch herzustellen. Das Rheometer selbst ist für einen breiten Anwendungsbereich durch die modulare Bauweise ausgelegt.[4] Die Anwendung dieser Produkte findet sich nicht nur in der Petrochemie, sondern in weiteren Gebieten wie der Lebensmittelindustrie, Kosmetik- und Arzneimittelindustrie wieder.[5]

Abb. 2: Firmenlogo der Anton Paar GmbH, Quelle: Anton Paar GmbH (2017), Online-Quelle [21.08.2017].

[3] Vgl. Anton Paar GmbH (2017), Online-Quelle [21.08.2017].

[4] Vgl. Anton Paar GmbH (2017), Online-Quelle [15.09.2017].

[5] Vgl. Anton Paar GmbH (2017), Online-Quelle [15.09.2017].

2 SYSTEMS ENGINEERING

Systems Engineering (SE) ist eine Systemtechnik, welche sich mit den Notwendigkeiten zur Entwicklung eines Systems beschäftigt. Dafür muss geklärt sein, wie ein System definiert ist und was das System genau ist. Nicht für jeden Auftraggeber ist das System gleich. Das hängt stark von der eigens definierten Systemgrenze ab. Beispielsweise ist für einen Lieferanten das zu liefernde Produkt das System, wobei im Gegenzug der Bezieher davon dieses nur als Teilkomponente seines Systems betrachtet.[6]

2.1 Definition des Systems

Als Definition kann auf den aus dem Jahr 1969 stammenden Militärstandard MIL-STD-499 für Systems Engineering Management zurückgegriffen werden, welcher definiert: *Ein System ist ein Gemisch von Betriebsmittel, Fähigkeiten und Techniken, welche eine Funktion ausführen und/oder die Funktion unterstützen. Das Gesamtsystem beinhaltet das ganze Equipment, Software, Dienstleistungen und Personen, die das System für den Betrieb benötigt. Es soll als eine autarke Einheit gesehen werden.*[7] In der Zwischenzeit hat sich die Beschreibung eines Systems jedoch weiterentwickelt. Folgende Eigenschaften sind mittlerweile hinzugefügt worden:[8]

- Ein System besteht aus komplexen Kombinationen von Ressourcen.
- Ein System beinhaltet eine Form von Hierarchie.
- Ein System ist in Subsysteme zerlegbar.
- Ein System muss einen Zweck erfüllen.

Die Definition eines Systems kann aber noch erweitert werden. Es kann verschiedene Arten und Ausprägungen eines Systems geben. Daher ist es zweckmäßig auch verschiedene Charakteristika eines Systems zu ermitteln, beispielsweise:[9]

- Natürlich oder künstlich erzeugte Systeme
  Bei natürlichen Systemen handelt es sich meist um einen natürlichen Prozess, wie z.B. bei Flüssen. Wohingegen künstliche Systeme ausschließlich von Menschen entwickelt werden.
- Physikalische oder konzeptionelle Systeme
  Konzeptionelle Systeme sind abstrakte Konzepte, welche nur am Computer oder auf Papier existieren und vor der physikalischen Realisierung verwendet werden. Physikalische Systeme sind dahingegen wirklich in Hardware aufgebaut.

[6] Vgl. Alt (2012), S. 7 f.

[7] Vgl. MIL-STD-499 (1969), S. 10.

[8] Vgl. Blanchard/Blyler (2016), S. 4.

[9] Vgl. Blanchard/Blyler (2016), S. 5 f.

- Statische und dynamische Systeme
  Systeme, welche ohne spezielle Aktivitäten eine bestimmte Struktur aufweisen, werden als statisch bezeichnet, beispielsweise Brücken. Im Gegenzug weisen dynamische Systeme nur mit einer Aktivität eine Struktur auf wie z.B. Produktionssysteme die mit einer Produktionsstätte kombiniert sind.
- Geschlossene oder offene Systeme
  Hier steht die Kommunikation nach außen im Fokus. Im Gegenteil zu den offenen, interagieren geschlossene Systeme nicht signifikant mit der Außenwelt.

Wie schon bei den Eigenschaften der Systeme kurz angesprochen, kann ein System auch aus Subsystemen, dem sogenannten *System eines Systems*, bestehen. Durch die vorhandene Hierarchie im Gesamtsystem kann durch die Einteilung des Gesamtsystems in Subsysteme die Auswirkung einer Aktion eindeutig für das Gesamtsystem festgestellt werden. Zusätzlich können auf Grund getrennter Behandlung des Gesamtsystems im Subsystem eigenständige Daten erzeugt werden, die nur im Subsystem verwendet werden.[10]

2.2 Bereiche des Systems Engineering

Wie in Abb. 3 ersichtlich, besteht das SE aus drei großen Bereichen. Der Bereich der *Systemanforderungen* ist der Part des RE sowie RM (siehe Kapitel 3). Hinzu kommen noch die *Systemarchitektur* und das *Systemverhalten*. Gemeinsam bilden sie alle Tätigkeiten des SE ab und definieren das System.[11] Als Abarbeitungsreihenfolge bietet sich an, mit den Anforderungen und der Architektur zu beginnen. Erst als zweiter Schritt soll das Systemverhalten bearbeitet werden.[12]

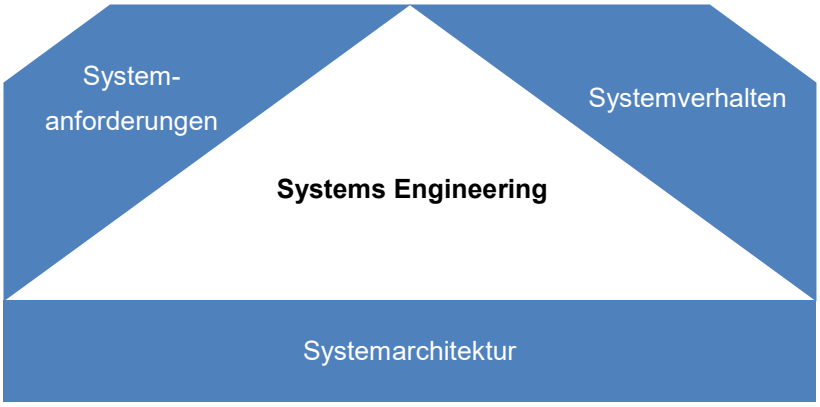


Abb. 3: Bereiche des Systems Engineering, Quelle: Eigene Darstellung.

[10] Vgl. Blanchard/Blyler (2016), S. 8 f.

[11] Vgl. Alt (2012), S. 9.

[12] Vgl. Alt (2012), S 15.

2.2.1 Systemarchitektur

Die Architektur beschreibt das System anhand der Struktur. Damit werden die Komponenten und die Schnittstellen zu anderen Systemkomponenten bzw. zur Systemgrenze festgelegt. Des Weiteren wird festgelegt, welche Daten, Materialien und Energie über die Schnittstellen ausgetauscht werden. Durch die statische Sicht auf das System wird z.B. der Zeitpunkt des Datenaustauschs nicht definiert. Daraus resultiert, dass die Architektur alleine nicht ausreicht, um ein System zu beschreiben.[13]

Die Systemarchitektur ist demnach nur eine Lösung von vorhandenen Problemstellungen, jedoch werden keine Funktionen beschrieben. Es ist das Verhalten des Systems, welches eine Funktion erfüllt. Die Architektur beschreibt die Lösung, welche gewählt wurde, sodass zum Schluss das System die gewünschte Funktion besitzt. Sie kann bei folgenden Punkten helfen:[14]

- Definition der Art und Weise der geforderten Funktionalität.
- Definition der Teile des zu realisierenden Systems.
- Überprüfung, ob das System die Erfüllung der geforderten Funktionalität mit der gewählten Architektur erfüllt.

2.2.2 Systemverhalten

Unter einem Systemverhalten sind grundsätzlich die funktionalen Anforderungen samt Unterkomponenten gemeint, welche alle Verhaltensweisen des Systems abdecken. Zusätzlich wird bei der Beschreibung des Systemverhaltens eine formale Darstellung der funktionalen Anforderungen hinzugefügt. Dabei beschreibt ein/eine TechnikerIn oder IngenieurIn die Anforderungen, sodass das Verhalten in eine Realisierung des Systems umgewandelt werden kann. Die Beschreibung soll so detailliert geschehen, dass es möglich ist, automatisiert Arbeitsprodukte zur Codegenerierung oder Testinformationen abzuleiten. Damit ist gemeint, dass sich Informationen auch rechnergestützt anhand von Algorithmen weiter verarbeiten lassen. Daraus kann ein Testablauf entwickelt werden. Um die Verhaltensbeschreibung durchzuführen, gibt es unzählige unterstützende Konzepte und Methoden. Je nach Domäne sind eigene Werkzeuge vorhanden. Modellierungssprachen wie die Systems Modeling Language (SysML) oder die Unified Modeling Language (UML) (siehe Kapitel 5.3), sind geeignet Systeme zu beschreiben.[15]

[13] Vgl. Alt (2012), S. 9.

[14] Vgl. Alt (2012), S. 10.

[15] Vgl. Alt (2012), 14 f.

3 REQUIREMENTS-ENGINEERING UND -MANAGEMENT

Wie bereits im Kapitel 2 erwähnt, sind das RE und RM Teilbereiche des SE, die sich mit den Systemanforderungen beschäftigen.

Der Sinn von RE respektive bei RM ist, dass bei einer Entwicklung die Anforderungen an das spätere Produkt bekannt sind, diese in der Entwicklung berücksichtigt werden und es damit zum gewünschten Ziel kommt. Schlussendlich soll das Produkt für den Kunden und somit den Benutzer erstellt werden, damit dessen Anforderungen erfüllt sind. Mit RE werden früh die Anforderungen der Produktqualität, der Integration und auch des Verhaltens ermittelt, um die Kundenzufriedenheit zu gewährleisten. Bei komplexen Systemen kommen Methoden und weitere Werkzeuge des RM zum Einsatz, die den Aufwand zwar erhöhen, aber durch die gewonnene Möglichkeit der frühen Fehlererkennung steigt die Chance Verzögerungen zu verhindern. Die Konzept-, Systembeschreibungs- und Detailanforderungen sollten somit die Grundlage jedes Entwicklungsprojekts sein. Daraus folgt, dass bei nicht vollständigen oder gar falschen Anforderungen das Projektziel nicht erreicht wird. Generell begleitet RM die laufende Entwicklung und regelt das Verwalten der Anforderungen im Entwicklungsprozess und stellt sicher, dass Änderungen richtig, konkret ergänzt und kontrolliert werden.[16]

3.1 Requirements-Engineering

RE wird angewandt, um die Anforderungen anhand der Aufgabenstellung respektive einer abstrakten Vorgabe zu präzisieren. Dies muss in einer Produktentwicklung anfänglich stattfinden. Dabei kommt es zu Überschneidungen mit Teilgebieten des SE (siehe Kapitel 2). Generell ist die Definition von RE so zu verstehen, dass es sich um das Ermitteln, Dokumentieren, Prüfen und Festlegen von Anforderungen handelt. Es soll dementsprechend sicherstellen, dass frühzeitig alle relevanten Anforderungen an das Produkt sowie den Detailierungsgrad soweit erfasst werden, dass alle Beteiligten den Inhalt verstehen und dass jeder Teilbereich weiß, was zu tun ist. Mit dem hiermit verteilten Wissen kann ein Commitment von allen Beteiligten eingeholt werden. Somit bildet RE die Grundlage zum Start der Entwicklung und stellt einen engen Bezug von Anforderungen und den entstehenden Modellen und Diagrammen (siehe Kapitel 5) her.[17]

Generell ist das RE mit dem SE stark im Austausch, da RE die Abstraktion der Anforderungen bildet und diese als Input für das SE zur Verfügung stellt. SE liefert genau dafür die Architektur als Basis für das RE. Beides muss in Abstimmung mit intensiver Kommunikation zwischen den beiden Bereichen stattfinden.[18]

[16] Vgl. Eigner/Roubanov/Zafirov (2014), S 53 f.

[17] Vgl. Eigner/Roubanov/Zafirov (2014), S 54 ff.

[18] Vgl. Weilkiens/Lamm/Roth/Walker (2016), S. 133 f.

3.2 Requirements-Management

Das RM beschäftigt sich mit der Verwaltung und Anpassung mit den aus dem RE gewonnen Anforderungen. Dabei können die Tätigkeiten in zwei grobe Bereiche eingeteilt werden:[19]

- Vorbereitende Tätigkeiten,
- Laufende Tätigkeiten.

Als vorbereitende Tätigkeiten, welche i.d.R einmalig durchgeführt werden, gelten Themen wie die Vorgehensweise des RE, die Definition des Analyseprozesses, die Abklärung der Zwischenergebnisse und die Festlegung der Dokumentationsstruktur (Methoden und Tools). Auch das Prüfverfahren kann vorab definiert werden. Grundlegender ist der zweite Teil, welcher die laufenden Tätigkeiten beschreibt. Hierzu gehören Tätigkeiten wie beispielsweise die Gruppierung von Anforderungen, Releases zu planen oder Änderungen einzupflegen. Wichtig ist, dass ein Verständnis herrscht, dass die im RE erhobenen Anforderungen nicht fixiert sind, sondern sich im Laufe der Zeit ändern. Daher ist es unerlässlich diese Anforderungen auch zu managen und Änderungen zu aktualisieren.[20]

Aus diesem Grund sind Anforderungen mit einer guten Struktur (siehe Kapitel 3.3.2) zu gestalten. Nach der ersten Dokumentation der Anforderungen, werden diese weiter verfeinert. Je mehr Personen daran arbeiten, desto mehr Änderungen wird die Anforderung erfahren. Daher ist es sinnvoll, vorab einen Leitfaden des RE zu entwickeln, um die Anforderungen dementsprechend eindeutig festlegen zu können. Wenn das RM durchdacht und geplant ist, kann es:[21]

- Die Kundenzufriedenheit erhöhen,
- die Qualität des Produkts verbessern,
- die Projektdurchlaufzeit und Projektkosten senken,
- die Komplexität reduzieren,
- die Kommunikation des Teams verbessern und
- Konflikte im Team zu vermeiden.

Die Notwendigkeit von RM ist gegeben, wenn eine hohe Zahl an Anforderungen und weiteren Informationen vorhanden ist, die Schätzungen der Lebensdauer des Produkts länger sind (zu viele Folgeversionen oder Wartungsintervalle), viele Personen involviert sind, schwer erreichbare Stakeholder betroffen sind und wenn ein umso komplexerer Entwicklungsprozess stattfindet. Im Grunde gibt es vier große Aufgabengebiete beim RM: der *Informationsaustausch*, die *Steuerung von Abläufen*, das *Verwalten der Zusammenhänge* und das *Auswerten und Projektsteuern*. Nicht jede Aufgabe ist in jedem Projekt gleichermaßen wichtig, dies muss immer individuell angepasst werden.[22]

[19] Vgl. Hruschka (2014), S. 295.

[20] Vgl. Hruschka (2014), S. 295 ff.

[21] Vgl. Rupp (2014), S. 369 ff.

[22] Vgl. Rupp (2014), S. 372 f.

3.2.1 Informationsaustausch

Sind viele Personen beteiligt, ist die Regelung des Informationsaustausches essentiell, da es schwierig ist alle Beteiligten immer am selben Informationsstand zu halten. Deswegen sollen alle Beteiligten auch laufend die Möglichkeit haben den aktuellen Stand der Anforderungen einzusehen und bei Änderungen benachrichtig werden. Helfen können hier Verzeichnisse auf einem Firmenserver, die Dokumentation auf einer Webseite oder spezielle RM Werkzeuge wie *Polarion* (siehe Kapitel 7.2.1), welche meist eine zentrale Datenbank aufweisen.[23]

3.2.2 Steuerung der Abläufe

Die Steuerung der Abläufe stellt sicher, dass nicht jede Anforderung unbedingt zu jeder Zeit von jedem bekannt sein muss. Vor allem darf nicht jede Anforderung von jedem abgeändert werden. Insbesondere bei sicherheitskritischen Anforderungen oder Projekten ist dies genau zu regeln. Wie die Steuerung der Abläufe zu erfolgen hat, hängt von den eingesetzten Methoden ab (siehe Kapitel 4). Je nach Methode können Zugriffe und Änderungen anders erfolgen.[24]

3.2.3 Verwalten der Zusammenhänge

Bei steigender Komplexität stehen Anforderungen zueinander in Verbindung und weisen Abhängigkeiten auf. Speziell wirkt sich ein dezentraler Dokumentationsort der Anforderung negativ aus, da bei einer Überarbeitung die Auswirkungen auf weitere Anforderungen nicht mehr im Blickfeld sind.[25]

Es soll daher eine sogenannte Traceability (Verfolgbarkeit bzw. Nachvollziehbarkeit) gegeben sein. Diese Verfolgbarkeit der Anforderungen kann klassifiziert werden. Wie in der Abb. 4 zur Verdeutlichung angeführt, stehen die Anforderungen in einer Beziehung zueinander und sind durch folgende drei Klassifizierungen zu unterschieden:[26]

- Pre-Requirements-Specification-Traceability
  Diese Nachvollziehbarkeit soll die Beziehung zum Ursprung der Anforderung beschreiben, welche schon verändert wurde.
- Post-Requirements-Specification-Traceability
  Das ist die Verfolgung der Anforderung zum Artefakt. Beispielsweise sind dies die Dokumentationen von Komponenten oder Implementierungen.
- Nachvollziehbarkeit zwischen den Anforderungen
  Hier findet die Betrachtung zwischen den Spezifikationen statt, welche Abhängigkeiten schaffen oder die Anforderung verfeinern.

[23] Vgl. Rupp (2014), S. 373 f.

[24] Vgl. Rupp (2014), S. 374.

[25] Vgl. Rupp (2014), S. 375.

[26] Vgl. Pohl/Rupp (2015), S. 132 ff.

Abb. 4: Beziehung der Nachvollziehbarkeits-Klassifizierungen, Quelle: In Anlehnung an Pohl/Rupp (2015), S. 133.

3.2.4 Auswertungen

Da in einem Entwicklungsprojekt immer operative Zwischenziele (Meilensteine) erreicht werden müssen, soll das RM hierbei unterstützen, sodass beispielsweise das Bearbeitungsdatum, der Umsetzungsstatus oder auch die vorhandenen Status der Testfälle zur statistischen Auswertung herangezogen und der Fortschritt aufgezeigt werden können.[27]

3.3 Die Anforderung

Terminologien im Sinne des Ingenieurwesens sind eindeutig definiert. Um einheitliches Verständnis für die Verwendung gewisser Terminologien im Sinne des Software Engineerings zu schaffen, ist der Standard ISO/IEC/IEEE 24765:2010(E) entwickelt worden, welcher die Begriffe textuell erklärt und festlegt.[28]

Grundsätzlich gibt es Design-, Funktions-, Implementations-, Schnittstellen-, physikalische und Leistungsanforderungen. Im Allgemeinen ist eine Anforderung wie folgt definiert:[29]

- Eine Bedingung oder Fähigkeit eines Anwenders ein Problem zu lösen, welches dabei hilft ein Ziel zu erreichen.
- Eine Bedingung oder Fähigkeit, die von einem System oder Systemkomponente erreicht werden muss, um die Spezifikation oder einen Standard zu erfüllen.
- Eine dokumentierte Darstellung einer Bedingung oder Fähigkeit im Sinne der ersten beiden Punkte.
- Eine Bedingung oder Fähigkeit, welche ein System besitzt.

[27] Vgl. Rupp (2014), S. 375 f.

[28] Vgl. ISO/IEC/IEEE 24765:2010(E) (2011), S. 1.

[29] Vgl. ISO/IEC/IEEE 24765:2010(E) (2011), S. 301.

3.3.1 Einteilung der Anforderung

Um Anforderungen besser einzuteilen, können Anforderungen nach ihrer Art gegliedert werden. Grundsätzlich wird zwischen *funktionaler* und *nicht funktionaler* Anforderung unterschieden. Allerdings ist die Einteilung in diesen zwei Arten nicht ausreichend, denn es können die nicht funktionalen Anforderungen noch weiter unterteilt werden. Eine weitere Auftrennung der nicht funktionalen Anforderungen kann daher lauten:[30]

- Technologische Anforderungen
  Beschreiben die Lösung, welche die Realisierung in der Umgebung des Systems einschränkt.
- Qualitätsanforderungen
  Definiert die Qualität des Entwicklungsprozesses bzw. des Systems.
- Anforderungen an die Benutzeroberfläche
  Regelt die Darstellung für den Benutzer.
- Anforderungen an sonstige Lieferbestandteile
  Sind zusätzliche Produkte, welche zur Lieferung des zu entwickelnden Systems benötigt werden.
- Anforderungen an durchzuführenden Tätigkeiten,
  welche für nachgelagerte Tätigkeiten anfallen.
- Rechtliche bzw. vertragliche Anforderungen.

Im Gegensatz zu nicht funktionalen Anforderungen handelt es sich bei funktionalen Anforderungen um Anforderungen an das Verhalten des Systems oder auch an die Systemkomponenten. Hierbei wird das dynamische Verhalten beschrieben. Beispielsweise kann es sich um Schnittstellen oder die Art des Datenaustauschs von und zu den Komponenten handeln. Gemeinsam mit der Architektur des Systems ist das System, in der Annahme es sind alle Anforderungen vollständig und ausreichend beschrieben, im Ganzen definiert.[31]

Um in den Anforderungen eine Priorisierung einfließen zu lassen, ist es von Vorteil die Anforderungen mit Bezeichnungen zu versehen, die bei Zeitknappheit oder einem engen Kostenrahmen die Entscheidung über die unbedingten und möglicherweise unwichtigeren Anforderungen erleichtern. Damit kann auch geklärt werden, welche sich daraus ergebenden Aufgaben als erstes umgesetzt werden müssen. Hier werden die Schlüsselwörter *must*, *will, must-not*, *required*, *shall*, *should* und *should not* eingesetzt. Die hierbei gewonnene Klassifizierung muss aber nicht in dieser Vielfalt stattfinden. Grundsätzlich reichen hier drei Einteilungen aus, *shall* (muss), *should* (sollte) und *will* (wird).[32]

[30] Vgl. Rupp (2014), S. 268 f.

[31] Vgl. Alt (2012), S. 10 f.

[32] Vgl. Rupp (2014), S. 18 f.

3.3.2 Qualität der Anforderungen

Um zu erkennen wie eine Anforderung formuliert sein muss bzw. um auch eine hochwertige Anforderung zu erlangen, sind Qualitätskriterien nötig. Diejenigen Qualitätskriterien, die genau einzuhalten sind, sind hilfreich, da erst mit der Erlangung einer exzellent ausgeführten Anforderung die Spezifikation nicht verfälscht wird. Anforderungen sollen daher *vollständig*, *notwendig, atomar, verfolgbar, technisch lösungsneutral, realisierbar, konsistent, eindeutig* und vor allem *prüfbar* sein. Als vollständig ist eine Anforderung zu bezeichnen, wenn die Spezifikation entsprechend beschrieben ist, sodass die Forderung der Stakeholder auch ohne deren Anwesenheit wiedergegeben werden kann. Zusätzlich sollen nur notwendige Anforderungen formuliert werden, welche eine Leistung oder Eigenschaft beinhalten, um das gewünschte Ziel zu erreichen. Dazu müssen diese auch immer aktuell gehalten werden. Des Weiteren ist nicht die Umsetzung beschrieben, sondern immer die Forderung an sich. Dabei sollen auch keine Einschränkungen des Systems erzwungen werden, die behindern. Damit werden Freiheiten gegeben, die keinen Lösungsweg ausschließen. Es muss auch darauf geachtet werden, dass jeder Anforderungssatz nicht mehr als eine Anforderung enthält. Dennoch muss eine Anforderung realisierbar sein, um das technische Risiko in Grenzen zu halten. Das Wichtigste ist, dass jede Anforderung durch einen Test respektive eine Messung auch bestätigbar ist, damit am Ende auch geprüft werden kann, ob das Ziel erreicht worden ist.[33]

[33] Vgl. Rupp (2014), S. 26 ff.

4 METHODEN IM REQUIREMENTS-ENGINEERING & -MANAGEMENT

Bei einer Produktentwicklung handelt es sich meist um eine technisch komplexe Entwicklung, welche den Einsatz von systematischen Methoden nötig macht. Mit diesen Methoden werden Produktkosten verringert, um eine größere Produktmarge zu erreichen. Meist stellen die Methoden während einer Produktentwicklung einen Mehraufwand dar, der sich aber rechnen, weil die Tätigkeiten strukturiert erledigt und somit Doppelarbeiten vermieden werden. Generell soll damit Arbeitsaufwand im Ganzen eingespart werden. Durch die verkürzte Produktentwicklungszeit kann sich am Markt ein Vorteil ergeben und es kann sich dadurch die Produktlebenszeit auch erhöhen. Da nicht nur ein Unternehmensbereich für eine ganze Produktentwicklung ausreicht, ist die Verbindung der Gebiete Forschung und Entwicklung, Produktion, Einkauf, Logistik bis hin zum Marketing äußerst wichtig. Es ist sicherzustellen, dass mittels des intensiven Einsatzes der Methoden sowohl die Abstimmung unter diesen beteiligten Abteilungen, die Spezifikation und als auch der Kundennutzen als Ziele festgelegt werden.[34]

4.1 Anforderungen ermitteln

Bevor die Anforderungen dokumentiert und verwaltet werden können, ist es unumgänglich die nötigen Anforderungen zu ermitteln. Damit soll erreicht werden, dass jeder das gleiche Wissen des Systemkontexts besitzt und eindeutig erkennbar ist, was entwickelt werden soll.[35]

4.1.1 Quellen der Anforderung

Um nun Anforderungen generieren zu können, muss geklärt werden, woher die Anforderungen kommen können und wer daher als Anforderungsquelle zur Verfügung steht. Als erstes gibt es den Stakeholder, welcher meist ein Benutzer, Auftraggeber, Tester oder Systemarchitekt ist und daher einen direkten Einfluss auf die Anforderung hat. Hier ist schon zu Beginn die Identifizierung der Stakeholder wichtig, da im späteren Verlauf der Entwicklung sonst unerwartete Anforderungen aufkommen können. Wenn Stakeholder z.B. erst mit dem fertigen Produkt das erste Mal in Berührung kommen, können die Änderungen respektive die neue Anforderung einen erheblichen Mehraufwand bedeuten oder auch dazu führen, dass das Projektziel nicht erreicht werden kann. Als Hilfe dafür kann eine Checkliste erstellt werden, die alle Stakeholder beinhaltet, sodass niemand übersehen werden kann. Zweitens gibt es Dokumente, wie z.B. Normen oder Gesetzestexte, welche Bedingungen beinhalten, die eine Anforderung ergeben. Als drittes sind noch die Systeme im Betrieb als Quelle vorhanden. Hierbei kann es sich um Vorgängersysteme handeln oder auch Konkurrenzsysteme, wobei hiermit durch die Benutzung der Stakeholder zusätzliche Anforderungen entstehen können.[36]

[34] Vgl. Graner (2015), S. 3 ff.

[35] Vgl. Pohl/Rupp (2015), S. 21.

[36] Vgl. Pohl/Rupp (2015), S. 21 ff.

4.1.2 Techniken zur Ermittlung von Anforderungen

Bei der Ermittlung von Anforderungen ist es das Ziel, die Wünsche und die Vorstellungen der Stakeholder an das Produkt zu einzuholen. Dabei gibt es verschiedene Methoden, bzw. Techniken, um die Randbedingungen zu klären. Eine universelle Methode für alle Projekte ist nicht auszumachen, aber verschiedene Methoden, die auf den jeweiligen Projektumfang als einzelne Methode oder in Kombination unterstützen können, sind vorhanden. Ungeachtet der gewählten Methode ist auf folgendes zu achten:[37]

- Unterschieden werden bewusste, unbewusste oder unterbewusste Anforderungen.
- Welche finanziellen Mittel, welcher zeitlicher Rahmen steht zur Verfügung und wie kann man Stakeholder kontaktieren.
- Der Einfluss von Erfahrungen von bestehenden Ermittlungstechniken.
- Die Risiken und Chancen des bevorstehenden Projektes.

Dennoch gibt es Methodenvorschläge, die sich eignen, um die Anforderungen zu ermitteln. Als Methode zum Ermitteln von explizitem Wissen der Stakeholder eignen sich Befragungstechniken, wie Interviews oder auch Fragebögen. Bei Interviews können genaue Fragen gestellt und dadurch neuaufgetretene Fragestellungen sofort beantwortet und notiert werden. Trotz des hohen Zeitaufwandes kommen auch unbewusste Anforderungen auf.[38]

Um die fast standardmäßig gewordenen Befragungstechniken (Interview und Fragebogen) zu erweitern, kann auch eine andere Art der Herangehensweise verwendet werden. Laut dem Kano-Model (siehe Kapitel 5.1) sind Anforderungen in drei Bereiche einzuteilen: Standard-/Basisfaktoren, Leistungsfaktoren und Begeisterungsfaktoren. Je nach Faktor der Anforderung hat dies Auswirkungen auf die Art der Erhebung, um die Wünsche von Kunden oder generell Anforderungen herauszufinden. Ungeachtet, dass Befragungstechniken, welche vorher schon beschrieben wurden, grundsätzlich immer anwendbar sind, können für Standard-/Basisfaktoren auch Beobachtungstechniken und vergangenheitsorientierte Techniken angewandt werden. Mittels Simulations- bzw. Animationstechniken sowie Kreativtechniken werden Begeisterungsfaktoren ermittelt. Schlussendlich gibt es noch Hilfsmittel und Werkzeuge, die bei der Anforderungsermittlung helfen können, wie beispielsweise im Kapitel 7.2 angeführt.[39]

Als Beispiel zum Ermitteln von Basisfaktoren ist die Methode der Dokumentenarchäologie zu nennen, welche dann zum Einsatz kommt, wenn keiner der Stakeholder verfügbar ist. Es werden dabei vorhandene Dokumente gesichtet und mit der derzeitigen Implementierung verglichen. Dabei wird meist festgestellt, was auch jetzt noch als Basisfaktor besteht. Alte Anforderungen sind meist auch Anforderungen, die für die Zukunft bestand haben. Der Nachteil ist der meist hohe Zeitaufwand, der dazu benötigt wird, da die Untersuchung der Implementierung und die damaligen Anforderungen nicht immer offensichtlich erkennbar sind.[40]

---

[37] Vgl. Pohl/Rupp (2015), S. 26.

[38] Vgl. Pohl/Rupp (2015), S. 27 f.

[39] Vgl. Hruschka (2014), S. 249 ff.

[40] Vgl. Hruschka (2014), S. 262 f.

Zum Ermitteln der Begeisterungsfaktoren sind grundsätzlich Kreativtechniken hilfreich. Hier kann Brainstorming oder auch das Brainstorming Paradox angewandt werden. Beim Brainstorming Paradox ist es das Ziel, Dinge zu suchen, welche verhindert werden sollen, unerwünschte Ereignisse, die eintreten können oder welche auf keinen Fall in diesem System geschehen sollen. Wenn diese Art des Brainstormings durchgeführt wird, lernt das Team, welche Risiken oder Maßnahmen in diesem Projekt anfallen können. Anfangs soll keine Beschränkung der Lösung, sondern eher der Fall der Anwendung deklariert werden. Das gibt dem Entwicklerteam mehr Freiheit, um das gewünschte Verhalten des Systems zu erschaffen. Als Hilfswerkzeug sind hier Use-Cases angedacht, die die Anforderung auf den Anwendungsfall beschränken, aber den Lösungsweg offen halten.[41]

Ein weiterer wichtiger Teil ist die Definition des Systems, welches entwickelt werden soll und wo dessen Grenzen sind, welche Umgebung notwendig ist und welche Schnittstellen benötigt werden. Dafür gibt es die sogenannte Kontextabgrenzung. Wie in der Abb. 5 veranschaulicht, werden dabei der Systemumfang (Scope), der Systemkontext (Umgebung) und die Systemgrenze definiert. Damit wird festgelegt, dass nur Einheiten, die das System betreffen auch Anforderungen erhalten. Ziel ist es zunächst jene materiellen und immateriellen Objekte zu identifizieren, welche auch eine Beziehung zum System aufweisen. Hiermit findet eine Erstellung einer Zukunftsvision des Systems statt, wenn es so wie geplant fertig integriert ist. Danach ist es notwendig, im Systemkontext die jeweils notwendigen Objekte (materiell und immateriell) zu identifizieren und zu spezifizieren. Dieses Diagramm bietet einen groben Rahmen für die weitere Anforderungsanalyse. Anfangs können noch Graubereiche der Grenzen entstehen, da meistens noch nicht alles ausdrücklich definiert ist. Diese Bereiche können erst nach dem Abschluss des RE komplett geklärt werden. Durchgeführt werden kann solch eine Kontextabgrenzung in Textform oder auch grafisch. Wichtig ist dabei nur eine der zwei Varianten anzuwenden, da sonst wieder Vermischungen stattfinden können und die Systemgrenze durch mögliche Doppeldeutungen einen Graubereich darstellt. Als grafische Variante dienen unter anderem Use-Case-Diagramme.[42]

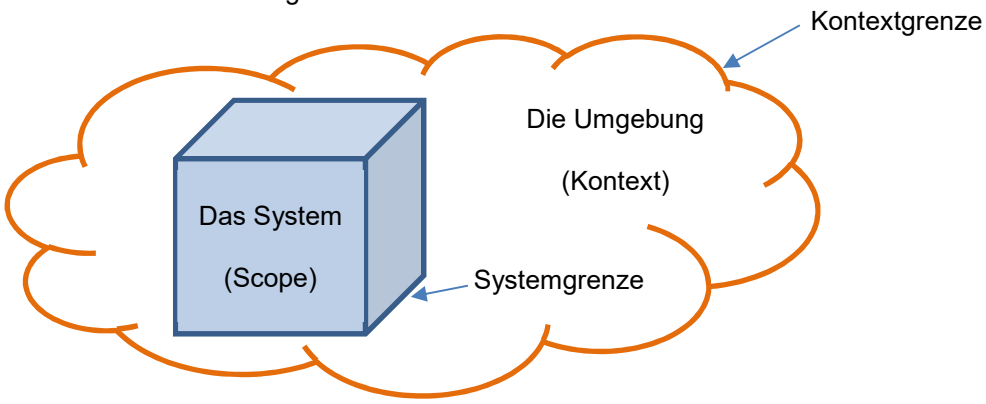


Abb. 5: Sinnbild der Kontextabgrenzung, Quelle: Eigene Darstellung.

[41] Vgl. Hruschka (2014), S. 264 ff.

[42] Vgl. Rupp (2014), S. 85 f.

4.2 Dokumentation von Anforderungen

Wenn die Anforderungen ermittelt sind, ist es zwingend notwendig, diese zu dokumentieren. Wichtig ist dabei, die Dokumentation dem zu entwickelnden System anzupassen. Grundsätzlich geht es um die formale Darstellung der Anforderung, sodass sie für jeden Stakeholder zur Erleichterung der Verständigung dient. Dabei kann die Anforderung in Prosaform bzw. in natürlicher Sprache oder durch Diagramme geschehen. Wenn Anforderungen dokumentiert werden, handelt es sich dabei auch um die Anforderungsspezifikation, welche danach eine Sammlung von Anforderungen widerspiegelt. All dies dient im RE als Basis für die Systementwicklung. Die Verwendung der Dokumentationsvariante des natürlichen Textes hat den Vorteil, dass jeder Beteiligte dies sofort verstehen kann und kein spezifisches Wissen von Modellierung vorausgesetzt ist. Dennoch kann mit einem Text eine gewisse Zweideutigkeit von Anforderungen nicht ausgeschlossen werden. Bei konzeptionellen Modellen bzw. Diagrammen erhält das Entwicklerteam zusätzlich eine Kompaktheit und erreicht eine höhere Eindeutigkeit, welche zu weniger Missverständnissen führt. Eines dieser Diagramme ist das sogenannte Use-Case-Diagramm, welches eine Übersicht der grundlegenden Verwendung und somit der Systemfunktionalität ermöglicht. Es beschreibt den Anwendungsfall der Funktionen, die das System für den Benutzer zur Verfügung stellen muss, damit dieser auch damit arbeiten kann. Jedoch nicht die Details, die dafür im System notwendig sind. Eine Mischform beider Varianten ist hier auch nicht ausgeschlossen, da der Einsatz von textueller Beschreibung und modellbasierter Dokumentation die Vorteile beider zusammenbringt.[43]

4.2.1 Textuelle Anforderungen

Zur textuellen Dokumentation ist ein vordefinierter Standard als Dokumentenstruktur sinnvoll. Dabei werden Hauptkapitel und verpflichtende Inhalte schon vorgegeben. Dies erleichtert bei neuen Mitarbeitern die Einarbeitung, Inhalte werden schneller erfasst, die Übersicht ist besser und die Vollständigkeit der Anforderung ist leichter überprüfbar. Die Vorlage laut V-Modell sieht hierzu die Verwendung eines sogenannten Pflichtenheftes und Lastenheftes vor. Wobei das Lastenheft bei der Produktentwicklung das *was* und *wofür* regelt und das Pflichtenheft die Umsetzung respektive die Realisierungsvorgabe und somit die Konkretisierung des Lastenheftes widerspiegelt. Die detaillierte Beschreibung ist im Kapitel 6 angeführt. Die hierbei erstellten Dokumente dienen dann als Grundlage für Planung, Architektur, Implementierungshilfe, Tests, Systemnutzung bzw. Wartung bis hin zum Vertragsmanagement. Als allgemeine Regel zur Dokumentation der Anforderungen sollte beachtet werden, dass nur kurze Sätze oder kurze Absätze und nur eine Anforderung pro Satz geschrieben werden soll. Verschachtelte Sätze sind nicht geeignet.[44]

4.2.2 Modellbasierte Anforderungen

Der Begriff *Modell* ist als abstrahierende Abbildung der Realität oder der noch zu schaffenden Realität zu verstehen.[45]

[43] Vgl. Pohl/Rupp (2015), S. 35 ff.

[44] Vgl. Pohl/Rupp (2015), S. 39 ff.

[45] Vgl. Pohl/Rupp (2015), S. 64.

Es besitzt drei Hauptmerkmale:[46]

- Abbildungsmerkmal
  Das Modell an sich ist immer eine Abbildung eines natürlichen oder künstlichen Originals. Dabei spielt es keine Rolle, wie das Original hergestellt wurde.
- Verkürzungsmerkmal
  Ein Modell beschreibt nie die gesamte Realität. Es werden nur Attribute erfasst, welche für das Modell als relevant eingestuft werden.
- Pragmatisches Merkmal
  Es ist keine eindeutige Zuordnung zum Original erforderlich, sondern stellt nur die Ersetzungsfunktionen (z.B. modellbenutzende Subjekte oder Zeitintervalle) dar.

Bei der modellbasierten Dokumentation werden somit die Anforderungen als abstrahierendes Abbild der Realität erstellt. Es handelt sich dabei um Zielmodelle und Use-Case Diagramme, welche im Kapitel 5 näher erläutert werden. Generell werden bei der modellbasierten Anforderung drei Ausprägungen erfasst:[47]

- Ziele, welche die Intention der Stakeholder erfassen, die auch miteinander in Konflikt stehen können,
- Use-Cases und Szenarien, welche die Nutzung des Produkts dokumentieren,
- Systemanforderungen, welche die detaillierte Funktionalität des Systems beleuchten, um die Eingaben für die Entwicklungsschritte aufzuzeigen.

Diese Form der Dokumentation unterscheidet drei Perspektiven der Anforderungen. Erstens die Strukturperspektive liefert Informationen der Ein- und Ausgabedaten und welche Nutzung und Abhängigkeitsbeziehungen das System besitzt. Zweitens die Funktionsperspektive, welche die Daten vom System in den Systemkontext widerspiegelt und drittens die Verhaltensperspektive, die den Zustand des Systemkontextes zustandsorientiert dokumentiert. Hierfür finden UML-Klassendiagramme, UML-Aktivitätsdiagramme und auch Statecharts ihren Einsatz.[48]

4.3 Prüfen von Anforderungen

Wenn Anforderungen ermittelt sind, ist es notwendig diese auf ihre Qualität und Korrektheit zu überprüfen. Ziel sollte sein, mögliche Fehler, wie z.B. Mehrdeutigkeiten, Unvollständigkeiten oder Widersprüche zu finden. Die somit freigegebenen Anforderungen dienen als Referenzdokumente für die danach folgende Entwicklung.[49]

Jeder Fehler in der Anforderung beeinflusst die Entwicklungstätigkeiten und kann den Aufwand der Entwicklung erhöhen. Wenn Fehler entdeckt werden, ist es notwendig die ganze Auswirkung zu erfassen

[46] Vgl. Stachowiak (1973), S. 131 ff.

[47] Vgl. Pohl/Rupp (2015), S. 63.

[48] Vgl. Pohl/Rupp (2015), S. 75 f.

[49] Vgl. Pohl/Rupp (2015), S. 95.

und alle betroffenen Dokumente, Quellcodes und Architektur abzuändern und anzupassen. Durchgeführt wird dies, in dem die gewonnenen Anforderungen mit den Wünschen bzw. Bedürfnissen der Stakeholder abgeglichen werden. Der Abgleich zwischen den Anforderungen und den Stakeholdern muss unbedingt vor der vertraglichen Vereinbarung stattfinden. Ziele wie Qualität, Zeitpunkt der Lieferung und der Leistungsumfang können sonst möglicherweise nicht eingehalten werden. Um die Qualität der Anforderungen gleichmäßig zu halten, ist es sinnvoll auf die Qualitätskriterien von Anforderungen zu achten (siehe auch Kapitel 3.3.2). Daher muss auch die Prüfung dementsprechend auf die Inhalte und mögliche Dokumentationskriterien durchgeführt werden.[50]

4.3.1 Überprüfungstechnik

Als Technik zur Überprüfung der Anforderungen sind die sogenannten Reviews geläufig. Davon gibt es drei Ausprägungen: *Stellungnahme, Inspektion* und *Walktrough*. Bei der *Stellungnahme* wird beispielsweise anhand einer dritten Person eine Expertise bezüglich der Qualität der Anforderungen eingeholt. Durch die vorher festgelegten Qualitätskriterien resultieren aus dem Feedback mögliche Qualitätsmängel. Die *Inspektion* nimmt eine systematische Suche nach Fehlern vor und beim *Walktrough* werden Anforderungen nur auf die Qualität hin von gezielt ausgesuchten Personen gesichtet.[51]

Über die Möglichkeit der Reviews hinaus haben sich noch drei weitere Techniken etabliert. Das *perspektivenbasierte Lesen, Prüfung durch Prototypen* und *der Einsatz von Checklisten*. Als Beispiel ist *das perspektivische Lesen* zu nennen. Hier kann aus verschiedenen Sichten die Anforderung auf unterschiedliche Arten geprüft werden. Aus Sicht der Kunden bzw. Nutzer kann die gewünschte Funktionalität geprüft werden. Durch die Sicht als Softwarearchitekt, zum Beispiel, wird die Architektur hinterfragt. Wenn Tester ihre Sicht einbringen, wird überprüft, ob sich aus den Anforderungen auch die nötigen Tests ableiten lassen. Je nach Komplexität und Umfang des Entwicklungsprojekts, sind weitere Sichten möglich.[52]

4.3.2 Konfliktlösungen

Konflikte im Bereich der Anforderungen können permanent auftreten. Daher ist es notwendig ein konsequentes Konfliktmanagement auszuweisen. Konflikte müssen identifiziert, analysiert, aufgelöst und dokumentiert werden. Sind Stakeholder nicht der gleichen Meinung bezüglich einer Anforderung, stellt dies einen Konflikt dar. Solche Situationen dürfen auf keinen Fall ignoriert werden und es gilt diese zu lösen.[53]

Nach den folgenden vier Punkten ist beim Konfliktmanagement vorzugehen:[54]

- Konfliktidentifikation
  Wie schon erwähnt, ist ein Konflikt eine Uneinigkeit bei Anforderungen. Wichtig ist die frühzeitige Erkennung dieser Problemstellungen.

[50] Vgl. Pohl/Rupp (2015), S. 95 ff.

[51] Vgl. Pohl/Rupp (2015), S. 104 ff.

[52] Vgl. Pohl/Rupp (2015), S. 107 ff.

[53] Vgl. Pohl/Rupp (2015), S. 112 f.

[54] Vgl. Pohl/Rupp (2015), S. 112 ff.

- Konfliktanalyse
  Hierbei steht die Ursache des Konfliktes im Fokus, um die Auflösung des Konflikts zu ermöglichen. Dabei wird die Konfliktart (Sachkonflikt, Interessenskonflikt, Wertekonflikt, Beziehungskonflikt etc.) festgestellt.
- Konfliktauflösung
  Die Art und Weise einer Konfliktauflösung bestimmt den späteren Willen der Beteiligten. Unumgänglich ist, dass alle Beteiligten an der Konfliktlösung mitarbeiten. Möglich ist eine Einigung, ein Kompromiss, eine Abstimmung, Ober-sticht-Unter oder eine Entscheidungsmatrix.
- Dokumentation der Konfliktlösung
  Die Auflösung eines Konflikts muss unbedingt dokumentiert werden. Ist das nicht der Fall, kann es zur erneuten Behandlung des gleichen Konfliktes kommen bzw. kann die dokumentierte Lösung als Überprüfung bei Anforderungsänderungen als Informationsquelle dienen.

4.4 Verwalten der Anforderungen

Bei der Verwaltung geht es darum, dass bei der Anforderungssammlung, welche bisher dokumentiert wurde, der Überblick erhalten bleibt. Es sollen keine Anforderungen übersehen oder redundante Anforderungen gebildet werden. Um diese Übersicht zu erlangen, gibt es mehrere Standardstrukturen, die die Anforderungen gliedern. Eine Art der Standardgliederung ist das V-Modell, das eine Lastenheftstruktur als standardisiert vorgibt.[55] Details dazu im Kapitel 6 Das V-Modell.

Unterstützend sollen Software-Werkzeuge zum Einsatz kommen, welche eine Übersicht der gewonnen Anforderungen strukturiert darstellen und Änderungen nachvollziehbar in allen Bereichen einfließen lassen. Grundsätzlich werden *Modellierungswerkzeuge* und *RM-Werkzeuge* unterschieden. Modellierungswerkzeuge liefern die Möglichkeit die vorhandenen Modelle (Kapitel 5) aufzunehmen und darzustellen. Es sollte dabei zumindest die Möglichkeit einer Schnittstelle zum RM-Werkzeug aufweisen, damit Änderungen im Laufe des Entwicklungsprojekts im Modell, als auch in den Anforderungen im RM-Werkzeug erfolgen. [56]

Das RM-Werkzeug wird eingesetzt, um die gewonnen Anforderungen in eine logische Beziehung zu bringen. Mit eindeutiger Identifikationsnummer der Anforderungen, Verfolgbarkeit, Zugriffkontrolle, Report-Erstellung und Ergebnisdokumentation soll die Software dahingehend unterstützen. Die eigens dafür geschaffenen Werkzeuge können über eine Benutzeroberfläche bedient werden. Durch verschiedene Import/Export-Funktionen ist die Kompatibilität mit anderen Werkzeugen gesichert.[57]

Zum Modellieren bieten sich die Modellierungssprachen UML bzw. auch *SysML* an (siehe Kapitel 5.3). Die dafür nötige Software können *Microsoft Visio*, Sparxsystems *Enterprise Architect* oder andere UML

[55] Vgl. Rupp (2014), S. 379 f.

[56] Vgl. Pohl/Rupp (2015), S. 149 f.

[57] Vgl. Pohl/Rupp (2015), S. 151 f.

unterstützende Modellierungsprogramm sein. Ein mögliches Werkzeug zur Verwaltung der Anforderungen ist *Polarion* (Details im Kapitel 7.2.1).[58]

Anforderung können Änderungen erfahren und unterschiedlichste Zustände während der Produktentwicklung einnehmen. Diese Zustände werden via Zustandsautomaten in eine Abhängigkeit gebracht. Diese Abhängigkeit regelt auch den RM-Prozess, damit dieser auch eingehalten wird. Als Beispiel sind Zustände wie *Angelegt, Umgesetzt, Getestet* zu nennen. Diese können aber beliebig je nach Projekt variieren. Es können Meilensteine als Zustand oder auch notwendige Tätigkeiten gewählt werden. Im Abb. 6 ist ein einfaches Beispiel einer solchen Abhängigkeit abgebildet.[59]

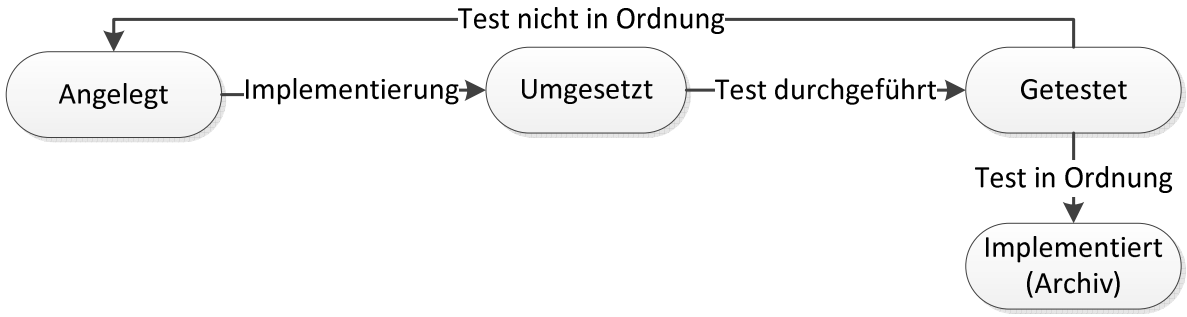


Abb. 6: Beispiel der Zustandsabhängigkeit von Anforderungen, Quelle: Eigene Darstellung.

4.5 Änderung von Anforderungen

In einem Entwicklungsprojekt ist es sehr wahrscheinlich, dass sich Anforderungen ändern. Auch bei einem umfangreichen RM werden auf Grund von Erfahrungen Änderungen stattfinden. Wenn dies geschieht soll der Ablauf einer Änderung geregelt in der Form eines vorherfestgelegten *Change-Managements* stattfinden. Dabei ist es irrelevant, ob die Änderung von EntwicklerInnen (Entw) stammt, die die Spezifikation aus technischen Gründen ändern möchten, ob Änderungen aus Informationen einer Iteration stammen oder das laufende System überarbeitet werden muss. Es gilt immer die Änderung der Anforderung auf die Qualität des Systems abzustimmen.[60]

[58] Vgl. Eigner/Roubanov/Zafirov (2014), S. 73.

[59] Vgl. Rupp (2014), S. 390.

[60] Vgl. Rupp (2014), S. 446 f.

Bei Änderungswünschen oder Gründen, die eine Änderung notwendig machen, ist es sinnvoll jeder Änderung auch eine eigene Nummer zu geben. Zusätzlich soll der Änderung eine Bezeichnung gegeben werden, damit die Tragweite der Änderung schon vorher abgeschätzt werden kann und eine Kategorisierung stattfindet. .[61]

Als Kategorien kommen in Frage:[62]

- *Bug*
  Hierbei handelt es sich um eine fehlerhafte Implementierung der Funktion, bei der aber eine korrekte Anforderung vorliegt. Somit muss nur der Fehler in der Implementierung behoben werden.
- *Defect*
  Dabei handelt es sich um eine Funktion, welche so nicht erwünscht ist. Es muss daher auch die Anforderung oder sogar die Architektur angepasst werden, um das fehlerhafte Verhalten zu beseitigen.
- *Innovation*
  Das sind Änderungswünsche, die von Stakeholdern gewünscht werden. Je nach Umfang lässt sich dieser Wunsch in das bestehende System integrieren oder es kann auch notwendig sein, dass ein eigenes Folgeprojekt in Auftrag gegeben werden muss.
- *Tuning*
  Bei dieser Kategorie handelt es sich um Anpassungen, die der Stabilität oder der Performance dienen. Es wird keine Funktion oder Anforderung geändert, sondern das System beispielsweise durch eine Kalibration verbessert.

Wenn ein Änderungswunsch aufgenommen wird, ist es essentiell, dass die Änderung und der Sachverhalt so gut beschrieben sind, dass auch ohne Nachfragen eindeutig ersichtlich ist, was geändert werden soll. Ist dies erfolgt, bietet es sich an, die anstehenden Änderungen in einem Gremium aus mehreren Stakeholdern zu diskutieren. In dieser Diskussion ist der Änderungswunsch z.B. eines Bugs aufgrund der Häufigkeit und der kritischen Situation, die durch den Fehler entsteht, zu priorisieren. Danach kann bzw. soll die Lösung grob skizziert werden, damit der Aufwand abgeschätzt werden kann. Sind diese Stadien durchlaufen, gilt es die Änderung mit anderen Aufgaben zeitlich abzustimmen und in die Ablauf- bzw. der Terminplanung auszunehmen.[63]

4.6 Methoden des Requirements-Engineerings im agilen Umfeld

Bei einer Produktentwicklung handelt es sich in der Firma Anton Paar GmbH oft um eine Entwicklung, welche mehrere Spezialbereiche benötigt (Mechanik, Elektronik, Firmware und Software). Dadurch kommen unterschiedliche Arbeitsweisen in einem Projekt vor. Vor allem bei Software- und Firmwareprojekten wird meist agil mittels Scrum gearbeitet. Daher wird kurz auf die agile Methode *Scrum* eingegangen. Darauffolgend wird angeführt, wie RM in Bezug auf Scrum stattfindet.

[61] Vgl. Rupp (2014), S. 449 f.

[62] Vgl. Rupp (2014), S. 449 f.

[63] Vgl. Rupp (2014), S. 451 ff.

4.6.1 Scrum

Scrum ist eine agile Projektmanagementmethode, bei der in sogenannten Sprints gearbeitet wird. Abb. 7 zeigt dazu den groben Ablauf. Ein Sprint ist ein begrenzter, sich wiederholender Zeitbereich (30 Tage oder weniger), in welchem die Abarbeitung von Arbeitspaketen stattfindet. Der Product-Owner vertritt die Sicht des Kunden und erhält die Anforderungen. Die von ihm vordefinierten Arbeitspakete werden in einem Backlog angehäuft. Der sogenannte Scrum-Master organisiert die Planung der Sprints in Abstimmung mit dem Entwicklungsteam und plant die Arbeitspakete in die nächsten Sprints ein. Um die Kommunikation im Team hoch zu halten, werden täglich kurze *daily stand-up meetings* abgehalten. Mit dem Fokus auf die kurzen Zeitintervalle ist die Planung agiler und die Arbeitspakete werden sukzessive abgearbeitet.[64]

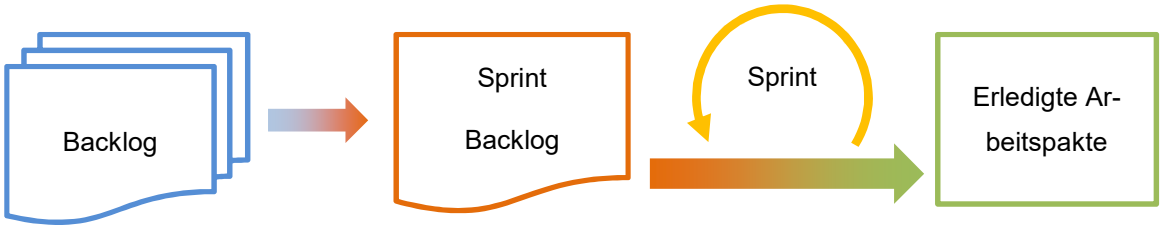


Abb. 7: Scrum Ablauf, Quelle: Eigene Darstellung.

4.6.2 Requirements-Engineering in Scrum

Aus einem Scrum-Prozess gehen grundsätzlich die Tätigen der Anforderungserhebung und -verwaltung in Bezug auf RE und RM nicht hervor. Trotzdem sind Tätigkeiten vorhanden, die den Einsatz von RE notwendig machen. Als Beispiele kann angeführt werden, dass der Product-Owner Ermittlungstechniken zur Erstellung des Backlogs anwendet, damit die Anforderungen der Stakeholder aufgenommen werden können. Zusätzlich muss das Entwicklungsteam die detaillierten Anforderungen in der Sprintplanung erfahren können. Wie in Abb. 8 gezeigt, gibt es generell drei Zeitpunkte, in welchen das RE bzw. RM anwendbar sind: vor der Entwicklung, im Ablauf und nach der Entwicklung.[65]

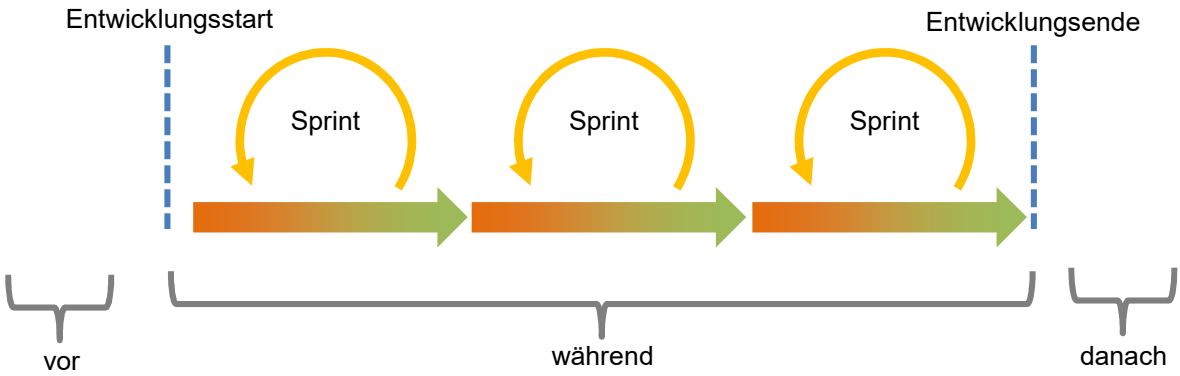


Abb. 8: Zeitpunkte des RE bei der Verwendung von Scrum, Quelle: In Anlehnung an Rupp (2014), S. 62.

[64] Vgl. Leffingwell (2011), S.15.

[65] Vgl. Rupp (2014), S. 62.

Bei der Verwendung von *RE und RM vor der Entwicklung* wird in den Scrum-Prozess nicht eingegriffen. Streng nach Scrum sollte auch die Anforderungsanalyse während eines Sprints stattfinden, um die benötigte Information zum richtigen Zeitpunkt zu erhalten. Wenn das Ermitteln der Anforderungen mittels RE schon vor der Entwicklung stattfindet, hat es den Vorteil, dass die Projektinhalte schon definiert sind. Dadurch werden Abschätzungen des Gesamtprojektes übersichtlicher und genauer. Im Laufe der Entwicklung kann der Product-Owner oder auch das Entwicklungsteam auf die schon vorhandenen Anforderungen zurückgreifen, diese ausbauen oder abändern.[66]

Bei der Anwendung von *RE während der Entwicklungsphase*, wird das RE in den Ablauf des Scrum-Prozesses integriert und Entwickler müssen selbst im Sprint Analysetätigkeiten durchführen. Hier werden hohe RE-Kompetenzen bei den Entwicklern gefordert, da sie selbst die Anforderung vom Product-Owner erheben und die Auswirkungen auf alle betroffenen Systeme selbst einschätzen müssen. Das Entwicklerteam handelt hier mit hoher Eigenverantwortung, welches der Scrum-Arbeitsweise am nächsten kommt. Der dadurch gewonnene Vorteil ist die Aktualität der Anforderung. Auch *nach dem Entwicklungsende* ist RE möglich. Dies kann geschehen, wenn eine Dokumentation der relevanten Aspekte der Implementierung erstellt werden soll. Es entsteht eine Dokumentation von Anforderungen, welche keiner Änderung mehr bedürfen. Findet *RE nach der Entwicklung* statt, handelt es sich um eine Dokumentation der Leistung des Produkts. Diese Variante ist die schlechteste Variante, da das Team nach einer gewissen Zeit nicht mehr genau weiß, wie die Anforderung genau umgesetzt wurde und sich wieder in die Thematik einarbeiten muss. [67]

Generell gilt RE auch als eine sich immer wieder wiederholende Tätigkeit (Ermitteln, Dokumentieren, Abstimmen und Anpassen) und die dadurch zu Scrum passt. Wird das RE durch einen Use-Case getrieben, werden davon die User-Stories, die für Scrum benötigt werden, abgeleitet. Zusätzlich können die gewonnenen Anforderungen aus einem Sprint als Grundlage für den nächsten Sprint genommen werden. Darum ist es sehr gut möglich RE in eine agile Arbeitsweise zu integrieren.[68]

4.7 Projektfortschritt anhand von Anforderungen verfolgen

Anhand der Anforderungen in einem Projekt kann mit Zuhilfenahme des Status der vorhandenen Anforderungen (siehe Kapitel 4.4) ein Projektfortschritt ermittelt werden. Die Zahl der Anforderungen und deren Status mit den geleisteten Aufwänden werden dabei gegenübergestellt. Als Übersicht kann ein sogenannter *Burndown-Graph* (siehe Abb. 9) erstellt werden, welcher von 100 % - 0 % die offenen Anforderungen mit dem Zeitraum, der zur Entwicklung zur Verfügung steht, zeigt. Die Summe der zu leistenden Aufwände, die zur Implementierung aller Anforderungen benötigt werden, sind in Tagen auf der Y-Achse aufgetragen und repräsentieren dabei 100 % des Aufwandes. Die X-Achse zeigt den Zeitraum, bei dem der aktuelle Status wieder aufgenommen wird. Ob der Zeitraum einen Tag, eine Woche, einen Monat

[66] Vgl. Rupp (2014), S. 62.

[67] Vgl. Rupp (2014), S. 63 f.

[68] Vgl. Rupp (2014), S. 69.

oder auch einen Sprint (siehe Kapitel 4.6.1) beträgt, ist je nach Projekt unterschiedlich. Die implementierten Anforderungen werden abgezogen bis am Fertigstellungstermin 0 % übrig bleibt. [69]

Missstände, Änderungen oder Abweichungen sind somit schnell erkennbar. Die daraus gewonnene Information wird zur Projektsteuerung herangezogen. Um weitere Hinweise auf Verzögerungen oder Budgetüberschreitungen zu erlangen, kann der Nutzen der implementierten Anforderungen zusätzlich in die Betrachtung miteingebracht werden. Generell sind die Restaufwände des Projekts mit dieser Methodik sehr gut darstell- bzw. feststellbar und helfen dem Projektmanagement bei der Übersicht des Gesamtprojektes.[70]

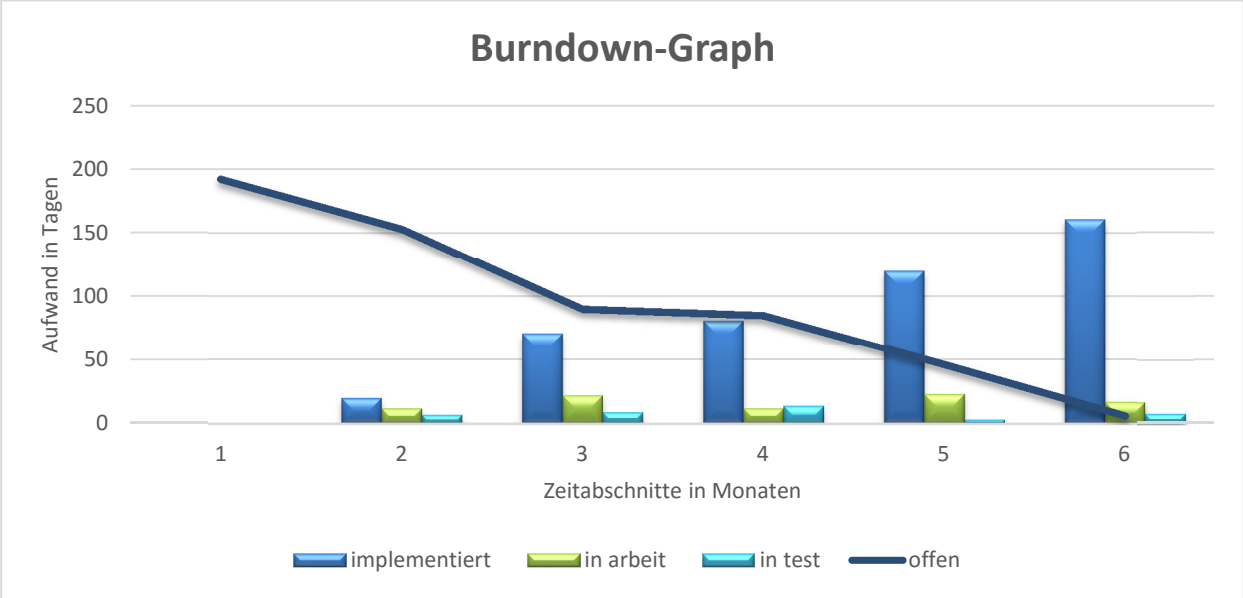


Abb. 9: Beispiel eines Burndown-Graphen, Quelle: Eigene Darstellung.

[69] Vgl. Herrmann/Knauss/Weißbach (2013), S. 84 f.

[70] Vgl. Herrmann/Knauss/Weißbach (2013), S. 85 f.

5 MODELLE UND DIAGRAMME

Generell eignen sich Modelle und Diagramme, wenn bei Entwicklungsprojekten mehrere Disziplinen beteiligt sind. Auch Spezifikationen sind in Form von Modellen darstellbar. Vor allem diese Modelle helfen zur besseren Interpretation des zu entwickelnden Systems und reduzieren die wahrgenommene Komplexität. Bei der Nutzung dieser Modelle sind zentrale Systemmodelle anhand der Anforderungen zu erstellen, die später von allen Projektmitgliedern weiterverwendet und bearbeitet werden können.[71] Wie schon in den Kapiteln 4.1.2 und 4.2.2 angesprochen, gibt es verschiedene Modelle respektive Diagramme, welche beim RE und RM ihre Verwendung finden.

5.1 Das Kano-Modell

Beim Ermitteln und Einteilen von Anforderungen (siehe Kapitel 4.1.2) hilft das Kano-Modell. Es kann vorkommen, dass Auftraggeber und Auftragnehmern keine Vorstellung haben, welche Anforderungen sie explizit fordern werden und welche Anforderungen vorausgesetzt sind. Es wird von *bewussten, unbewussten* und *unerwarteten* Anforderungen gesprochen. Auftraggeber und Auftragnehmern sollten gemeinsam voneinander lernen, da nicht einer alleine alle Anforderungen kennt. Nur gemeinsam kann die gesamte Sicht der Anforderungen erhoben werden. Um diese Unterschiede der Anforderungen aufzuzeigen, ist von Professor Noriaki Kano in den 1980er-Jahren in Bezug auf die Kundenzufriedenheit das Kano-Modell entwickelt worden. Wie in Abb. 10 zu sehen, zeigt das Kano-Modell an der Y-Achse die Zufriedenheit, wobei im Gegensatz die gegenüberliegenden Werte die Unzufriedenheit der Kunden abbilden. Die X-Achse spiegelt den Erfüllungsgrad wider. Im rechten Teil der Achse sind die Anforderungen vollständig und im linken Teil der Achse völlig unzureichend erfüllt.[72]

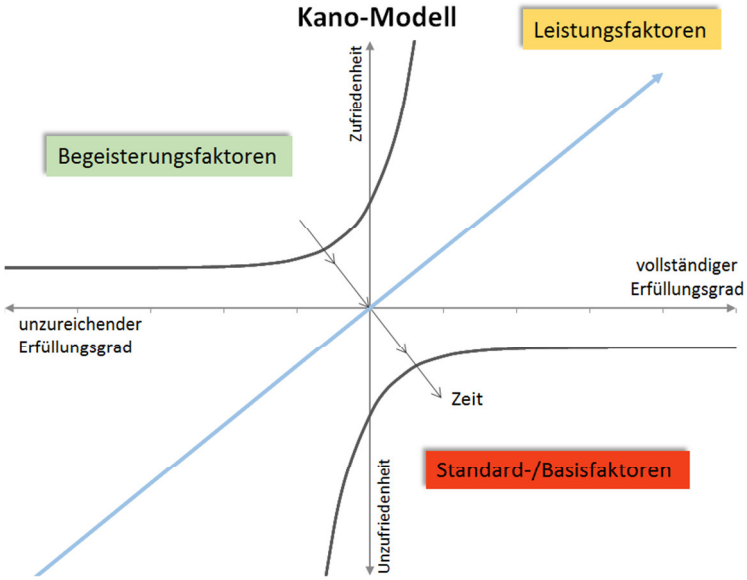


Abb. 10: Das Kano-Modell, Quelle: Eigene Darstellung.

[71] Vgl. Eigner/Roubanov/Zafirov (2014), S. 80 ff.

[72] Vgl. Hruschka (2014), S. 250 f.

Dabei entstehen folgende Gruppen von Anforderungen:[73]

- Standard-/Basisfaktoren
  Drücken die unbewussten Anforderungen aus, welche vorausgesetzt werden.
- Leistungsfaktoren
  Sind Anforderungen, die explizit gefordert werden (bewusste Anforderungen).
- Begeisterungsfaktoren
  Erhöhen die Kundenzufriedenheit, da es Anforderungen sind, die nicht erwartet wurden.

Die Schwierigkeit ist, dass *Standard-/Basisfaktoren* vom Auftraggeber vorausgesetzt werden. Das heißt auch, dass er sie, ohne sie zu erwähnen, erwartet (unbewusste Anforderungen). Werden diese Anforderungen nicht geliefert, ist der Auftraggeber völlig unzufrieden. Im Gegensatz dazu ist feststellbar, dass bei Erfüllung aller unbewussten Anforderungen auch keine übermäßige Zufriedenheit aufkommt, da der Auftraggeber die Erfüllung als selbstverständlich sieht. *Leistungsfaktoren* sind die explizit angesprochenen und geforderten Anforderungen, für die Kunden respektive Auftraggeber auch mehr bezahlen würden. Der lineare Zusammenhang zwischen Erfüllung und Zufriedenheit zeigt eindeutig – je mehr davon erfüllt wird, desto glücklicher wird der Auftraggeber sein. Wird es auf ein Entwicklungsprojekt bezogen, kämpft das Team meist mit Standard-/Basisfaktoren. Wenn nun aber keine Leistungsfaktoren erfüllt werden, wird der Auftraggeber dies nicht gut heißen. Im Bereich der *Begeisterungsfaktoren* besteht die Möglichkeit den Auftraggeber zu überraschen. Diese Anforderungen wurden nicht gefordert. Bekommt der Auftraggeber aber ein zusätzliches, nicht gefordertes Feature, ist er umso glücklicher. Als letzter Faktor im Kano-Modell verändert die Zeit die Anforderungen. Jede Begeisterungsanforderung wird über die Zeit hinweg schwächer und somit zu Leistungsanforderung herabgestuft. Des Weiteren werden Leistungsanforderungen später vorausgesetzt und gehen daher zum Standard über.[74]

5.2 Zielmodelle

Viele Tätigkeiten im Bereich des RE setzen sich mit den Vorstellungen der Stakeholder und den zu entwickelnden Zielen auseinander. Der Aufwand für die Berücksichtigung von Zielen fällt im Gegenzug zum positiven Effekt auf Anforderungen gering aus. Im Allgemeinen wird ein Ziel, als vom Stakeholder gefordertes Merkmal verstanden, das beispielsweise im System implementiert werden soll. Zusätzlich werden Ziele zur Verfeinerung des Systems angewandt. Die Dokumentation der Ziele kann entweder textuell (in natürlicher Sprache) oder in sogenannten Zielmodellen stattfinden. Zielmodelle sind einfache *Und-Oder-Bäume*, welche unter den Zielen die hierarchischen Auflösungen aufzeigen. Die hierarchische Reihenfolge findet von oben nach unten statt. Der Unterschied zwischen einer UND oder eine ODER Verbindung von Anforderungen ist anhand der Linienführung (siehe Abb. 11) ersichtlich. Natürlich können in einem Modell beide Varianten vorkommen.[75]

[73] Vgl. Hruschka (2014), S. 251.

[74] Vgl. Hruschka (2014), S. 252.

[75] Vgl. Pohl/Rupp (2015), S. 67 f.

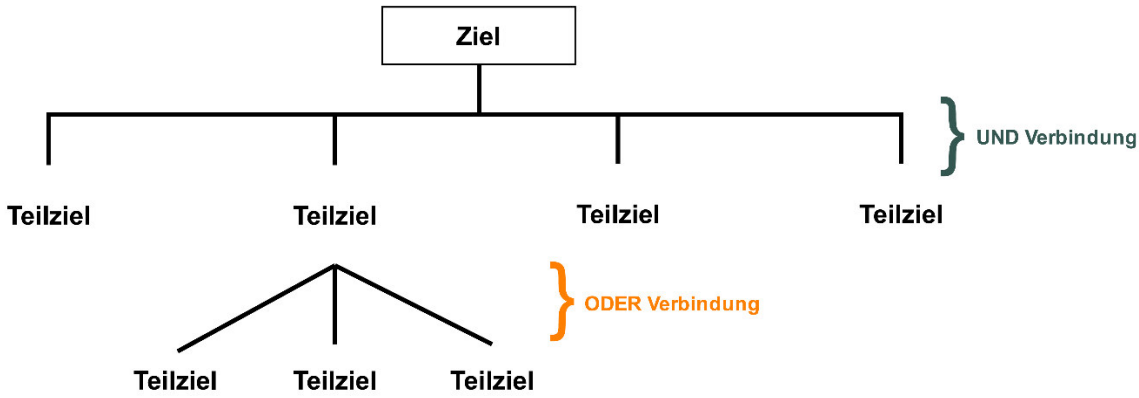


Abb. 11: Allgemeiner Und-Oder-Baum eines Zielmodells, Quelle: Eigene Darstellung.

5.3 Modellierungssprachen zur Systembeschreibung

In der modellbasierten Systementwicklung gibt es eigene Modellierungssprachen, um Systeme darzustellen und zu beschreiben. Eine davon ist die Sprache SysML. Basierend auf der UML, die aus dem Software-Entwicklungsbereich stammt, sind gewisse Diagramme von UML behalten und spezifische SysML Diagramme hinzugefügt worden, welche für die Beschreibung von Systemen erforderlich sind. Variablen, die nur in der Softwareentwicklung ihre Verwendung finden, wie beispielsweise Klassen und softwarespezifische Komponenten, wurden entfernt.[76] Zusätzlich sind bei SysML die Diagramme in drei Bereiche eingeteilt, die folgende Gebiete beinhalten:[77]

- *Strukturdiagramme*
  Dieser Bereich beinhaltet Blockdiagramme, Parameterdiagramme und Paketdiagramme.
- *Anforderungsdiagramme*
- *Verhaltensdiagramme*
  Dieser Bereich beinhaltet Anwendungsfalldiagramme (sogenannte Use-Case-Diagrams), Zustandsdiagramme, Aktivitätsdiagramme und Sequenzdiagramme.

In SysML wird des Weiteren zwischen dem Modell und der Sicht unterschieden. Von außen sind meist nur das Modell und der dazugehörige Name sichtbar (Sichtbereich). Die einzelnen spezifischen Daten dazu, wie z.B. Größen, hinterlegte Daten oder Beschreibungen, sind im Modell gespeichert (Modellinformationen).[78]

In den folgenden Kapiteln wird ausschließlich auf die für den Praxisteil relevanten Verhaltensdiagramme, insbesondere Sequenzdiagramme, Kontextdiagramme und Use-Case-Diagramme eingegangen. Struktur- und Anforderungsdiagramme werden somit nicht weiter betrachtet.

[76] Vgl. Weilkiens/Lamm/Roth/Walker (2016), S. 315 f.

[77] Vgl. Weilkiens/Lamm/Roth/Walker (2016), S. 316.

[78] Vgl. Weilkiens/Lamm/Roth/Walker (2016), S. 317 f.

5.4 Sequenzdiagramme

Bei Sequenzdiagrammen wird mit der Modellierungssprache UML gearbeitet. Dabei wird der zeitliche Verlauf der Kommunikation und/oder der Interaktion zwischen Komponenten in einem System abgebildet (siehe Abb. 12). Die Komponenten und/oder Akteure sind in horizontaler Richtung nacheinander gereiht. Nach unten in vertikaler Richtung verläuft die Zeit. Nun wird mittels Pfeilen der zeitliche Verlauf oder der Datenfluss eingezeichnet. Die hiermit abgebildete Sequenz ist immer nur eine bestimmte Interaktion. Das heißt, dass keine vollständige Spezifizierung möglich ist. Nützlich sind Sequenzdiagramme aber dadurch, dass wichtige oder schwierige Abläufe genau darstellbar sind. Sie helfen das Systemverhalten zu dokumentieren. Durch die mittlerweile geschaffene Möglichkeit von Schleifen oder alternativen Abläufen können Sequenzdiagramme sehr schnell, sehr groß und unübersichtlich werden. Daher sollen diese nicht zu aufwändig gestaltet werden und zur Diskussionsgrundlage bei Besprechungen dienen.[79] Auch Testfälle sind in Sequenzdiagrammen darstellbar.[80]

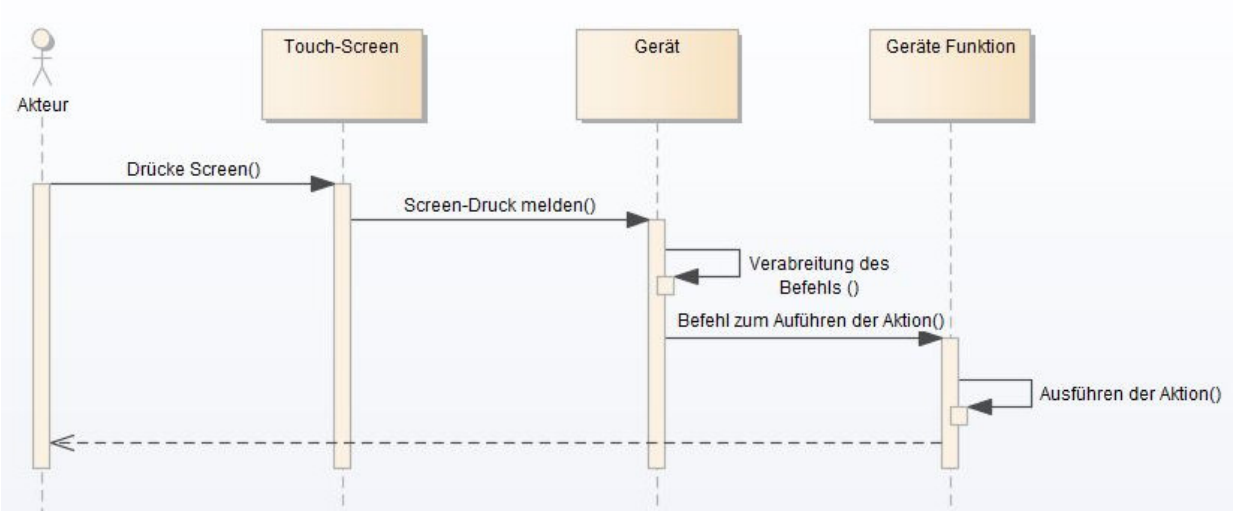


Abb. 12: Einfaches Beispiel eines Sequenzdiagramms, Quelle: Eigene Darstellung.

5.5 Kontextabgrenzung

Wie schon im Kapitel 4.1.2 erwähnt, ist es essentiell, das System mit einer Kontextabgrenzung darzustellen. Dies kann als erste Architekturarbeit verstanden werden, um sicherzustellen, welche Teile des zu entwickelnden Systems mit anderen Komponenten oder Schnittstellen in Verbindung stehen. Wichtig ist, dass das System an sich als Black-Box betrachtet wird und nur Verbindungen zu anderen Teilen abbildet. In SysML ist solch ein Kontextdiagramm nicht explizit als Möglichkeit auswählbar. Trotzdem kann mittels Blockdiagramm das System-Modell mit den Verbindungen zu den Akteuren und anderen Systemen dargestellt werden, welches als Kontextabgrenzung fungiert.[81]

[79] Vgl. Alt (2012), S. 53 f.

[80] Vgl. Alt (2012), S. 119 ff.

[81] Vgl. Alt (2012), S. 110.

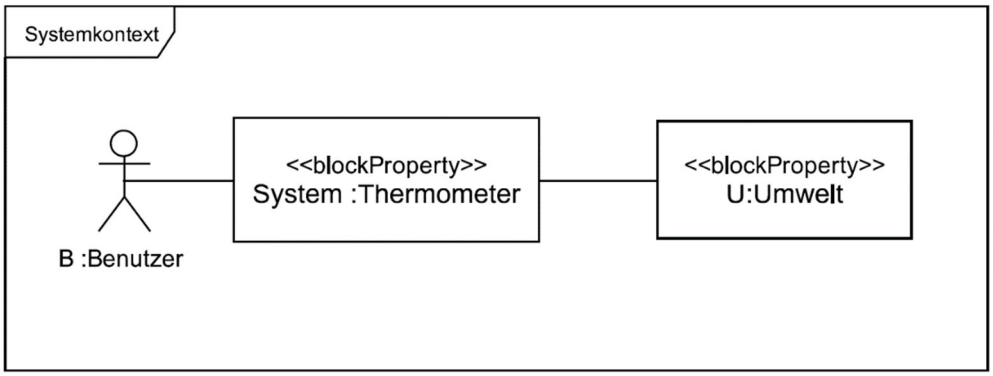


Abb. 13: Grundlegendes System-Kontextdiagramm. Quelle: In Anlehnung an Alt (2012), S. 110.

Wie in Abb. 13 zu sehen ist, soll das zu entwickelnde System in der Mitte dargestellt werden. Mit Linien zu den anderen Objekten wird das Eingreifen oder die Schnittstelle angedeutet. Das zu entwickelnde System sollte schon in dieser Phase einen eindeutigen Namen aufweisen, damit es später zu keinen Fehlinterpretationen kommt. Weiterführend ist die Darstellung als Wirkkette möglich, die mit sogenannten Flow-Ports auch eine Signalrichtung im Modell spezifizieren. Die hieraus gewonnenen Kenntnisse des Modells werden danach mit den bestehenden Anforderungen abgeglichen. Es kann daher vorkommen, dass auf Grund der Kenntnisse eine Anpassung der derzeitigen Anforderungen stattfindet.[82]

Um das System weiter zu spezifizieren, empfiehlt es sich das System als *technisches Wirkkettenmodell* weiter zu entwickeln. Bis zu diesem Zeitpunkt war das Modell als eine Black-Box betrachtet worden. Mit der Ausführung in ein technisches Wirkkettenmodell soll schon anhand der Anforderungen erste technische Lösungen erarbeitet und als grundlegende Architektur so abgebildet und dokumentiert werden. Als grundlegendes einfaches Beispiel ist dies in Abb. 14 dargestellt. Im Ablauf der Wirkkette ist nun der Ablauf als grundlegende technische Realisierung dargestellt. Je nach Lösungsart können mechanische, elektronische oder auch Software-Eigenschaften in Verbindung abgebildet werden. In diesem Stadium werden die Schnittstellen der einzelnen Blöcke auch schon richtig bezeichnet wie z.B. Register für einen Softwareschnittstelle.[83]

[82] Vgl. Alt (2012), S. 110 f.

[83] Vgl. Alt (2012), S. 112 ff.

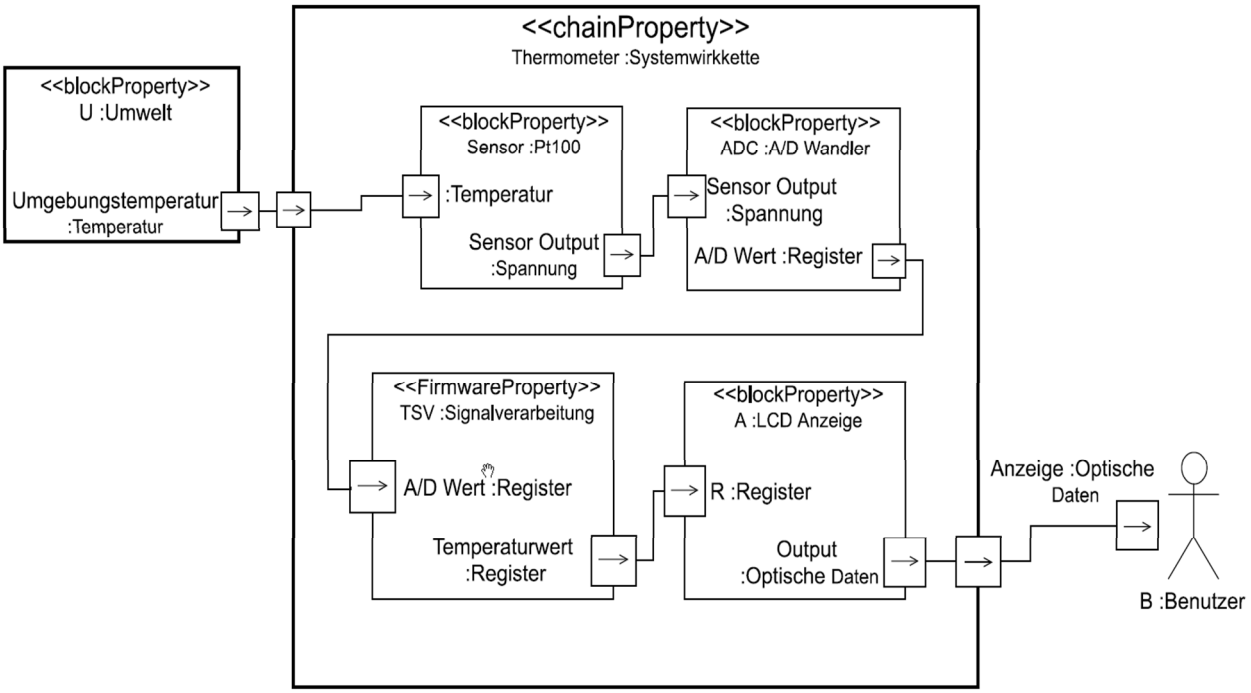


Abb. 14: Einfaches technische Wirkkettenmodell, Quelle: In Anlehnung an Alt (2012), S. 113.

5.6 Use-Case-Diagramme

Ein Use-Case-Diagramm (UCD) bzw. System-Use-Case-Diagramm (SUCD) wird dazu verwendet, um die Interaktion von Systemen mit einem Anwender darzustellen. Der Anwender führt hierbei eine Aktion mit dem System aus, das ihm ein gewünschtes Ergebnis bringt. Zusätzlich wird der Use-Case (UC) noch textuell beschrieben, um die Interaktion besser zu beschreiben. Ein SUCD bildet verschiedene UC untereinander inklusive den Akteuren ab, die das System erfüllen kann.[84]

5.6.1 System-Use-Case-Diagramm

Um ein SUCD und dessen Anwendungsfälle (Schritte) zu erstellen, sind folgende Fragen hilfreich:[85]“

- Welches System unterstützt diesen Schritt im Moment?
- Wie unterstützt dieses System den Schritt?
- Soll dieses System auch in Zukunft für die Unterstützung des Schritts genützt werden oder nicht?
- Existiert bereits ein anderer System-Use-Case, der diesen Schritt unterstützt?
    - Falls ja, wird der Schritt diesem System-Use-Case zugeordnet.
    - Falls nein, wird ein neuer System-Use-Case aufgenommen.“

Die Fragen sollen als Grundlage für das SUCD dienen, welches in weiterer Folge immer weiter verfeinert wird. Der Inhalt des System-Use-Case (SUC) bezieht sich auf das nach außen sichtbare Systemverhalten. Deshalb ist der SUC ein Teil der Anforderung an das System. Wie in Abb. 15 ersichtlich, bietet das SUCD eine Übersicht über die Hauptfunktionen des Systems. Mit dieser zusätzlich gewonnenen Klarheit

[84] Vgl. Rupp (2014), S. 189.

[85] Vgl. Rupp (2014), S. 190.

sind weitere Detaillierungen der einzelnen UC einfacher möglich. Ein weiterer Vorteil des SUCD ist, dass auch bei den Anwendern die Akzeptanz sehr hoch ist, da ohne große Erklärungen die Anwendungen ersichtlich sind.[86]

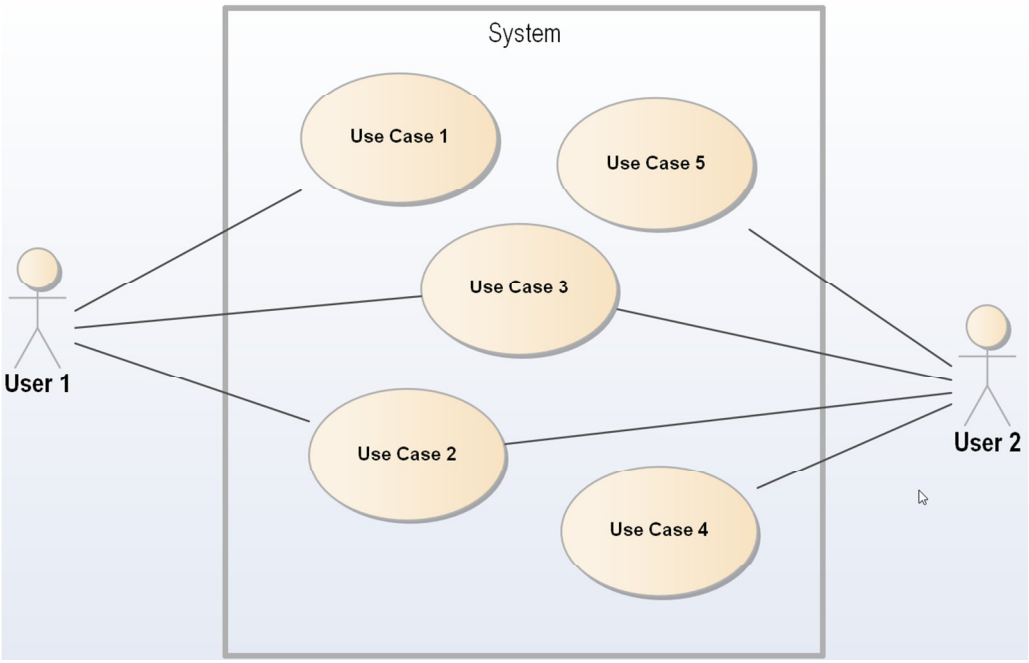


Abb. 15: Beispiel eines SUCD, Quelle: Eigene Darstellung.

5.6.2 Beschreibung von Use-Cases

Zusätzlich zu den einzelnen UCD ist es wichtig den jeweiligen UC genauer zu beschreiben, da das Diagramm nur die Übersicht zur Funktionalität (funktionale Anforderungen) gibt. Durch eine vorgegebene Notation können auch Informationen eingeholt werden. Des Weiteren hilft die tabellarische Ausführung zur Übersicht und erleichtert das Verständnis des UCD. Ziel sollte eine knappe Beschreibung der Anwendung sein. Durch die Einbindung von Text ist es auch möglich, nicht funktionale Anforderungen in den UC einzubringen.[87] In Tab. 1 sind Kategorie-Beispiele einer UC-Beschreibung angeführt. Die Tabelle ist in drei Spalten aufgeteilt. In der einen Spalte sind die vorgegebenen Kategorien, die beantwortet werden müssen, angeführt. In der zweiten Spalte *Beschreibung* sind die Erklärungen der Kategorien aufgelistet. In der dritten Spalte ist ein kurzes Beispiel angeführt, wie die Beschreibung beginnen kann. Ein vollständiges Beispiel ist aus Platzgründen nicht möglich, da sich Anforderungen über mehrere Seiten erstrecken können.

Die Kategorien wie z.B. *Nachbedingung, Prioritäten, Ergebnis* u.v.m. sind möglich und die Beschreibung ist dementsprechend erweiterbar.[88]

[86] Vgl. Rupp (2014), S. 190 ff.

[87] Vgl. Rupp (2014), 192 ff.

[88] Vgl. Pohl/Rupp (2015), S. 72 f.

Kategorie	Beschreibung	Beispiel
Name	Name des UC	Temperaturregelverhalten
Kurzbeschreibung	Kurze Beschreibung des UC	Das Temperaturregelverhalten zur Probentemperierung, soll im Temperaturbereich…
Akteure	Beteiligte Akteure des UC	Produktspezialist, Kunde
Vorbedingung	Voraussetzungen für den UC	Ein funktionierendes Messgerät in Kombination mit der Software wird benötigt. Des Weiteren wird folgendes Zubehör verwendet, um…
Fachlicher Auslöser	Grund des UC	Zur Viskositätsbestimmung muss eine stabile Probentemperatur herrschen, da…
Hauptszenario	Möglicher Ablauf des UC	Um das Temperaturregelverhalten zu ermitteln werden folgende Temperatursprünge vorgegeben:…
…	Weitere Kategorien sind möglich	

Tab. 1: Use-Case-Beschreibung mit Erklärung, Quelle: Eigene Darstellung.

5.6.3 Elemente im Use-Case-Diagramm

Um ein UCD darzustellen, sind verschiedene Modellelemente möglich. In Abb. 16 werden die wichtigsten Objekte gezeigt, die für ein UCD benötigt werden. Das ovale Element stellt einem UC dar, welcher in der Mitte den Namen des UC angibt. Systeme werden in rechteckigen Kästchen mit der Bezeichnung des Systems angegeben. Dabei stellt der Rand die Systemgrenze dar. Im Kästchen werden die UC inklusive Namen platziert, die das System erfüllen soll. Akteure die mit dem System interagieren, werden als Strichmännchen außerhalb des Systems angegeben. Mit Linien wird die Interaktion von Akteuren und den dazugehörigen UC miteinander dargestellt (siehe Abb. 17).[89]

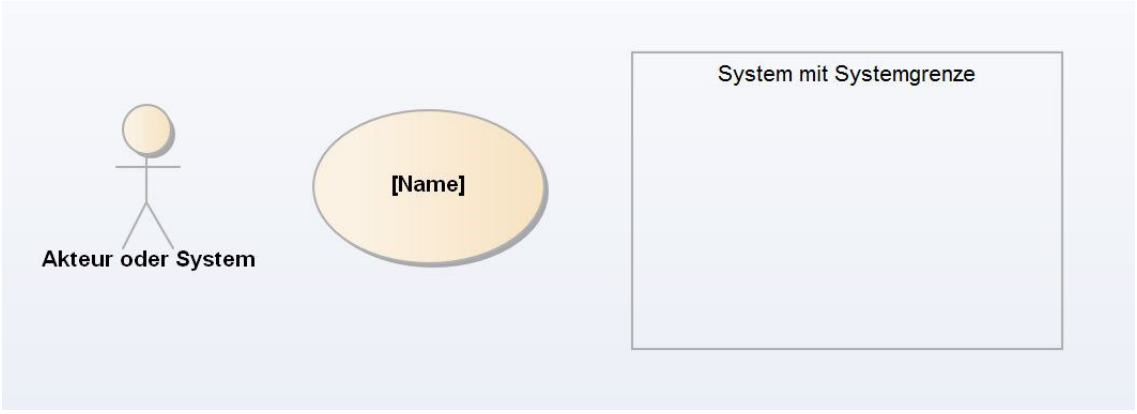


Abb. 16: Objekte im Use-Case Diagramm, Quelle: Eigene Darstellung.

[89] Vgl. Pohl/Rupp (2015), S. 69 f.

Abb. 17 zeigt, dass verschiedene UC in einem System in einer Beziehungen stehen können. Der Begriff <<extend>> sagt aus, dass der UC-A eine Erweiterung eines bestimmten Punkts von UC-B ist. Wobei hingegen <<include>> bedeutet, dass die Interaktionsfolge des UC-D auch in UC-C inkludiert ist.[90]

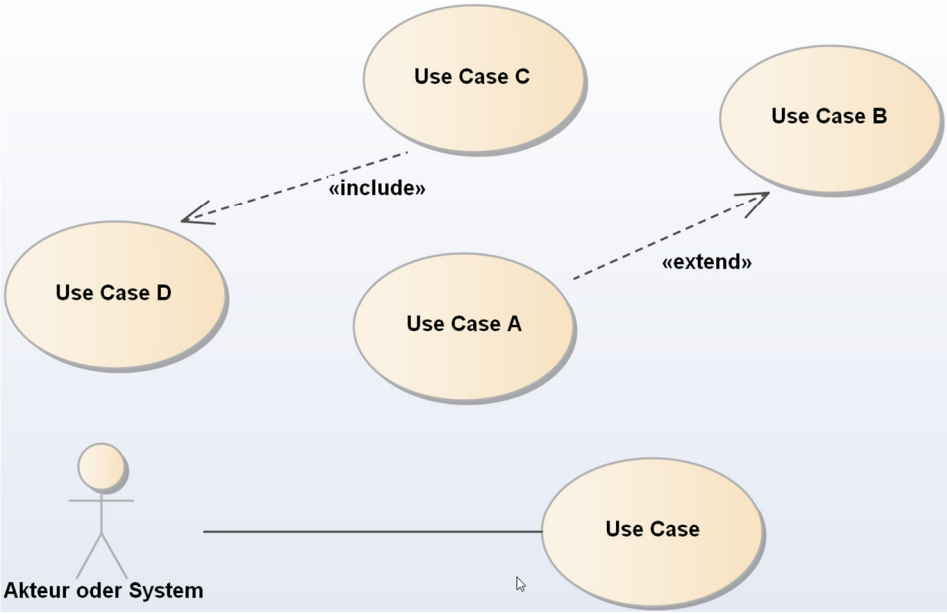


Abb. 17: Beziehungen in Use-Case-Diagrammen, Quelle: Eigene Darstellung.

5.6.4 Anforderungen anhand eines Use-Case ableiten

Um Anforderungen aus einem UC abzuleiten, kann eine sogenannte UC-Analyse durchgeführt werden. Dies ist eine Vorgehensweise, die funktionale Anforderungen aus dem UC ableitet und diese verfeinert. Wenn ein UC definiert und beschrieben ist, kann die Analyse gestartet werden. Dabei handelt es sich um vier Schritte. Schritt 1 ist die grundlegende Definition und Beschreibung des UC, die als Voraussetzung der Analyse gegeben sein muss. Hier sind auch die Akteure, die das System benutzen, angegeben. In den weiteren Schritten werden nun die UC verfeinert. Ziel ist es eine genauere Beschreibung des UC zu erhalten.[91] Im Schritt 2 sind Abläufe zu definieren und im Schritt 3 sind Zustände und Verhalten zu ermitteln. Als letztes werden im Schritt 4 die Anforderungen in natürlicher Sprache dokumentiert. Sind die Anforderungen noch nicht genau genug beschrieben, kann die Verfeinerung in Schleifen wiederholt werden. Als zusätzlichen Schritt 5 kann zu diesem Zeitpunkt auch der Testfall zu den ermittelten Anforderungen ermittelt und dokumentiert werden. Der Vorteil der frühen Testdefinition ist, dass bei einer nicht möglichen Testdefinition die Anforderung noch überarbeitet werden muss.[92]

[90] Vgl. Pohl/Rupp (2015), S. 70.

[91] Als benötigte Methoden kann auf das Kapitel 4 zurückgegriffen werden.

[92] Vgl. Rupp (2014), S. 46.

6 DAS V-MODELL

Als verbreitetes Vorgehensmodell bei einer mechatronischen Produktentwicklung hat sich das sogenannte V-Modell etabliert. Dieses Modell bietet in der Form eines V eine Vorgehensweise, welche die Entwicklungstätigkeiten und die Testaktivitäten je nach Verfeinerungsgrad von oben nach unten gegenüberstellt.[93] In Abb. 18 ist das Schema des V-Modells ersichtlich. Die Anforderungen sollen als Start-Input im Modell vorhanden sein und schlussendlich ein fertiges Produkt ergeben. Die Anforderungen dienen als Bezugsquelle für alle abgeleiteten Spezifikationen, Tätigkeiten und Tests nach denen das fertige Produkt später gemessen wird. Beim Systementwurf wird die Gesamtfunktion in einzelne Teilfunktionen zerlegt, damit die resultierenden Lösungselemente einem geeigneten Wirkprinzip zugeordnet werden können. Auf dieser Basis wird im domänenspezifischen (fachspezifischen) Entwurf die Lösung konkretisiert. Hierbei werden detaillierte Auslegungen und auch Berechnungen durchgeführt und das Produkt somit entwickelt. Da die Domänen meist unabhängig voneinander arbeiten, werden die Ergebnisse in der Systemintegration vereint und getestet. Während aller Phasen sind zur Abbildung und Untersuchung der Systemeigenschaften rechnergestützte Modelle im Einsatz (Modellbildung und Analyse).[94] Auch Modelle in SysML oder UML sind möglich.[95]

Wichtig ist, dass die Reihenfolge der Abarbeitung der im V-Modell vorhandenen Prozesse keine unmittelbare zeitliche Linie darstellt. Damit ist gemeint, dass Tests nicht erst am Ende der ganzen Entwicklung stattfinden sollen. Ganz im Gegenteil, Tests müssen frühestmöglich nach der Implementierung durchgeführt werden. Im Modell ist nur der Testlevel angegeben, der die zugeordnete Anforderung prüft. Die Testergebnisse liefern somit wieder neuen Input für die weitere Entwicklung. Zusätzlich wird im V-Modell erkannt, dass sobald eine Spezifikation erstellt ist, es auch möglich sein muss, einen Test dazu zu definieren.[96]

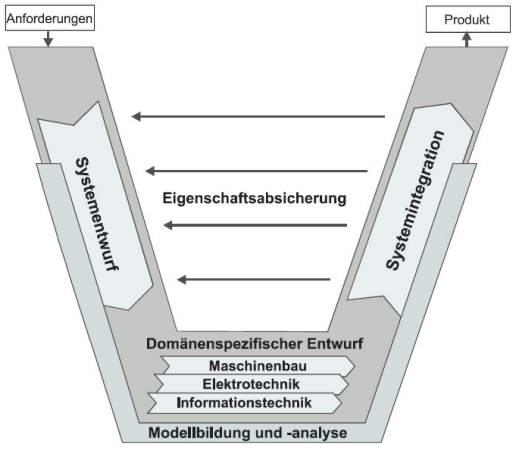


Abb. 18: Das V-Modell. Quelle: VDI 2206 (2004), S. 29.

[93] Vgl. Alt (2012), S. 85.

[94] Vgl. VDI 2206 (2004), S. 29.

[95] Vgl. Alt (2012), S. 85 f.

[96] Vgl. Alt (2012), S. 85 f.

Je nach Entwicklungsaufgabe ist es mit mehreren V-Modellen möglich, verschiedene Reifegrade des Produkts zu realisieren. Beispielsweise kann nach einem Durchlauf des V-Modells (V-Modell-Zyklus) ein Labormuster, nach einem weiteren Durchlauf ein Funktionsmuster bis hin zu einem Vorserien- oder schlussendlich dem Endprodukt erreicht werden. Je nach Projekt ist dies infolgedessen erweiterbar. Ob dementsprechende Zyklen stattfinden, hängt von der Komplexität, Art und Entwicklungsaufgabe ab.[97]

Die schon kurz angedeuteten vordefinierten Prozesselemente *Systementwurf, domänenspezifischer Entwurf, Systemintegration, Modellbildung und Analyse* sowie *Eigenschaftenabsicherung* beinhalten spezifische Tätigkeiten, auf die in den folgenden Unterkapiteln noch genauer eingegangen wird.

6.1 Systementwurf

Beim Systementwurf wird damit begonnen anhand der Anforderungen eine Abstraktion und Vorfixierung der Tätigkeiten durchzuführen, die den Lösungsraum dennoch offen halten. Als Ziel soll das Wesentlichste und Allgemeinste der Problemstellung bzw. Problemspezifikationen übrigbleiben. Als Funktionsstruktur werden die Problemspezifikationen in eine Gesamtfunktion übergeleitet. Bei mechatronischen Entwicklungen ist die Aufgabe meist zu komplex, um sofort aus einer Gesamtfunktion die technische Realisierung abzuleiten, daher wird die Gesamtfunktion in Teilfunktionen aufgeteilt. Letztendlich sollen dabei die Details von den Lösungselementen als Wirkprinzipien übrig bleiben. Jedes Lösungselement muss danach einer physikalischen Domäne (Kapitel 6.2) zugeordnet werden. Zum Schluss des Systementwurfs soll ein vollständiges Lösungskonzept mit den logischen und auch physikalischen Wirkungsweisen, die das Produkt beschreiben, zur weiteren Bearbeitung im domänenspezifischen Entwurf vorliegen.[98]

Um nun die Anforderungen in dieser Phase zu dokumentieren und auch abnehmen zu lassen, gibt es Dokumente, die das umfassen – das sogenannte Lasten-, und Pflichtenheft.

6.1.1 Lastenheft

Das Lastenheft beinhaltet die Anforderungen des Kunden bzw. des Auftraggebers. Darin sollen die gesamten Leistungen und Forderungen dokumentiert sein. Es handelt sich um die zentralen Komponenten des zu entwickelnden Systems. Mit diesem Dokument soll eindeutig ersichtlich sein, wofür das Produkt schlussendlich verwendet wird. Auch für diese Dokumentation können schon Modelle oder Grafiken verwendet werden. Der Lösungsansatz bleibt dabei aber offen.[99]

6.1.2 Pflichtenheft

Beim Pflichtenheft handelt es sich um die genaue Dokumentation, wie das Entwicklungsteam die geforderten Anforderungen des Lastenhefts erfüllen möchte. Als weitere Bezeichnung für das Pflichtenheft können auch die Begriffe Gesamtspezifikation, Fachfeinspezifikation oder Fachspezifikation verwendet

[97] Vgl. VDI 2206 (2004), S. 30 f.

[98] Vgl. VDI 2206 (2004), S. 32 ff.

[99] Vgl. Eigner/Roubanov/Zafirov (2014), S. 60.

werden, die die Zielsetzung beschreiben. Generell wird als Grundlage hierfür immer die Kundenspezifikation des Lastenhefts verwendet. [100]

Folgende Begriffe sollen im Pflichtenheft vorkommen:[101]“

- Ist-Zustand,
- Projektziel,
- Systemeinsatz und die Systemumgebung,
- funktionale Anforderungen,
- technische Anforderungen,
- Abnahmekriterien,
- Glossar zur Erläuterung der Fachbegriffe.“

6.2 Domänenspezifischer Entwurf

In dieser Phase wird je nach spezifischer Domäne respektive nach spezifischen Fachbereich anhand der etablierten Entwicklungsmethodik, den Denkweisen, Begriffswelten und auch Erfahrungen entwickelt.[102] Wie in Abb. 18 ersichtlich, sind das in der mechatronischen Systementwicklung die Bereiche Maschinenbau, Elektrotechnik und die Informationstechnik.

6.3 Systemintegration

In der Phase der Systemintegration werden die Entwicklungen, wie z.B. Funktionen, Komponenten oder Teilsysteme der verschiedenen Domänen, zusammengeführt. Dabei kann es sich je nach Reifegrad um Funktionsmuster, Vorserien oder das schlussendliche Produkt handeln. Um den Integrationsgrad so hoch wie möglich zu halten, ist es schon vor dem domänenspezifischen Entwurf notwendig die Kompatibilität der einzelnen Wirkprinzipien zu prüfen und die Schnittstellen vorzubereiten. Die noch vorhandenen Unverträglichkeiten der Teillösungen sollen hier erkannt und die optimale Gesamtlösung gefunden werden.[103]

6.4 Modellbildung und Analyse

Bei komplexen Produktentwicklungen wird das V-Modell als Vorgehensweise eingesetzt. Um die Komplexität überhaupt zu überblicken und zu beherrschen sind Modelle mit domänenübergreifenden Charakter notwendig. Zur Modellierung und Analyse des Systems anhand der rechnergestützten Modelle können Aspekte wie Dynamik, Schwingungen, Erwärmungen u.v.m. betrachtet werden.[104]

---

[100] Vgl. Eigner/Roubanov/Zafirov (2014), S. 60 f

[101] Vgl. Eigner/Roubanov/Zafirov (2014), S. 61.

[102] Vgl. VDI 2206 (2004), S. 35.

[103] Vgl. VDI 2206 (2004), S. 35 ff.

[104] Vgl. VDI 2206 (2004), S. 35.

6.5 Eigenschaftenabsicherung

Mit der Eigenschaftenabsicherung werden Lösungsvarianten inklusive deren Eigenschaften mit den vorher definierten Anforderungen überprüft. Dabei wird von der *Verifikation* und der *Validierung* gesprochen, die je nach Stufe die geforderten Systemeigenschaften überprüfen. Anhand der erfolgten Bewertung können Lösungsvarianten untereinander oder mit der idealisierten Lösung verglichen werden. Solche Bewertungen sind virtuell, real oder in Kombination durchführbar, um zu einem sicheren Ergebnis zu kommen. Virtuell geschieht die Untersuchung anhand der Modelle, und bei realen Überprüfungen muss ein physikalisch aufgebautes System zur Verfügung stehen. Hilfreich sind dabei einerseits die Methoden des Hardware-in-the-Loop (HIL), bei dem echte Komponenten und Simulationsmodelle gemeinsam das Gesamtsystem in Echtzeit simulieren. Die Hardware kann somit auf ihre Tauglichkeit überprüft werden. Andererseits ist auch die Methode des Software-in-the-Loop (SIL) möglich, der mit dem modellierten Prozess als Systemmodell gemeinsam mit der Simulationsumgebung interagiert. Dabei können dynamische Prozesse, wie z.B. Regelstrecken, untersucht werden.[105]

6.5.1 Verifikation

Bei der Verifikation handelt es sich um die Überprüfung von technischen Systemen. Verglichen wird, ob das realisierte System mit der Spezifikation übereinstimmt. Dabei wird die Frage beantwortet, ob das Produkt korrekt entwickelt wurde.[106] Zusammengefasst werden dabei Tätigkeiten durchgeführt, die sicherstellen, dass das Entwicklungsergebnis auch den Entwicklungsaufgaben und dessen Anforderungen entspricht.[107]

6.5.2 Validierung

Ursprünglich ist unter Validierung zu verstehen, ob die Testresultate auch wirklich das erfassen, was durch die Tests bestimmt werden soll. Wird das auf technische Produkte umgelegt, handelt es sich um die Prüfung des Einsatzzwecks. Es wird dabei die Erwartungshaltung des Fachexperten und die des Anwenders geprüft. Ziel ist die Beantwortung der Frage, ob das Produkt richtig entwickelt wurde.[108] Die Produkte und die Dienstleistungen werden auf ihren beabsichtigten Gebrauch hin überprüft. Es sagt aus, ob die Erfüllung der Anwendung der Anforderung entspricht.[109]

[105] Vgl. VDI 2206 (2004), S. 38 ff.

[106] Vgl. VDI 2206 (2004), S. 38.

[107] Vgl. EN ISO 9001 (2015), S. 54.

[108] Vgl. VDI 2206, S. 39.

[109] Vgl. EN ISO 9001 (2015), S. 54.

7 IST-STAND-ERHEBUNG IN DER ANTON PAAR GMBH

Bei der Firma Anton Paar GmbH wird nach dem standardisierten PEP ein Produkt entwickelt. Hierbei werden verschiedene Phasen durschritten, wobei für jede Phase verpflichtende und optionale Dokumente vorhanden sein müssen. Folgende Phasen und Dokumente sind aus technischer Sicht im PEP vorhanden[110]:

- Projektstart
  In dieser Phase soll die Zusammenfassung der später folgenden Rahmenvorgabe erstellt werden, um den Kundenwunsch und die grundlegende Idee des zu erstellenden Produkts grob fassen zu können.
- Projektauftragsphase
  Hier ist die Rahmenvorgabe als Dokument fertigzustellen, bei der das physikalische Prinzip entschieden und der Kundenwunsch ausformuliert wird. Die Detailierung soll so weit reichen, dass der komplette Kostenrahmen des Projekts erstellt werden kann.
- Spezifikationsphase
  Das Ziel dieser Phase ist es, eine technische Spezifikation zu erreichen, die via Simulationen und/oder Funktionsmuster ermittelt wird. Als verpflichtende Dokumente sind ein Spezifikationsdokument, Verifikationsplan und Validierungsplan zu erstellen.
- Entwicklungsphase
  In der Entwicklungsphase ist es das Ziel seriennahe Prototypen zu erstellen und deren Funktion zu prüfen. Zu diesem Zeitpunkt sind schon Checklisten mit Abarbeitung von Nachweisen gefordert.
- Überleitungsphase
  Hier wird das Produkt zu Serienreife geführt und alle Tests abgeschlossen, damit das Produkt zur Auslieferung an Kunden bereit steht.
- Marktbeobachtungsphase
  Als letzte Phase ist die Marktbeobachtung zu nennen, in der über einen gewissen Zeitraum das Produkt am Markt betrachtet wird. Schlussendlich soll das Projekt zu diesem Zeitpunkt abgeschlossen werden.

Bei der Ist-Stand-Erhebung werden als erstes die derzeit geforderten Dokumente *Rahmenvorgabe*, *Spezifikation*, *Verifikationsplan* und *Validierungsplan* gesichtet und auf die aktuelle Verwendung von RE und RM hin untersucht. Danach werden in einem Workshop vier Produktentwicklungsprojekte gesichtet und erhoben wie in diesen Projekten das RE, das RM, die Verwendung von Polarion und vorhandene Modelle ihren Einsatz gefunden haben. Von der Research and Development (R&D) Abteilung werden des Weiteren Informationen zur Verfügung gestellt, die die aktuelle Softwareverwendung abbildet. Anhand dieser Informationen wird im Kapitel 8 ein Leitfaden als Konzept zur Verwendung von RE und RM für die Produktentwicklung entwickelt.

[110] Auf Dokumente für Marketing- und Markteinführungszwecke wird nicht explizit eingegangen.

7.1 Analyse der bestehenden Dokumente

In diesem Kapitel werden nun die Inhalte der bestehenden Anton Paar GmbH Dokumente im PEP bezüglich des RM und des RE untersucht und beschrieben. Schlussendlich wird ein Fazit über den Ist-Stand gegeben, und Vorschläge über zusätzliche Inhalte für RE und RM eingebracht.

7.1.1 Rahmenvorgabe

Die Rahmenvorgabe spiegelt das im Kapitel 6.1.1 erwähnte Lastenheft wider. Dieses Dokument wird in die Kapitel 1. *Executive Summary*, 2. *Produkt*, 3. *Projekt*, 4. *Markt*, 5. *Wirtschaftliche Bewertung*, 6. *Projektrisikobewertung* und 7. *Ergänzende Informationen* eingeteilt.

In der aktuellen Rahmenvorgabe wird vorgegeben, dass das Produkt aus der Sicht des Kunden beschrieben werden muss. Explizit ist hier in Bezug auf Anforderungen angeführt, dass Eigenschaften wie das Mess- und Funktionsprinzip, sowie das Funktions- und Ausführungsmerkmal und zusätzlich die Qualitätsmerkmale anzugeben sind. Auch die Einsatzmöglichkeiten und Anwendungsbereiche und die angestrebten Unique Selling Propositions (USP) sollen festgelegt werden. Auch ‚Nicht-Ziele' sind zu erwähnen, die zur Festlegung des neuen Produkts und dem Ausschluss von Funktionen dienen sollen. Als zusätzlicher Punkt im Dokument sind auch Kundenanforderungen darzulegen, die entweder als Text angeführt oder zusätzlich mit einem Umfrageergebnis belegt werden können.

7.1.2 Spezifikationsdokument

Dieses Dokument soll das im Kapitel 6.1.2 beschriebene Pflichtenheft darstellen und die Themen aus Sicht der Entwicklungsabteilung beschreiben. Unterteilt wird es in die Kapitel 1. *Executive Summary*, 2. *Produkteigenschaften*, 3. *Preise und Kosten*, 4. *Anwendung* und 5. *Produktpositionierung*.

Bezüglich Anforderungen an das zukünftige Produkt sind das Mess- bzw. Funktionsprinzip sowie Funktions- und Ausführungsmerkmale anzuführen. Als expliziter Punkt im Dokument gilt es technische Spezifikationen zu erreichen, die das Produkt erfüllen muss. Auch Qualitätsmerkmale, USP, Einsatzmöglichkeiten und Anwendungsbereiche und Kundennutzen sind Gegenstand des Spezifikationsdokuments.

7.1.3 Verifikations- und Validierungsplan

Der Verifikationsplan zur Typprüfung beinhaltet grundsätzlich Tests in Form einer Liste, die technische Details überprüfen. Damit sind Normen, mögliche Temperaturbereiche, mechanische Eigenschaften etc. gemeint, die es zu prüfen gilt. Jeder Test muss mit einem Namen, Bereich, Kommentar und einem Status versehen sein. Zusätzlich besteht die Möglichkeit einen Link zu weiteren Dokumenten, die den Test und dessen Ergebnis beschreiben, hinzuzufügen.

Im Validierungsplan sind im aktuellen Stand Abnahmen enthalten, die die technische Sicherheit und technische Spezifikationen prüfen. Hinzu kommen noch Prüfungen im Sinne der Gebrauchstauglichkeit, Handhabung und Feldtests. Bei jedem Test sind Spezifikationen, Angaben zum Zeitpunkt und den Personen und eine Prüfbeschreibung inklusive Ergebnis anzuführen.

7.1.4 Fazit der Dokumentenanalyse

Die Rahmenvorgabe und das Spezifikationsdokument sind inhaltlich in vielen Punkten redundant. Da das Spezifikationsdokument von der Rahmenvorgabe abgeleitet werden soll, sollen die Inhalte nach technischer Spezifikation und Kundenanforderung getrennt werden.

In Bezug auf Anforderungen soll sich die Rahmenvorgabe auf Anwendungsfälle, Kundenanforderungen und USP beschränken. Eine Beschreibung der Mess- und Funktionsprinzipien ist nicht zwingend erforderlich, aber wenn diese vorhanden ist, muss diese die Unterstützung des Anwendungsfalls erklären und die technische Lösung offen halten. Beim Spezifikationsdokument soll der Fokus auf die technische Realisierung, der in der Rahmenvorgabe definierten Mess- und Funktionsprinzipien liegen. USP und Kundennutzen sind hier nicht mehr erforderlich. Ein besonderes Augenmerk sollte auf die technischen Umsetzungen inklusive aller Toleranzen zur Erfüllung der Anwendungsfälle, so wie im Kapitel 6.1.2 aufgelistet, gelegt werden. Auch Modelle und Diagramme jeglicher Form, die die Umsetzung erklären, sollen eingebracht werden.

Die Handhabung der Verifikation und Validierung ist nicht eindeutig getrennt. Derzeit ist das Verifikationsdokument eine Art Liste und das Validierungsdokument ist eine Beschreibung von Tests und beinhaltet auch technische Abnahmen, die nicht zu Validierung gehören. Vorgeschlagen wird eine Trennung der Abnahmen in technisch relevante Tests in der Verifikation (z.B. Temperaturtests, CE) und den Kunden betreffende Tests (z.B. Messung von Kundenproben) in der Validierung. Verifikations- und Validierungspläne müssen aber schon in der Entwicklungsphase eindeutige Spezifikationen und Kriterien für eine positive Bewertung inkludieren. Sind beide Pläne in der Produktentwicklung abgeschlossen, führt das zu einem verifizierten und auch validierten Produkt, das den Kundenanforderungen gerecht wird.

Generell fehlt es im PEP an Hilfestellungen, wie Anforderungen ermittelt werden können und in welcher Qualität diese vorhanden sein müssen. Grundsätzlich kann schwer eingeschätzt werden, wann eine Beschreibung ausreichend ist. Hier sind Qualitätskriterien gefordert, die den Autoren helfen ihre Arbeit selbst einzuschätzen. Des Weiteren ist der Umgang mit Anforderungen während den Phasen nicht erklärt. Dafür ist eine Vorgabe notwendig, die eine Übersicht über Anforderungen und deren Bedeutung gibt.

7.2 Bestehende Werkzeuge

In der Firma Anton Paar GmbH sind derzeit zwei verschiedene Software-Werkzeuge (Jira und Polarion) vorhanden, die beim Management von Aufgabenpaketen und Anforderungen zur Verfügung stehen. Zusätzlich gibt es noch eine weitere Möglichkeit Diagramme browserbasierend in der Software Confluence zu erstellen. Die Software Polarion steht für das Management von Anforderungen zur Verfügung. Die Software Jira bietet die Umgebung für eine agile Arbeitsweise und Confluence ist zur Dokumentation bzw. Diagrammerstellung vorhanden. Jira und Confluence (zur Dokumentation) sind schon seit vielen Jahren im Einsatz. Polarion ist erst seit einem Jahr in der Anton Paar GmbH vorhanden und der Umgang mit dieser sowie die Interaktion unter den Softwareprogrammen sind nicht geregelt.

7.2.1 Polarion

Wie im Kapitel 3 und 4.4 gefordert, ist die Software Polarion von der Firma Siemens dafür entwickelt, um Anforderungen zu managen. Sie dient zusätzlich der Kommunikation zwischen Entwicklerteams und stellt sicher, dass jeder Zugriff auf die Anforderungen hat. Je nach Version bzw. Lizenz (Polarion ALM, Polarion QA oder Polarion Requirements) hat die Software verschiedene Funktionen. Generell gibt es die Möglichkeit sogenannte Live-Documents zu erstellen. Diese Dokumente werden in der Software erzeugt und ähneln anderen Standardsoftwareprogrammen zur Dokumentengenerierung. Als Besonderheit bietet Polarion die Funktion Anforderungen als Work-Item in verschiedenen Kategorien zu erzeugen. Diese sind eigene Tasks, die selbst gestaltet und mit Text befüllt werden können. Die Work-Items können in den Dokumenten verlinkt werden, sodass die Anforderungen in einem Dokument eingebunden werden können. Individuell anpassbare Workflows, so wie in Kapitel 4.4, sind dazu geeignet den Ablauf der Bearbeitung eines Work-Items kontrolliert zu steuern. Zum Testen von Anforderungen können sogenannte Test-Cases zu jeder Anforderung erstellt werden, die in einem Test-Run abgearbeitet werden können. Der Vorteil dabei ist, dass diese Test-Runs zu verschiedenen Zeitpunkten wieder neu gestartet werden können und somit Tests in verschiedenen Stadien einer Entwicklung wiederholt durchführbar und nachvollziehbar sind. Um den Überblick über alle Anforderungen zu bewahren, den Ist-Stand der Abarbeitung zu erheben oder Auswertungen für Reporting-Zwecke zu generieren stehen automatisierte Widgets für Controlling-Tätigkeiten zur Verfügung. Wenn es gefordert ist, besteht mit dieser Software auch die Möglichkeit Work-Items in eine andere Software, wie beispielsweise Jira, zu synchronisieren.[111]

7.2.2 Jira

Generell dient die Software der Firma Atlassian zur Erstellung und Steuerung von Arbeitspaketen (sogenannten Issues). Damit sind speziell agile Methoden wie Scrum sehr gut durchführbar. Mit der Erstellung von sogenannten Scrumboards und den verschiedenen Möglichkeiten von Auswertungen, kann dezentral die Planung eines Entwicklerteams stattfinden. Jeder Issue kann einen eigenen Status innehaben und dem/der BearbeiterIn zur Bearbeitung zugewiesen werden. Zusätzlich können die Workflows individuell an den gegebenen Arbeitsstil angepasst werden. Links und Verbindungen zu anderen Softwarelösungen, wie beispielsweise Polarion oder Confluence sind möglich.[112] Derzeit wird die Software schon ab Beginn eines Entwicklungsprojekts eingesetzt.

7.2.3 Confluence

Zur Dokumentation von Lösungen und Erstellung von Modellen wird die Software Confluence der Firma Atlassian verwendet, welche als zentrale Wissensdatenbank fungiert. Hier können eigene Webseiten erstellt werden, die als Dokumentation in einem Projekt dienen. Besprechungsnotizen, Projektpläne bis hin zur Erstellung von Diagrammen via PlugIn sind möglich. Jeder hat somit Zugriff auf das Wissen Anderer und kann mit wenig Aufwand eigenes Wissen dokumentieren.[113]

---

[111] Vgl. Siemens Industry Software GmbH (2017), Online-Quelle [06.10.2017].

[112] Vgl. Atlassian (2017), Online-Quelle [08.10.2017].

[113] Vgl. Atlassian (2017), Online-Quelle [08.10.2017].

7.3 Aktuelle Verwendung von Polarion

In der Firma Anton Paar GmbH wird vereinzelt bereits die Software Polarion verwendet. Hinsichtlich deren Verwendung gibt es aber noch keine konkreten Vorgaben. Durch die individuelle Gestaltungsmöglichkeit der Software wurde diese in mehreren Projekten mehrmals unterschiedlich angewandt. Es werden hier vier Projekte dahingehend untersucht, wie und von welcher Rolle des Entwicklerteams die Software verwendet wurde.

Die Auswahl der Projekte wurde demnach so getroffen, dass ein Mix aus Umfang des Projekts, Projektphase und Typ des Projekts (Produktentwicklung, Softwareentwicklung) in Kombination mit Polarion vorhanden ist.

Folgende vier Projekte wurden ausgewählt und via Fragestellungen untersucht:

- Pico 3000
  Hierbei handelt es sich um ein Produktentwicklungsprojekt mit 766 Work-Items in Polarion. Das Projekt befindet sich in der Marktbeobachtungsphase und kann als abgeschlossen betrachtet werden.
- Xsample 530 heated
  Dieses Produktentwicklungsprojekt ist in der Entwicklungsphase und hat 222 Work-Items in Verwendung.
- AP Connect
  In diesem Projekt wird mit einem externen Partner gemeinsam eine Software entwickelt, welche via Polarion die Anforderungen mit 3582 Work-Items abbildet.
- Multi Drive
  Hierbei wird via Firmware und Software eines bestehenden Produkts derzeit das Projekt spezifiziert und ist daher noch am Anfangsstadium mit 134 Work-Items.

Mit der Beantwortung verschiedener Fragestellungen, welche gemeinsam mit der Firma Anton Paar GmbH erarbeitet wurden, soll die aktuelle Verwendung von Poalrion in der folgenden Projektanalyse geklärt werden. Dazu wird die Frage in der Tabelle angegeben und je nach Projekt in Spalten die Antwort dazu eingetragen.

7.3.1 Projektanalyse

In der folgenden Tab. 2 wird die Erhebung von Requirements (RQ) mittels verwendeter Work-Items bestimmt. Damit ist ersichtlich, welche Projekte derzeit welche Anforderungstypen erheben und welche Work-Items dafür eingesetzt werden.

Projekte	Pico 3000	Xsample 530 heated	AP Connect	Multi Drive
Work-Items	Welche Work-Items werden verwendet?			
User-Story	-	-	Ja	-
Use-Case	-	-	Ja	-
User RQ	-	Ja	Ja	Ja
System RQ	Ja	Ja (5 Stück)	Ja	-

Task	Ja	-	Ja	-
Bug	Ja	Ja (2 Stück)	Ja	-
Software RQ	Ja	Ja	-	-
Firmware RQ	Ja	Ja	-	-
Hardware RQ	Ja	Ja	-	-
Mechanical RQ	Ja	Ja	-	-
Test-Case	Ja	Ja	Ja	Ja (2 Stück)
Change RQ	-	-	Ja (3 Stück)	-
Component	-	-	Ja	-
System Interface RQ	-	-	Ja	-

Tab. 2: Verwendung der Work-Items in Polarion, Quelle: Eigene Darstellung.

In Tab. 3 sind die eingesetzten Status der Work-Items zu den Projekten gelistet, um die unterschiedlichen Workflows der einzelnen Work-Items darzustellen.

Projekte	**Pico 3000**	**Xsample 530 heated**	**AP Connect**	**Multi Drive**
Work-Items	Welche Status werden bei den Work-Items verwendet?			
User-Story	-	-	Open, Draft, Approved, In Progress, Partially Implemented, Implemented, Part. Impl. Verified, Part. Impl. Verified-Fail, Part. Impl. Verified-Bug, Verified, Rejected, Verified Fail, Verified Bug	-
Use-Case	-	-	Open, Rejected, Approved	-
User RQ	-	Open	Open, Draft, Approved, Rejected	In Definition, Open
System RQ	Open, Rejected, In Definition	Open	Open, Approved, Draft, Rejected	-
Task	Open, Resolved	-	Draft, Closed	-
Bug	Open, Closed	-	Open, Resolved, Closed	-
Software RQ	Open, Implemented, In Definition	Open	-	-
Firmware RQ	Open, Implemented, In Definition	Open	-	-
Hardware RQ	Open, Implemented, In Definition	Open	-	-
Mechanical RQ	Open, Implemented, In Definition	Open	-	-

Test-Case	Open, Rejected	Draft	Draft, Open, Closed	Open, Closed
Change RQ	-	-	Open, Draft, Fail, Verified, Approved, Verified Bug	-
Component	-	-	Open, Rejected, Approved	-
System Interface RQ	-	-	Open, Draft, Approved, Rejected	-

Tab. 3: Status der Work-Items in den Entwicklungsprojekten, Quelle: Eigene Darstellung.

In Entwicklungsprojekten werden die Work-Items von verschiedenen Personen in verschiedenen Rollen verfasst. Tab. 4 listet die Verfasser der einzelnen Work-Items in ihren Rollen im Projekt auf. Folgende Rollen sind als Verfasser generell aufgetreten:

- ProjektleiterIn (PL)
- System Engineer (SEn)
- Software Developer (SD)
- Product Specialist (PS)
- Test-Engineer (TE)
- Firmware Developer (FD)
- Externer Partner (EP)
- Technical Documentation (TD)
- Product Engineer (PE)
- Product Manager (PM)
- Software TesterIn (ST)
- Projekt AuftraggeberIn (PAG)
- Product Owner (PO)

Projekte	**Pico 3000**	**Xsample 530 heated**	**AP Connect**	**Multi Drive**
Work-Items	Wer verfasst die Work-Items?			
User-Story	-	-	PS, SEn, PL, EP, TD	-
Use-Case	-	-	PS, SE, PL,	-
User RQ	-	PL	SEn, PL, PS	PL, PO, FD
System RQ	PL, SE	PL	EP, SD, SE	-
Task	SD	-	PS	-
Bug	TE	TE	PS, EP, TD, PE, PM	-
Software RQ	PL, SD	PL	-	-
Firmware RQ	PL, FD	PL	-	-
Hardware RQ	PL	PL	-	-
Mechanical RQ	PL	PL	-	-
Test-Case	PL, SD, TE	PL	EP, PS, ST, TD	PL, FD
Change RQ	-	-	-	-
Component	-	-	PAG	-
System Interface RQ	-	-	SE	-

Tab. 4: Verfasser der Work-Items, Quelle: Eigene Darstellung.

In Polarion können Test-Cases zusammengefasst mit sogenannten Test-Runs durchgeführt werden. Des Weiteren kann Polarion Work-Items nach Jira synchronisieren. In Tab. 5 wird erfasst, welche Work-Items mit Polarion getestet und welche Work-Items nach Jira synchronisiert wurden.

Projekte	Pico 3000	Xsample 530 heated	AP Connect	Multi Drive
	Welche Work-Items werden via Polarion getestet?			
	Software RQ, Firmware RQ, Hardware RQ, Manual System Test	Software RQ	User Story, System RQ, Automated Test, Manual Test	
	Wurde die Schnittstelle zu Jira genützt? Wenn ja, welche Work-Items wurden synchronisiert?			
	Ja Software RQ	Ja Software RQ	Nein	Nein, noch nicht.

Tab. 5: Auflistung der Tests und Transfer zu Jira, Quelle: Eigene Darstellung.

7.3.2 Fazit der Projektanalyse

Anhand der Ist-Stand-Erhebung der Verwendung von Polarion ist ersichtlich, dass jedes Projekt (egal ob Produktentwicklung oder Softwareentwicklung) Polarion unterschiedlich nutzt. In den Projekten gibt es unterschiedlichste Work-Items, die unterschiedliche Anforderungen abdecken. Vor allem in den Produktentwicklungsprojekten sind keine UC, aber die domänenspezifischen Anforderungen sehr stark verwendet worden. Zusätzlich ist auch aufgefallen, dass eine Softwareentwicklung andere Bestandteile gegenüber einer mechatronischen Produktentwicklung benötigt.

Die Verwendung der Status und der damit verbundenen Workflows ist nicht überall vorhanden und zudem sind diese nicht einheitlich. Dabei können Verwirrungen bei der Erstellung von Anforderungen entstehen. Es gilt eine Definition zu finden, die von der Erstellung und bis hin zur Abarbeitung durchgängig funktioniert.

Erstellt wurden die meisten Work-Items von PL, das damit einen wesentlichen Mehraufwand zur Folge hat. Auch in diesem Punkt muss definiert werden, wer welche Work-Items erstellt und auch prüft. Zusätzlich ist durch die schon bestehenden Abläufe in der Firma auch Jira in Verwendung. Die Synchronisation der Work-Items muss geregelt und dadurch mögliche Redundanzen in beiden Softwareprogrammen zu vermeiden.

8 REQUIREMENTS-MANAGEMENT FÜR DIE PRODUKTENTWICKLUNG BEI DER ANTON PAAR GMBH

Damit es eine definierte Vorgehensweise für das Requirements-Management in einer Produktentwicklung gibt, wird in diesem Kapitel anhand der Literatur und der Erkenntnisse der Ist-Stand-Erhebung ein Requirements-Management-Leitfaden als Konzept entwickelt, das vorgibt, wie bei einer Produktentwicklung Anforderungen eingeteilt und zueinander in ein Verhältnis gestellt werden. Dazu werden Kategorien der Anforderungen erstellt, definiert und in Anlehnung an das V-Modell aus Kapitel 6 zur hierarchischen Darstellung und Abarbeitung als Requirements V-Modell abgebildet.

8.1 Requirements-Kategorien

Grundsätzlich lassen sich viele Kategorien für Anforderungen erstellen. Um die Übersicht zu behalten, werden nur die folgenden vier Kategorien von Anforderungen (Work-Items für Polarion) definiert:

- Use-Case
  Ein Use-Case ist ein Anwendungsfall, der bei einer Produktentwicklung eine Messung, die ein Kunde mit dem zu entwickelnden Produkt durchführen können muss, widerspiegelt. Dabei ist der Anwendungsfall derart zu beschreiben, dass ohne zusätzlichem Wissen die Messung durchführbar ist. Verfasser sollte ein PM, PS, PO oder PAG sein.
- User Requirement
  In dieser Anforderung sollen Komponenten oder Voraussetzungen angegeben werden, die beschreiben, was der Anwender benötigt, um den vorhergehenden Use-Case durchzuführen. Es besteht die Möglichkeit, dass es die Komponenten bereits gibt oder diese erst entwickelt werden müssen. Als Verfasser dieser Kategorie kommen wieder PM, PS, PO oder PAG in Frage.
- System Requirement
  Diese Anforderung wird als technische Spezifikation gesehen. Es muss beschrieben werden, was das zu entwickelnde Produkt respektive das System zur Verfügung stellen muss, um die geforderten Use-Cases und User-Requirements zu erfüllen. Hier müssen als Verfasser PL, SEn und die benötigten EntwicklerInnen tätig werden.
- Test-Case
  Diese Work-Items sind zum Testen der vorher beschriebenen Anforderungen notwendig. Zur besseren Übersicht müssen keine separaten Work-Items zur Verifikation und Validierung erstellt werden. Dennoch muss im Test-Case angegeben werden, um welche der beiden Arten es sich handelt, um die Tests trotzdem eindeutig zu trennen. Verfasst werden die Tests von PL, SEn, TE, ST, PE, PM, PS und/oder PO.

Auf User-Stories wird absichtlich verzichtet, da es sich bei diesem Leitfaden um ein Konzept für eine mechatronische Produktentwicklung handelt. User-Stories eignen sich für Softwareneuentwicklungen besser, da kein physikalisches Produkt entsteht und die Beschreibung mittels Use-Case nicht ausreicht. Bei einer mechatronischen Produktentwicklung ist schlussendlich ein physikalisches Produkt vorhanden,

welches mit einem Use-Case ausreichend beschrieben werden kann. Die User-Story bringt daher keinen weiteren Vorteil.

8.2 Das Requirements V-Modell

Der Ablauf und der Zusammenhang der definierten Anforderungen, werden in Abb. 19 dem Requirements V-Modell illustriert und damit in eine Beziehung zueinander gestellt. Das Requirements V-Modell wurde so entwickelt, dass alle Anforderungen in einer Beziehung zueinander stehen. Hierbei ist eine hierarchische Struktur gegeben, damit untergeordnete Anforderungen davon abgeleitet werden können. Grundsätzlich sind in diesem Modell *drei Ebenen* vorhanden:

- Die erste Ebene beschäftigt sich mit dem RE sowie RM und den dazugehörigen Tests (Validierung und Verifikation). Dafür ist Polarion auf Grund des möglichen Anforderungs- und Testmanagements ausgewählt worden.
- Die Abarbeitung der Arbeitspakete behandelt die zweite Ebene und wird in Jira vollzogen, da die Prozesse und agilen Abarbeitungsmethoden der Firma Anton Paar GmbH darauf abgestimmt und etabliert sind.
- Den ganzen Prozess entlang bilden Modelle und Diagramme aus dem Kapitel 5 die dritte Ebene, welche in Confluence erstellt und zu den benötigten Anforderungen oder Arbeitspaketen (Polarion oder Jira) verlinkt werden können.

Zum Erstellen von Use-Cases dienen die Methoden, welche im Kapitel 4 angeführt wurden. Nach deren Bestimmung können davon die User-Requirements abgeleitet werden. Nach der Bestimmung dieser zwei Kategorien (Use-Case und User-Requirements) müssen die Komponenten und Voraussetzungen, welche der Anwender benötigt, erhoben sein. Im Live-Document der *Rahmenvorgabe* (Frameworkspecification), welches in Polarion erzeugt wird (siehe Abb. 44 auf Seite 69), sind diese Definitionen der Anforderungen anzuführen. Dem folgend kann damit begonnen werden, die Anforderungen, welche das zu entwickelnde System zur Verfügung stellen muss (System Requirements), zu definieren und in das Live-Document der *Spezifikation* (Specification) einzubringen. Zu diesem Zeitpunkt sind alle Tests zur Prüfung der Use-Cases und User-Requirements sowie System Requirements zu erstellen. Schon zu diesem Zeitpunkt können diese in sogenannte Test-Runs zusammengefasst und dadurch ein Validierungs- und Verifikationsplan erstellt werden (siehe Abb. 41).

Um Redundanzen von Arbeitspaketen zu vermeiden und die Abarbeitung in ein mögliches agiles Umfeld zu bringen, werden mittels *Connector* die System Requirements in einen Epic in Jira synchronisiert (siehe Kapitel 9.3.3). Ein Epic ist in Jira ein Haupttask, der die Tätigkeiten des System Requirement mit untergeordneten Subissues beinhaltet. Diese Issues können verschiedenen Disziplinen zugeordnet werden. Durch die unterschiedlichen Arbeitsweisen der Disziplinen kann separat entschieden werden, ob die Abarbeitung im agilen Umfeld (siehe Kapitel 4.6) oder klassisch mit einem Zieldatum und einer Zuweisung an ein Teammitglied durchgeführt werden kann.

Ist die Implementierung eines Arbeitspaketes abgeschlossen, wird dieses in Jira mittels Unit-Test (Entwicklertest, Code-Review, HIL-Test oder SIL-Test etc.) getestet. Ist dieser erfolgreich, kann als erster integrativer Test das System Requirement als technische Implementierung übergeordnet mittels Test-Run verifiziert werden. Somit ist gewährleistet, dass das Produkt technisch richtig funktioniert. Der Verifi-

kation übergeordnet findet die Validierung anhand der Abschlussmessungen aus Sicht des Kunden statt. Dabei wird ein Test-Run der Validierungstests abgearbeitet der die User-Requirements und Use-Cases prüft.

Fällt ein Test negativ aus, werden die Informationen dokumentiert und in dem jeweils verlinkten Requirement eingearbeitet, sodass das falsche Verhalten entfernt werden kann. Hierbei handelt es sich generell um einen iterativen Prozess. Auch Änderungen, wie in Kapitel 4.5 beschrieben, sind einzuordnen. Je nach Änderungswunsch einer Anforderung sind die Auswirkungen durch die hierarchischen Abhängigkeiten sofort eindeutig ersichtlich und der Änderungsaufwand besser abschätzbar.

Das ganze Requirements V-Modell wird von Diagrammen begleitet (siehe Abb. 19), die helfen sollen die Anforderungen übersichtlicher oder in zusammenfassender Weise darzustellen. Bei unterschiedlichen Kategorien der Anforderungen können Modelle und Diagramme aus Kapitel 5 in Confluence erstellt und dokumentiert werden.

Durch die geforderten Dokumente und der vorgegebenen Phasen im PEP ist die Erstellung der verschiedenen Requirements zeitlich fixiert. Use-Cases, User Requirements und die Tests für den Validierungsplan sind demnach in der Projektauftragsphase zu erstellen. Die Definition der System Requirements (und Epics), des Verifikationsplans und der Abarbeitungstasks finden der Spezifikationsphase statt. Schlussendlich wird in der Entwicklungsphase die Implementierung anhand von Prototypen via Unit-Tests, Verifikationstests bis hin zu Validierungsmessungen im Projekt durchgeführt. In der Überleitungsphase wird mittels Nullserie die Verifikation und Validierung im Serienprozess wiederholt. Durch den iterativen Prozess ist dargelegt, dass einzelne Tests schon zwischenzeitlich auch vorher durchgeführt werden können, um wichtige Testresultate schon vorab im Entwicklungsprojekt einfließen zu lassen.

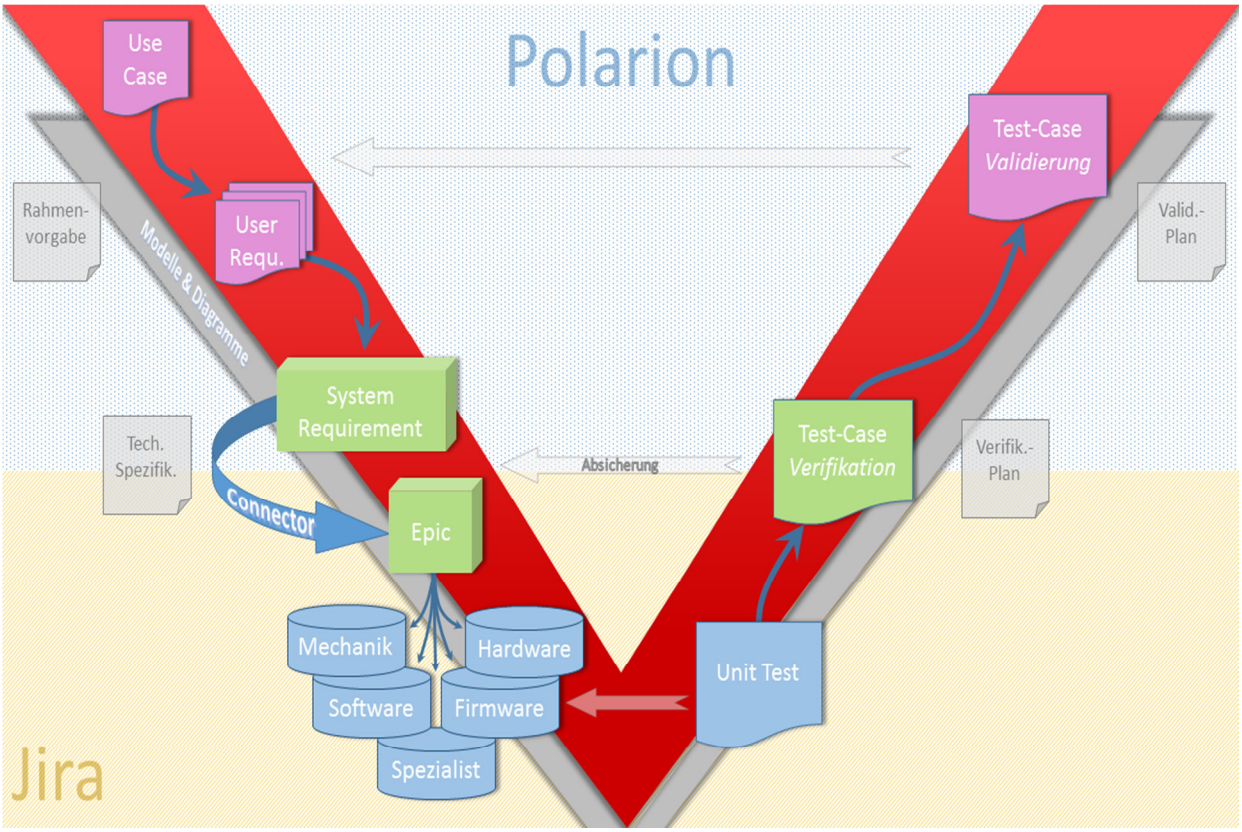


Abb. 19: Das Requirements V-Modell, Quelle: Eigene Darstellung.

8.3 Qualitätskriterien für die Erstellung von Requirements

Um eine qualitativ hochwertige Anforderungen zu erhalten, sind, wie im Kapitel 3.3.2 erwähnt, Qualitätskriterien für die Anforderungen in Polarion zu definieren.

Als erste Maßnahme für eine hochwertige Anforderung sollen die erstellten Anforderungen von einer zweiten Person gegengelesen werden, damit etwaige Missverständnisse schon nach dem Erstellen der Anforderung erkannt werden. Als Überprüfer der Anforderung bzw. des Test-Cases kommen Personen in Frage, welche auch als Ersteller für die jeweilige Kategorie möglich sind (siehe Kapitel 8.1). Erstellt ein PM einen Use-Case, soll dieser beispielsweise von einem anderen PM oder einem PS geprüft werden.

Als zweite Qualitätsmaßnahme werden Fragestellungen aufgelistet, welche beim Erstellen und auch beim Überprüfen der Anforderung helfen sollen. Können die Fragen beantworten werden, gilt die Anforderung als vollständig. Folgende Fragen soll sich der Ersteller bzw. der Prüfer einer Anforderung stellen:

- Ist die Anwendung oder der Test mit diesen Angaben von einer Person durchführbar?
- Ist geklärt wer diese Anwendung benötigt?
- Lässt sich davon eine weitere Anforderung ableiten?
- Welche Materialien, bzw. Hilfsmittel werden dafür benötigt?
- Ist die Anforderung mit anderen Anforderungen verlinkt bzw. wird davon eine Anforderung abgeleitet?
- Lässt die Anforderung die technische Realisierung offen (gilt nicht für System Requirements)?
- Ist die Anforderung realistisch?

8.4 Definition des Work-Item Workflows

Damit der Workflow der einzelnen Work-Items (Anforderungen) geregelt abläuft und übersichtlich gehalten wird, wird der Workflow in Abb. 20, so wie in Kapitel 4.4 gefordert, definiert. Ziel ist es, den Workflow schlank zu halten und für die meisten Projekte nutzbar zu machen. Nach der Erstellung einer Anforderung ist diese als *Draft* markiert. Nun muss die Anforderung einem oder einer PrüferIn zugewiesen werden, welche bzw. welcher den Status auf *Open* setzt oder zur Nacharbeit dem/der ErstellerIn wieder zuweist. Ist eine Anforderung durch die Entwicklungstätigkeit im System implementiert, erhält die Anforderung den Status *Ready for Test*, damit ein Testlauf gestartet werden kann. Ist der Test erfolgreich gilt die Anforderung als V*erified*. Fällt der Test negativ aus, wird zur Nacharbeit wieder der Status *Open* verwendet. Um in verschiedenen Entwicklungsphasen eine implementierte Anforderung nochmals testen zu können, kann eine Anforderung via T*est again* mehrmals getestet und wieder geöffnet werden. Erst wenn die Anforderung in der Überleitungsphase das letzte Mal vor einer Serienfreigabe getestet wird, sind alle Tätigkeiten dabei erledigt und die Anforderung ist als *Closed* zu betrachten. Sollte dennoch auffallen, dass eine Implementierung einer abgeschlossenen Anforderung nochmals überarbeitet werden muss, kann mit *Reopen* die Anforderung nochmals in den Workflow eingebracht werden.

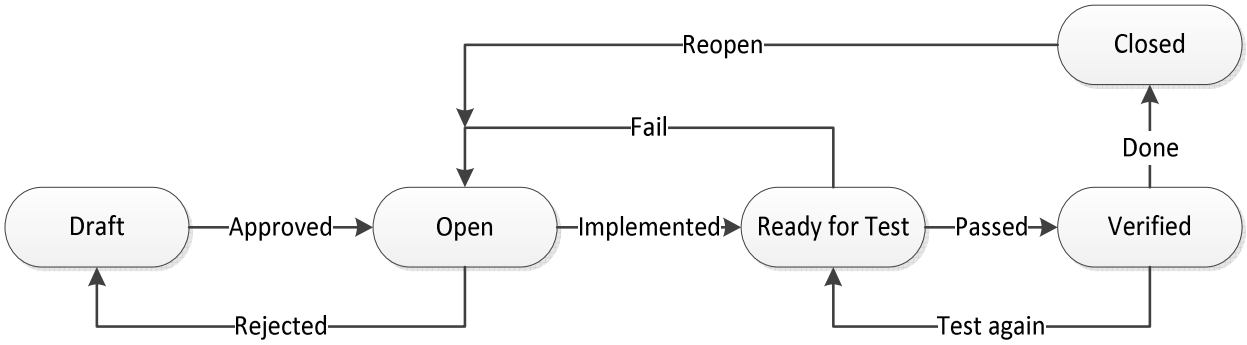


Abb. 20: Workflow für die Work Items, Quelle: Eigene Darstellung.

Für die Work-Items, welche Test-Cases sind, ist es notwendig einen eigenen Workflow zu definieren, da sie keine Implementierung beinhalten. Wie in Abb. 21 ersichtlich, wird der Status *Open* entfernt und sofort nach der Überprüfung als *Ready for Test* angegeben.

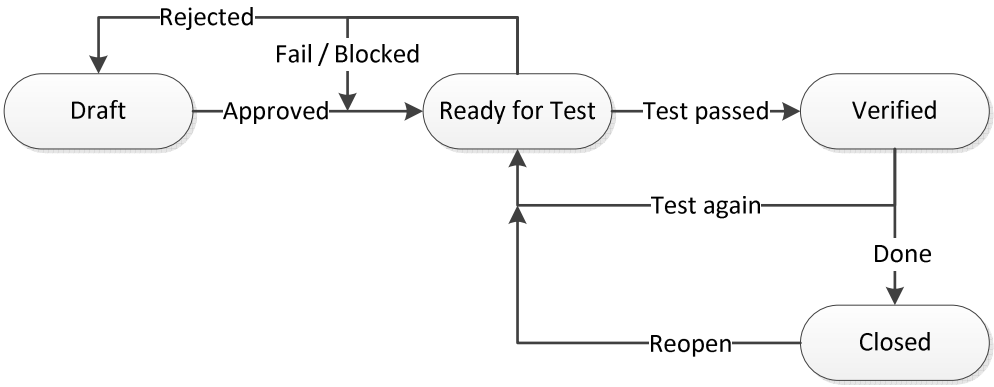


Abb. 21: Workflow für Test-Cases, Quelle: Eigene Darstellung.

Auch bei Test-Cases soll eine Prüfung durch eine zweite Person erfolgen. Befindet die zweite Person den Test-Case für ausreichend beschrieben, soll der Status *Ready for Test* angeben werden, sodass der Test-Case bereit für eine Aufnahme in einen Test-Run ist (siehe Kapitel 9.5). Wird der Test positiv bewertet, gilt er als *Verified*. Bei negativer Bewertung bleibt der Status erhalten, da die Korrektur später wieder getestet werden muss. Wenn der Test beispielsweise auf Grund von fehlenden Equipment nicht ausgeführt werden kann, wird durch die Angabe von *Blocked* der Test übersprungen und *Ready for Test* bleibt erhalten. Zusätzlich besteht die Möglichkeit in verschiedenen Stadien des Projekts mit der Funktion *Test again* nochmals den Test, beispielsweise einmal in der Entwicklungsphase und einmal in der Überleitungsphase, auszuführen. Schlussendlich kann via *Done* der Test geschlossen werden, und falls doch später etwas wieder getestet werden muss, soll via R*eopen* der Status *Ready for Test* hergestellt werden.

9 EINSATZ DES REQUIREMENTS-MANAGEMENTS IM ENTWICKLUNGSPROJEKT

In diesem Kapitel kommt der entwickelte Requirements-Management-Leitfaden zum Einsatz und wird bei einem laufenden Entwicklungsprojekt in der Firma Anton Paar GmbH eingesetzt. Grundsätzlich läuft das Entwicklungsprojekt laut dem derzeitigen PEP und deshalb wird für diese Masterarbeit das Projekt anhand des entwickelnden Requirements-Leitfadens redundant erstellt. In Polarion werden alle Anforderungen und Tests anhand des V-Modells aufgenommen und die System Requirements nach Jira in ein Testprojekt synchronisiert, damit die anfallenden Tätigkeiten erstellt werden könnten. Des Weiteren werden in Polarion der Verifikations- und Validierungsplan, sowie die Rahmenvorgabe und das Spezifikationsdokument erstellt. In der Software Confluence werden die benötigten Modelle als Basis der Anforderungen dokumentiert, welche in Polarion und Jira verlinkt werden können.

Der Abarbeitungsworkflow in Jira ist in der Firma Anton Paar GmbH etabliert und nicht Thema dieser Masterarbeit. Daher wird dies nur bis zur Synchronisation mit Jira behandelt, damit auf die Verwendung der Anforderungen, den Modellen und Diagrammen der Fokus gelegt wird.

9.1 Kurzbeschreibung des Entwicklungsprojekts

Das Entwicklungsprojekt ist im Geschäftsbereich der Rheologie der Firma Anton Paar GmbH in Auftrag gegeben worden. Abb. 22 zeigt das High-End Rheometer MCR 702 TwinDrive mit zwei luftgelagerten Messmotoren, welche jeweils wechselbare Messkörper besitzen (derzeit nur mit automatischer Radio Frequency Identification (RFID) Funktion zur Erkennung der Messgeometrie gelöst). Um die Messkörper herum können verschiedene Temperierkammern montiert werden. Auf dieser Abbildung ist die Temperierkammer mit dem Namen Convection Temperature Device 180 (CTD 180) montiert, die via Konvektion die Probe zwischen den beiden Messkörpern temperiert.

Im unteren der beiden Messkörper soll eine Temperaturmessung entwickelt werden, welche den Temperaturwert der Probe am rotierenden Motor kontaktlos an das MCR 702 TwinDrive weitergibt. Diese Messkörper werden als Temperatursensor-Messkörper (TS-Messkörper) bezeichnet. Das Entwicklungsprojekt wird daher als TS-Messkörperprojekt geführt und befindet sich derzeit in der Entwicklungsphase.

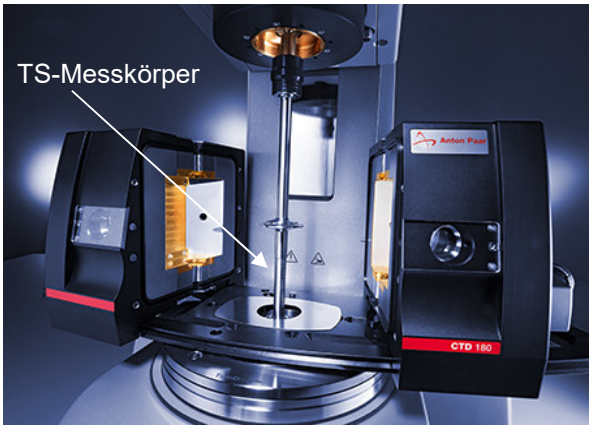


Abb. 22: MCR 702 TwinDrive, zu entwickelnder TS-Messkörper und Temperierkammer, Quelle: Anton Paar GmbH (2017), Online-Quelle [16.10.2017] (leicht modifiziert).

Wie mittels Requirements-Management-Leitfaden von einem Use-Case auf alle weiteren Anforderungen abgeleitet wird, soll die Abb. 23 darstellen.

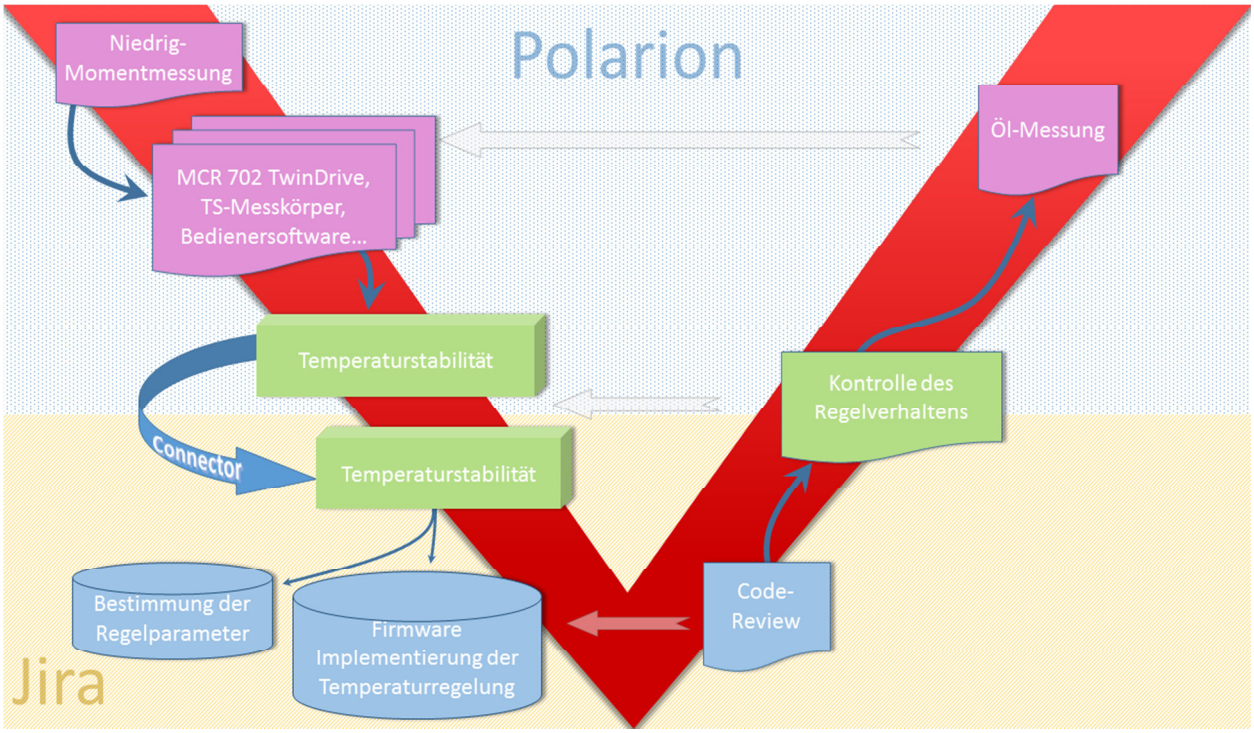


Abb. 23: Beispiel des Requirements-Management-Leitfadens, Quelle: Eigene Darstellung.

In diesem Beispiel ist der Use-Case *Niedrig-Momentmessung* definiert. Als nötige User-Requirements sind das *MCR 702 TwinDrive, der TS-Messkörper* und die *Bedienersoftware* angegeben. Dafür werden im realen Projekt nun mehrere System Requiremet benötigt. Beispielsweise ist die *Temperaturstabilität* der Probe notwendig. Um diese zu gewährleisten, werden die Aufgaben *Bestimmung der Regelparameter* und die *Firmware Implementierung der Temperaturregelung* erstellt. Als Unit-Test wird nun ein *Code-Review* durchgeführt, um die Implementierung zu überprüfen. Als Verifikation wird die *Kontrolle der Temperaturstabilität* mittels Temperatursprünge verifiziert. Zum Schluss muss es möglich sein, den Use-Case und das User-Requirement mittels *Öl-Messung* zu validieren.

9.2 Modelle und Diagramme des Entwicklungsprojekts

Die Modelle und Diagramme für das TS-Messkörper Entwicklungsprojekt wurden in Confluence mit dem *DrawIO-Diagrams* Plug-In, welches UML unterstützt, erstellt, und dienen als Unterstützung der Anforderungen. Sie stellen die Grundlage für die Entwicklung dar und helfen dem Team bei der Übersicht über alle Anforderungen. In den folgenden Kapiteln beginnen die Diagramme mit der Abgrenzung der Systeme und werden immer detaillierter. Die Kontextabgrenzung, das Zielmodell, das Use-Case Diagramm sowie die Einteilung der Anforderung im Kano-Modell unterstützen die Use-Cases und die User-Requrements in Polarion. Das Sequenzdiagramm ist zur Unterstützung der Erstellung von System Requirements und den daraus untergeordneten Tasks zur Abarbeitung vorhanden.

9.2.1 Kontextabgrenzung

Im EPr werden zwei Diagramme als Kontextabgrenzung erstellt. Das erste Diagramm in Abb. 24 zeigt alle Systeme in diesem Projekt und ob sie miteinander in Kontakt stehen. In blau sind die noch nicht vorhandenen und somit zu entwickelnden Systeme *TS-Messkörper* und *Licht-Interface* zu sehen. Die kontaktlose Datenübertragung ist in das System des bestehenden *TwinDriveMotor* (unterer luftgelagerter Motor) zu integrieren. Die in beige gehaltenen Systeme sind bestehende Systeme, welche adaptiert werden müssen, um die Temperaturerfassung zu ermöglichen. Die Anpassung im *MCR 702* ist als Änderung der bestehenden Firmware gedacht. Die beiden Systeme *RheoCompass* und *RheoServiceTool* sind Softwareprogramme, die ebenfalls angepasst werden müssen, damit die Temperatur am Computer verarbeitet werden kann. Die Linien, welche die einzelnen Systeme verbinden, sind die Schnittstellen und mögliche Verbindungen der einzelnen Systeme. Somit ist ersichtlich, dass der Benutzer Zugriff auf die Softwareprogramme, das MCR 702, die sogenannte CTD und auch auf den Messkörper selbst hat.

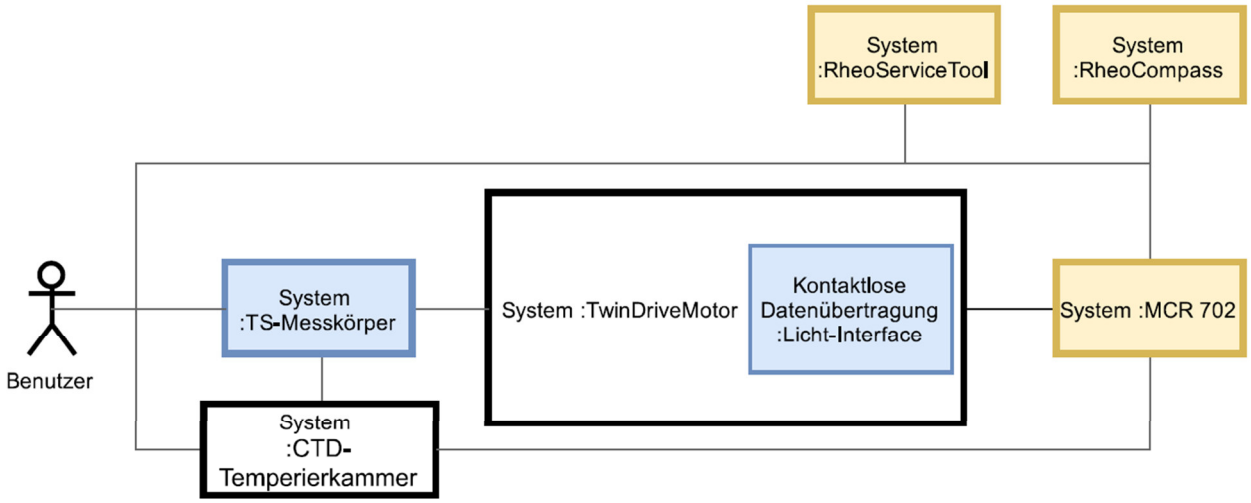


Abb. 24: Kontextabgrenzung des TS-Messkörpers, Quelle: Eigene Darstellung.

Im zweiten Diagramm der Kontextabgrenzung ist die Wirkkette der Temperaturerfassung aufgezeigt. Hier wird der Detailierungsgrad erhöht, um einen Einblick zu bekommen, wie die Temperatur bis zur ersten Platine (EXE2) des TwinDrive-Motors gelangt.

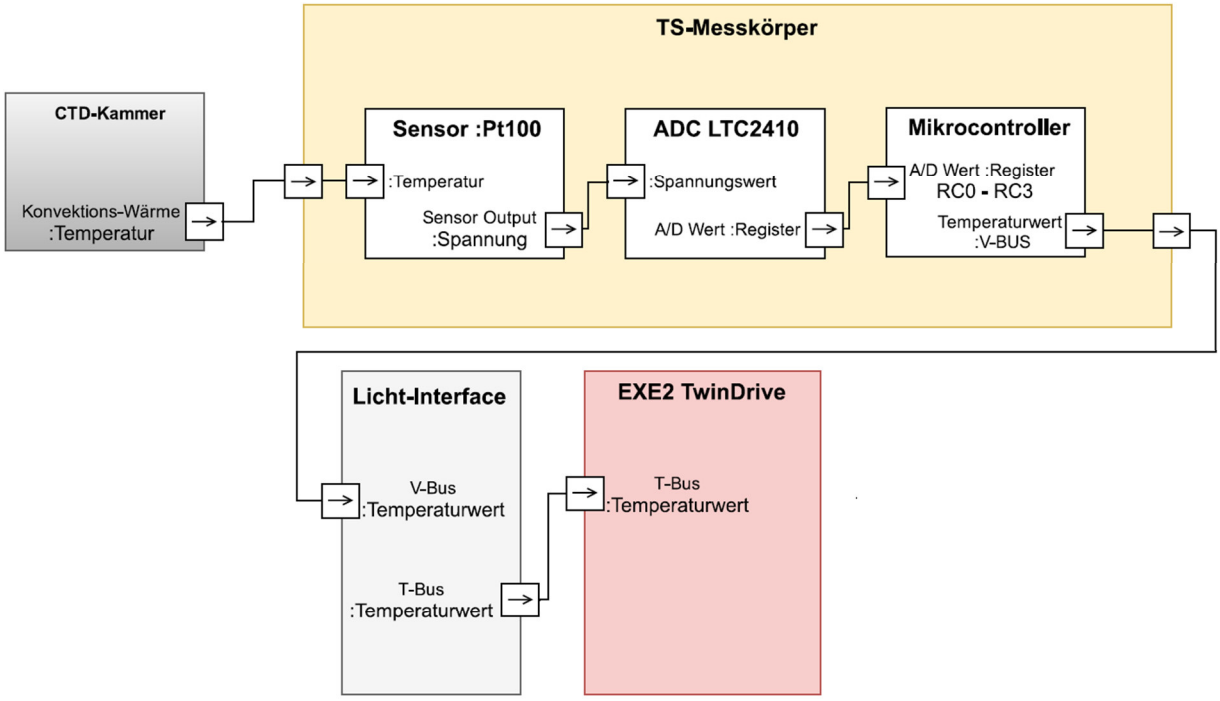


Abb. 25: Wirkkettendiagram des Entwicklungsprojekts, Quelle: Eigene Darstellung.

In der Abb. 25 wird gezeigt, wie die Konvektionswärme der Temperierkammer CTD vom Pt100 Sensor im TS-Messkörper erfasst, mit dem Analog Digital Converter (ADC) in einen digitalen Wert umgewandelt wird und via Mikrocontroller und Licht-Interface zur Platine EXE2 im TwinDriveMotor gesendet wird.

9.2.2 Zielmodell des Entwicklungsprojekts

Das Zielmodell des TS-Messkörperprojekts in Abb. 26 zeigt auf, welche Ziele im Projekt erreicht werden sollen. Dieses Modell wurde entsprechend dem Kapitel 5.2 erstellt. Das oberste Ziel ist die Entwicklung des TS-Messkörpers. Die darunter liegenden Ziele sind dem obersten Ziel untergeordnet. Sie tragen aber zur Erfüllung des Hauptziels bei. Mit diesem Modell ist eine sehr gute Übersicht über die Ziele des Projekts vorhanden auch sind einzelne Tätigkeiten sind im Laufe des Projekts den Zielen zuordenbar. Durch diese Darstellung bekommen Tätigkeiten eine Bedeutung.

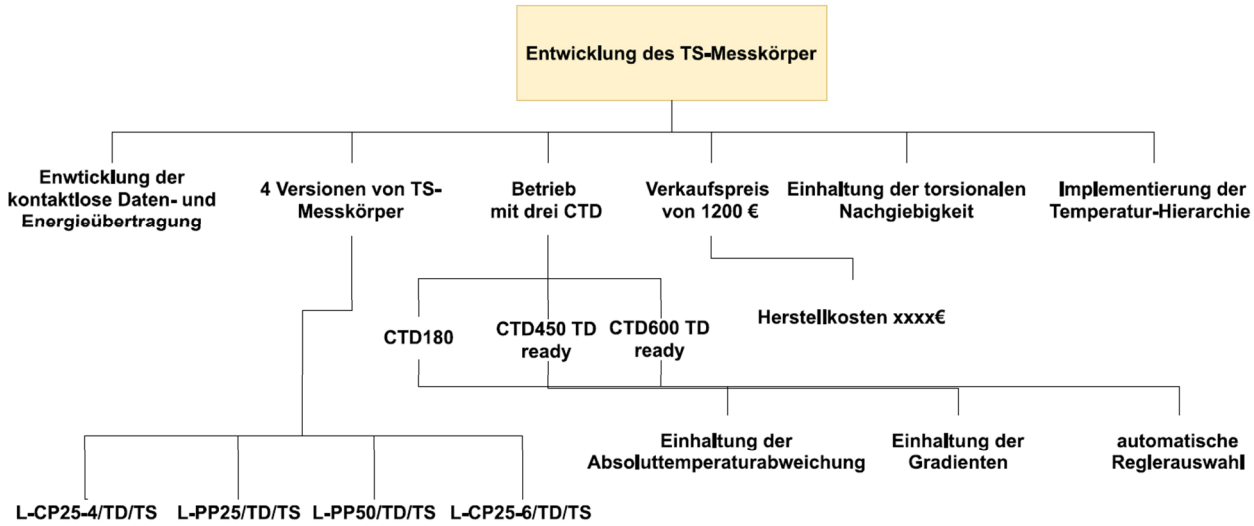


Abb. 26: Zielmodell des TS-Messkörperprojekts, Quelle: Eigene Darstellung.

In diesem Zielmodell gibt es nur UND-Verknüpfungen der Ziele. Die untergeordneten Ziele sind als Zwischenziele respektive Teilziele oder auch als Etappen zu sehen, welche im Projekt als Meilensteine definiert werden können. Das Entwicklerteam sieht somit das große Ganze und die Teilziele bei diesen sie mitwirken.

Die untergeordneten Teilziele zeigen die Entwicklung eines kontaktlosen Daten- und Energieübertragungssystems, welches zum Auslesen der Temperatur im Messkörper notwendig ist. Es ist eine eigenständige Entwicklung eines Teilsystems. Des Weiteren sollen vier verschiedene Varianten der Messkörper entwickelt werden. Zusätzlich soll ein Betrieb von drei CTD, bewerkstelligt werden. Bei jedem dieser CTD soll die Absoluttemperturabweichung, der Gradient und die automatische Heizungsreglerauswahl realisiert werden. Der Verkaufspreis mit den dazugehörigen Herstellkosten, die Einhaltung der torsionalen Nachgiebigkeitswerte für Messkörper und die Implementierung einer Temperatur-Hierarchie in der Firmware, sind weitere Ziele, die im Projekt erreicht werden müssen.

Die Angabe der genauen Herstellkosten, welche erreicht werden müssen, wird in dieser Masterarbeit aus Datenschutzgründen nicht angegeben. Es ist aber ein Ziel, dass bestimmte Herstellkosten erreicht werden müssen, um den Verkaufspreis erreichen zu können.

9.2.3 Use-Case Diagramm des Entwicklungsprojekts

Das Use-Case Diagramm der TS-Messkörper aus Abb. 27 zeigt vier Anwendergruppen: die VerkäuferInnen, die MontagemitarbeiterInnen, die Service MitarbeiterInnen und die WissenschafterInnen. Damit ist eine Übersicht aller Anwendungen der zu entwickelnden Messkörper gegeben.

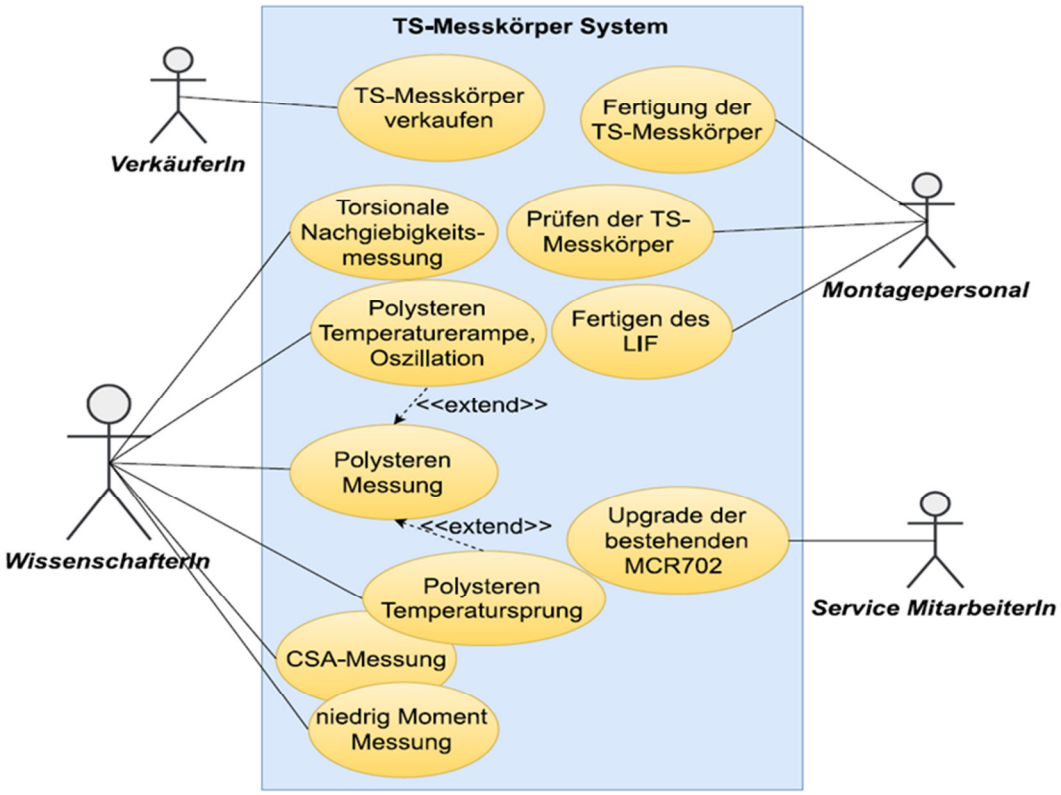


Abb. 27: Use-Case Diagramm der TS-Messkörper, Quelle: Eigene Darstellung.

Mit dieser Darstellung der Use-Cases ist die Entwicklung der TS-Messkörper auf definierte Anwendungen bezogen. Der Vorteil darin besteht, dass schon zu Beginn des Entwicklungsprojekts festgelegt ist, für welche Zwecke das zu entwickelnde Produkt eingesetzt wird. In diesem Fall sind vor allem die Messungen des Endkunden wichtig. Speziell hierfür ist das Projekt in Auftrag gegeben worden. Dennoch sind zusätzliche Anwender mit Use-Cases erarbeitet worden, die meist übersehen werden, aber wichtige Anforderungen liefern. Hiermit ist aufgezeigt, dass beispielsweise das Montagepersonal durch den Use-Case *Fertigen des TS-Messkörpers* einen definierten Fertigungsprozess benötigt und hieraus schon von Anfang an Anforderungen bestehen.

9.2.4 Gliederung der Anforderungen anhand des Kano-Modells

Die Anforderungen nach dem Kano-Modell aus Kapitel 5.1 drücken beim TS-Messkörperprojekt aus, worauf ein spezielles Augenmerk gelegt wird. Es ist dadurch beispielsweise offensichtlich, welche Anforderungen den Kunden besonders begeistern und dadurch zum Kauf bewegen sollen bzw. was vorausgesetzt wird. Die folgende Aufzählung zeigt die Anforderungen des TS-Messkörperprojekts in den drei Bereichen Basisfaktoren, Leistungsfaktoren und Begeisterungsfaktoren:

- Basisfaktoren
    - Die automatische Messkörpererkennung, da die bestehenden RFID-Systeme das bereits unterstützen. Der neue TS-Messkörper muss diese Funktionalität somit auch bieten.
    - Kein Einfluss auf die Messgenauigkeit des MCR 702.
    - Einhaltung der torsionalen Nachgiebigkeit des TS-Messkörpers.
    - Auswahl der richtigen Temperatur durch die Firmware, da durch die hinzugekommene Messkörpertemperatur ein Prioritäten Konflikt im Messgerät stattfinden kann.
    - Verwendung der Messkörperdaten in der Software RheoCompass.
    - Programmierung der TS-Messkörper mit der Software RheoserviceTool.
    - Mechanisch robuste Messkörper.
- Leistungsfaktoren
    - Temperaturmessung in einem Messkörper für TwinDrive-Motoren.
    - Die Temperaturmessung soll so nah wie möglich bei der Probe erfolgen.
    - Betrieb der Messkörper in den vier Temperierkammern CTD180, CTD450 TD ready und CTD600 TD ready.
    - Einhaltung der geforderten Absoluttemperaturabweichung und Gradienten.
    - Einhaltung der Herstellkosten und somit die Realisierung des geforderten Verkaufspreises.
    - Geforderter Temperaturbereich von -160 °C bis 600 °C.
- Begeisterungsfaktoren
    - Vier Messkörpervarianten mit Temperaturmessung.
    - Kontaktlose Energie- und Datenübertragung im TwinDrive-Motor.
    - Geringe Temperaturabweichungen, durch die Nähe des Temperaturfühlers an der Probe.

9.2.5 Sequenzdiagramme

Wie schon in der Beschreibung des Entwicklungsprojekts kurz erwähnt, sollen neue Messkörper (TS-Messkörper) entwickelt werden. Da zurzeit die Funktionalität der Standard-Messkörper via RFID besteht, sind zwei Sequenzdiagramme erstellt worden, damit der Erkennungsablauf der Messkörper in der Firmware des Rheometers bei der Verwendung beider Messkörper abgebildet ist. Diese Sequenzdiagramme sind als Präzisierung der geforderten Funktionalitäten, wie beispielsweise der automatischen Temperaturregler-Erkennung notwendig. Zusätzlich zeigt dieses Diagramm auf, wie der neue zusätzliche Temperaturwert im Rheometer eingebunden wird. Somit sind hier auch schon Architekturentscheidungen mit eingebunden.

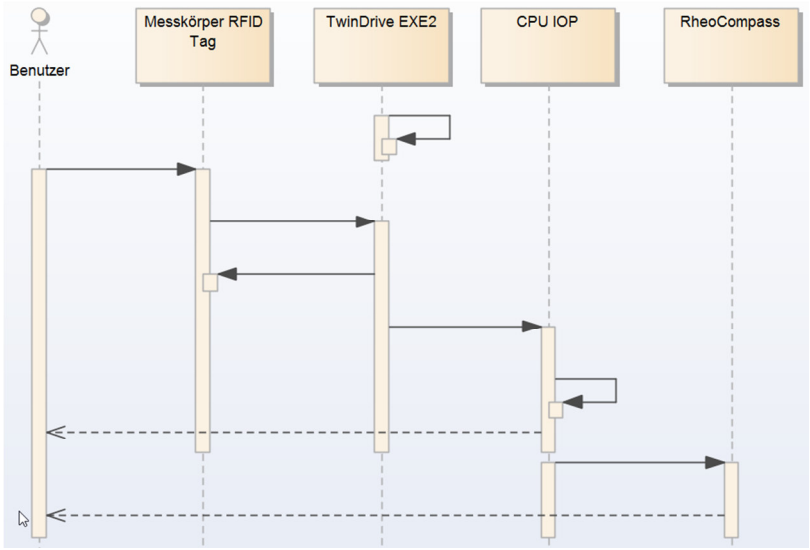


Abb. 28: Simplifiziertes Sequenzdiagramm mit Standard-Messkörper, Quelle: Eigene Darstellung.

In Abb. 28 handelt es sich aus Lesbarkeitsgründen um das simplifizierte Bild des derzeitigen Datenaustauschs verschiedener Teilsysteme, welcher beim Einstecken des RFID-Messkörpers in das Rheometer entsteht. Der *RFID Tag* ist dabei im Messkörper implementiert. Die Elemente *TwinDrive EXE2* und *CPU IOP* sind im Rheometer verbaut, wobei das Element *RheoCompass* die Software am Computer darstellt.

Im detaillierten Sequenzdiagramm des Standard-Messkörpers im Anhang 1 (siehe Abb. 47) ist erkennbar, dass schon vor dem Einstecken des Messkörpers nach einem Messkörper gesucht wird. Das geschieht so lange, bis sich auf die Such-Anfrage ein RFID-Messkörper meldet, der vom Benutzer in der Zwischenzeit eingesteckt wurde. Sobald sich dieser Messkörper meldet, werden die Daten des Messkörpers (Block_MSrv) geladen und an den *CPU IOP* weitergegeben. Dieser wählt je nach dem angeschlossenen Equipment und Messkörper den geeigneten Temperaturregler und die Temperatur mit der höchsten Hierarchiestufe als Substanztemperatur aus. Danach werden die Messkörperdaten an die Software *RheoCompass* übergeben. Sind die Daten fertig übermittelt, wird die Substanztemperatur am Display angezeigt. Dem folgend beginnt die Übertragung aller vorhandenen Temperaturen. In diesem speziellen Fall sind es drei Temperaturen: die Substanztemperatur (SubstanzT) und die beider Temperaturen der angeschlossenen CTD600 (CTD_L, CTD_R). Werden diese von der Software empfangen, stehen sie dort zur Verfügung und die Sequenz ist abgeschlossen.

In dem folgenden zweiten simplifizierten Sequenzdiagramm ist der neue Ablauf mit der Verwendung eines neuen TS-Messkörpers angedeutet.

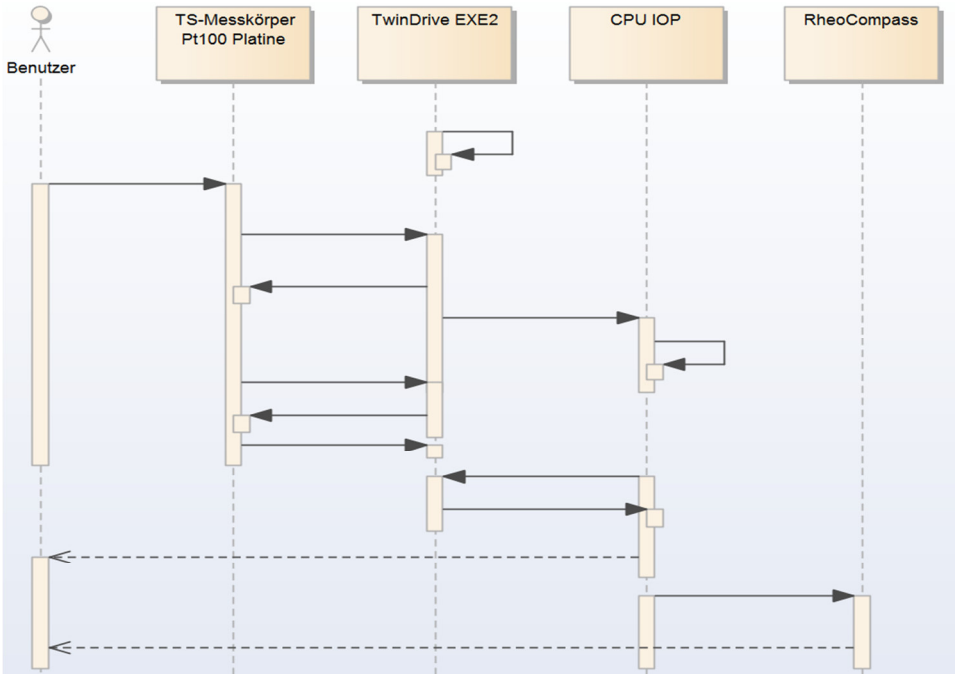


Abb. 29:Simplifiziertes Sequenzdiagramm mit TS-Messkörper, Quelle: Eigene Darstellung.

Im detaillierten Sequenzdiagramm mit der Verwendung des TS-Messkörpers im Anhang 2 (Abb. 48) wird der Datenaustausch und die Temperaturwertübertragung mit den geleichen Elementen wie beim Standard Sequenzdiagramm *TwinDrive EXE2, CPU IOP* und *RheoCompass* dargestellt. Nur das erste Element *TS-Messkörper Pt100 Platine*, welche im TS-Messkörper implementiert ist, unterscheidet sich.

Auch hier wird vor dem Einstecken des Messkörpers nach einem Messkörper gesucht. Die Suche nach einem Messkörper ist dahingehend unterschiedlich, dass abwechselnd nach einem Standard- und einem TS-Messkörper gesucht wird. Meldet sich ein Standard-Messkörper, wird die Sequenz der Abb. 28 ausgeführt.

Meldet sich nach dem Einstecken eines Messkörpers ein TS-Messkörper bei der *TwinDrive EXE2*, wird kontrolliert, ob der Messkörper bereits einmal eingesteckt wurde und möglicherweise Daten davon vorhanden sind. In der Sequenz der Abb. 48 sind noch keine Daten gespeichert und daher werden diese abgefragt, geladen und gespeichert. Anhand der Daten ist ersichtlich, dass hier wesentlich mehr Daten zur Verfügung stehen. Es werden grundsätzlich die gleichen Parameter (Block_MSrv) und zusätzlich die Daten (AxialCompliance, Masse und Temperaturregel-Parameter) geladen.

Nach dem Speichern der Daten sendet das Element *TwinDrive EXE2* alle Parameter an den *CPU IOP*. Anhand der Kontrolle des Gesamt-Equipments wird die Substanztemperatur nach der Hierarchie ausgewählt und der Temperaturregler bestimmt. Danach werden an die *RheoCompass* Software nur mehr die relevanten Daten weitergeleitet.

Die Temperaturwertübertragung beginnt zwischen dem *TS-Messkörper Pt100 Platine* und der *TwinDrive EXE2* nach dem Übermitteln der Messkörperdaten. Auch wenn noch nicht nach der Temperatur gefragt wird, ist der aktuellste Wert bereits im Element *TwinDrive EXE2* vorhanden. Wird der Temperaturwert von

dem *CPU IOP* abgefragt sendet das *TwinDrive EXE2* Element den letzten empfangenen Temperaturwert. Der empfangene Temperaturwert wird zusammen mit allen anderen Temperaturen an die *RheoCompass* Software weitergeleitet und die Substanztemperatur am Display angezeigt. Sind die Temperaturwerte in der Software verarbeitet, stehen sie zur Verfügung und die Sequenz ist beendet.

9.3 Einstellungen in Polarion für das Entwicklungsprojekt

Um das Requirements V-Modell in Polarion anwenden zu können, müssen Einstellungen getroffen werden. Auf Grund des Lizenzmodells des Herstellers der Software hat nicht jeder die Möglichkeit dazu. Als Projektleiter dieses Entwicklungsprojekts und wegen der vorhandenen Lizenz Polarion QA ist der Zugriff auf die Administration im Polarionprojekt freigeschaltet.

Beim Start eines Projekts sind Grundeinstellungen als Template schon vorhanden, die nun auf das Requirements V-Modell adaptiert werden.

9.3.1 Einstellung der Work-Items

Die im Requirements-Management-Leitfaden definierten Work-Items Use-Case, User-Requirement, System Requirement und den Test-Case sind in Polarion mühelos einzustellen. Des Weiteren kann ein Icon in der Farbe, wie im Leitfaden in Abb. 19 dargestellt, gewählt werden, um mehr Bezug dazu herzustellen.

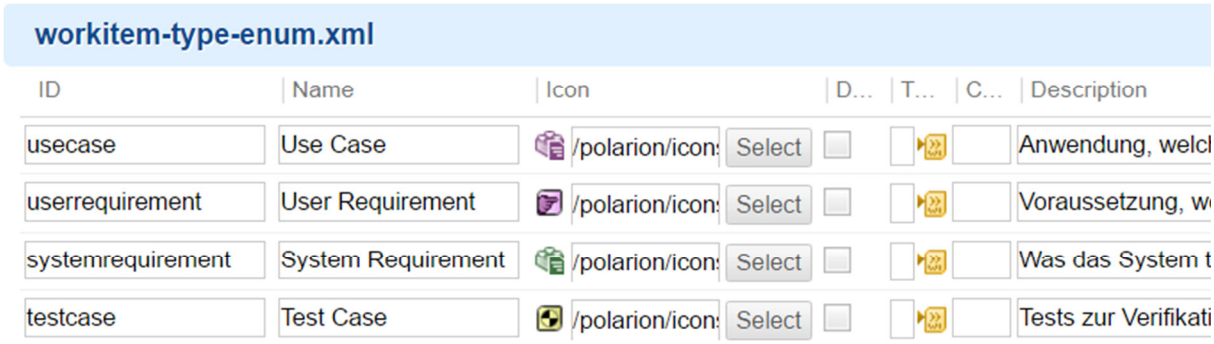


Abb. 30: Einstellung der Work-Items, Quelle: Eigene Darstellung.

In Abb. 30 sind zwei Ausschnitte von den Einstellungen der Work-Items aus Polarion abgebildet. Es ist erforderlich eine ID, einen Namen und ein Icon zu definieren. Die Beschreibung ist optional und muss nicht unbedingt erfolgen. In diesem Projekt ist die Beschreibung der Work-Items an das Requirements V-Modell angepasst. Sind diese Work-Items konfiguriert, können sie verwendet und angelegt werden.

Es gibt noch eine Reihe an weiteren Einstellungsmöglichkeiten. Erwähnenswert ist dabei, dass zwei Felder hinzugefügt werden, die in den Work-Items verwendbar sind. Das erste Feld ist das Feld *Actor.* In diesem Feld können bei UC und User-Requirements die aus dem UCD definierten Anwender eingetragen werden. Das zweite Feld ist das Feld *Typ*. Dieses Feld dient bei den Test-Cases als Deklarierung des Tests als Verifikation oder Validierung.

Die Einstellungen der Live-Documents sind im gleichen Stil wie bei Abb. 30 vorhanden. Beim vorliegenden TS-Messkörperprojekt wurden zwei Dokumente, Frameworkspecification (Rahmenvorgabe) und Specification (technische Spezifikation) eingestellt.

9.3.2 Einstellung des Wokflows

Um den definierten Workflow aus Abb. 20 nun in Polarion zu erreichen, wird der Workflow wie in Abb. 31 gezeigt, eingestellt. In den Feldern *Statuses* sind die möglichen Status anzugeben. Im Bereich der *Actions* sind die Übergänge von einem zum nächsten Status einzutragen. Im obersten Bereich *Transitions* finden die Verbindungen der *Statuses* mittels *Actions* statt, sodass sich der gewollte Workflow ergibt.

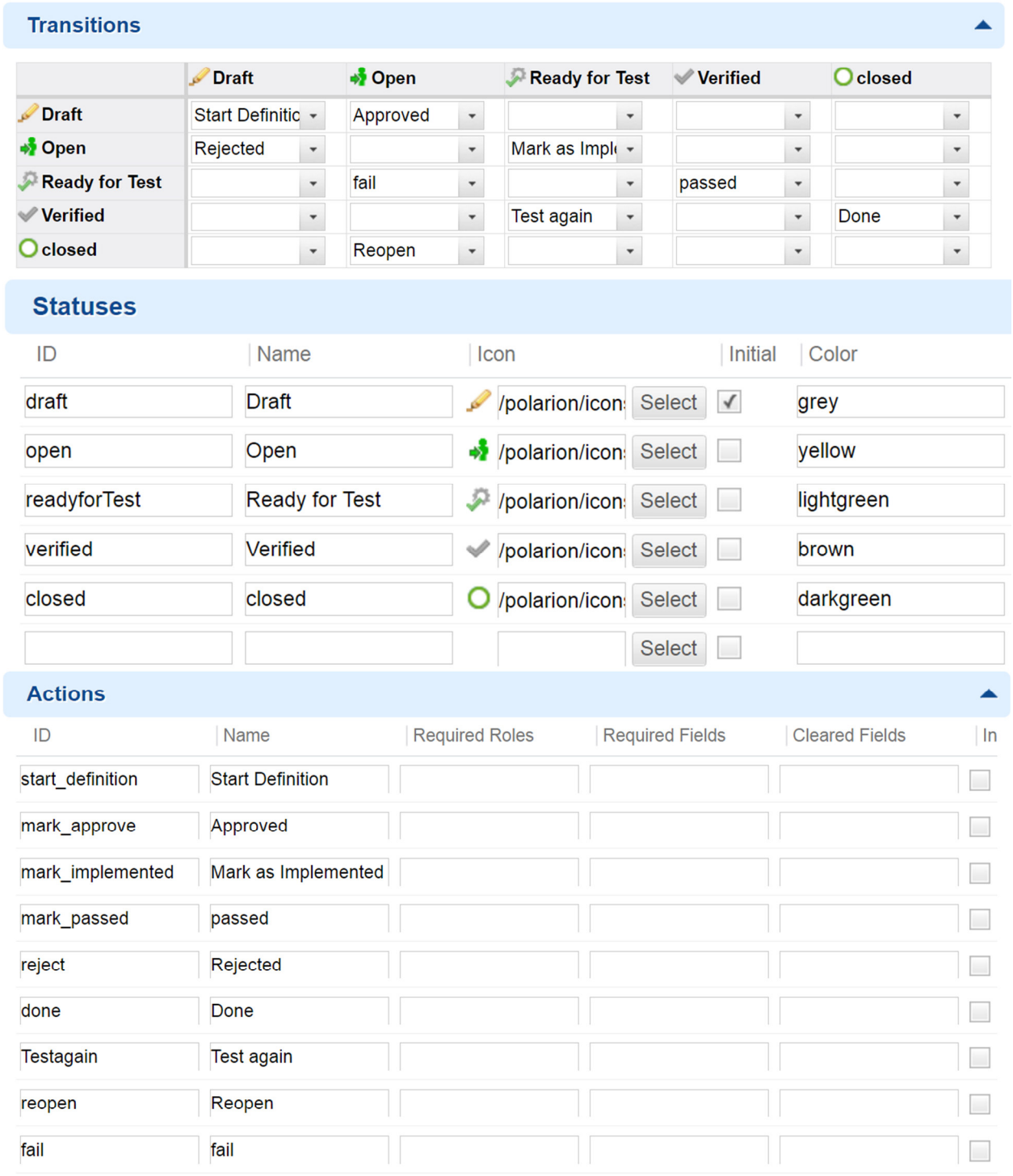


Abb. 31: Definition des Workflows in Polarion, Quelle: Eigene Darstellung.

9.3.3 Einstellung des Connectors zu Jira

Die Konfiguration des Connectors für die Synchronisation der Work-Items aus Polarion zu Epics in Jira, muss einmal in Polarion vorgenommen werden. Dazu sind zwei Schnittstellen angegeben. Als erstes ist auf der linken Seite das Polarion Projekt anzugeben und als zweites das Jira Projekt rechts.

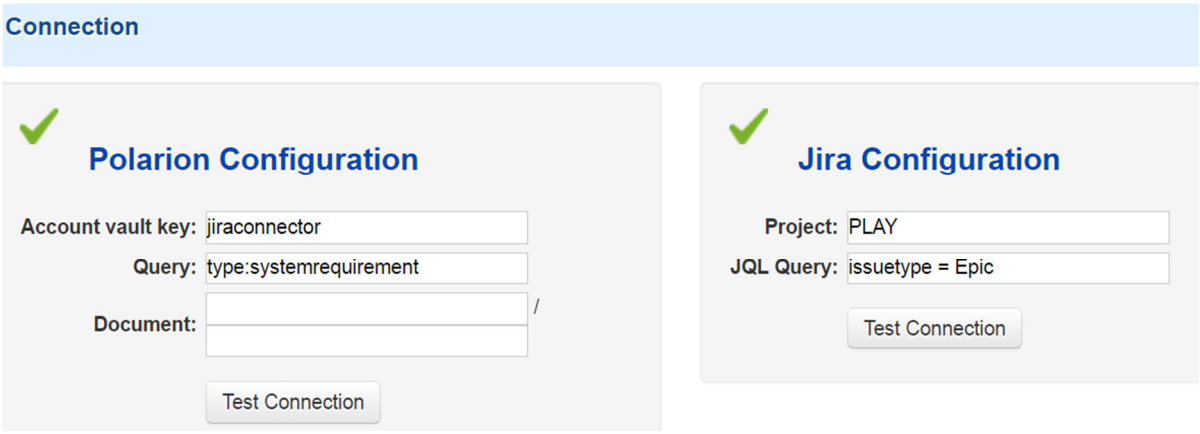


Abb. 32: Connector Konfiguration, Quelle: Eigene Darstellung.

Abb. 32 zeigt nun diese Einstellung, bei der zusätzlich das Projekt und ein dazugehörender Query angegeben werden muss. Hier wird schon das erste Mal entschieden, was genau synchronisiert wird. Wie der Leitfaden aus Kapitel 8 vorgibt, wird aus einem System Requirement in Polarion ein Epic in Jira erstellt. Zum Schluss kann schon via Test Connection die Verbindung der beiden Projekte getestet werden, welche mit einem grünen Haken als funktionierend dargestellt wird.

In diesem Projekt wird vom Polarion Projekt *TS-Messkörper* zu einem Testprojekt *Play* in Jira synchronisiert, damit die laufenden Projekte nicht gestört werden.

Sind diese Voreinstellungen der grundsätzlichen Verbindung des Connectors zu Jira getätigt, wird das Mapping der einzelnen Typen (Type Mapping) und den Feldern (Field Mapping) eingestellt. Als Typ muss hier ein zweites Mal angegeben werden, dass ein System Requirement zu einem Epic synchronisiert wird (siehe Abb. 33).

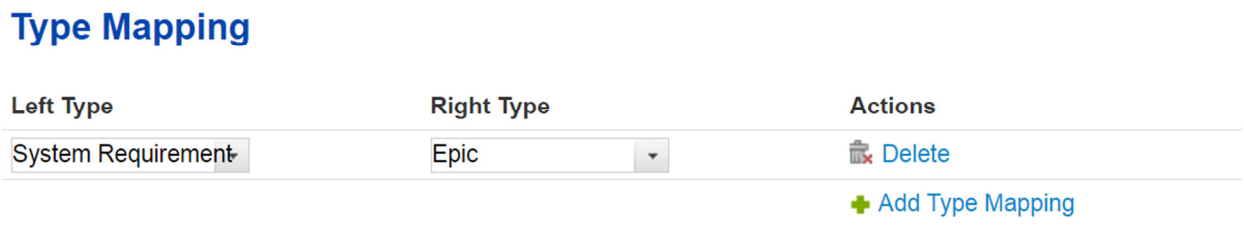


Abb. 33: Type Mapping Einstellung in Polarion, Quelle: Eigene Darstellung.

Ist der Typ festgelegt, werden die Einstellungen der Felder im *Field Mapping* vorgenommen. Hier wird jedes Feld, welches synchronisiert werden soll, angeführt. Dabei kann die generell möglichen Richtungen (*Bidirectional, Left to Right oder Right to Left*) und die primär gewollte Richtung eingestellt werden. Im Fall der Abb. 34 werden die ersten Felder angezeigt und grundsätzlich sind alle als *Bidirectional* eingestellt.

Das bedeutet, es wird je nach Änderung in beide Richtungen synchronisiert. Die Hauptrichtung ist dennoch von Polarion nach Jira (*Left to Right*), damit beim Synchronisieren die Epics in Jira erstellt werden.

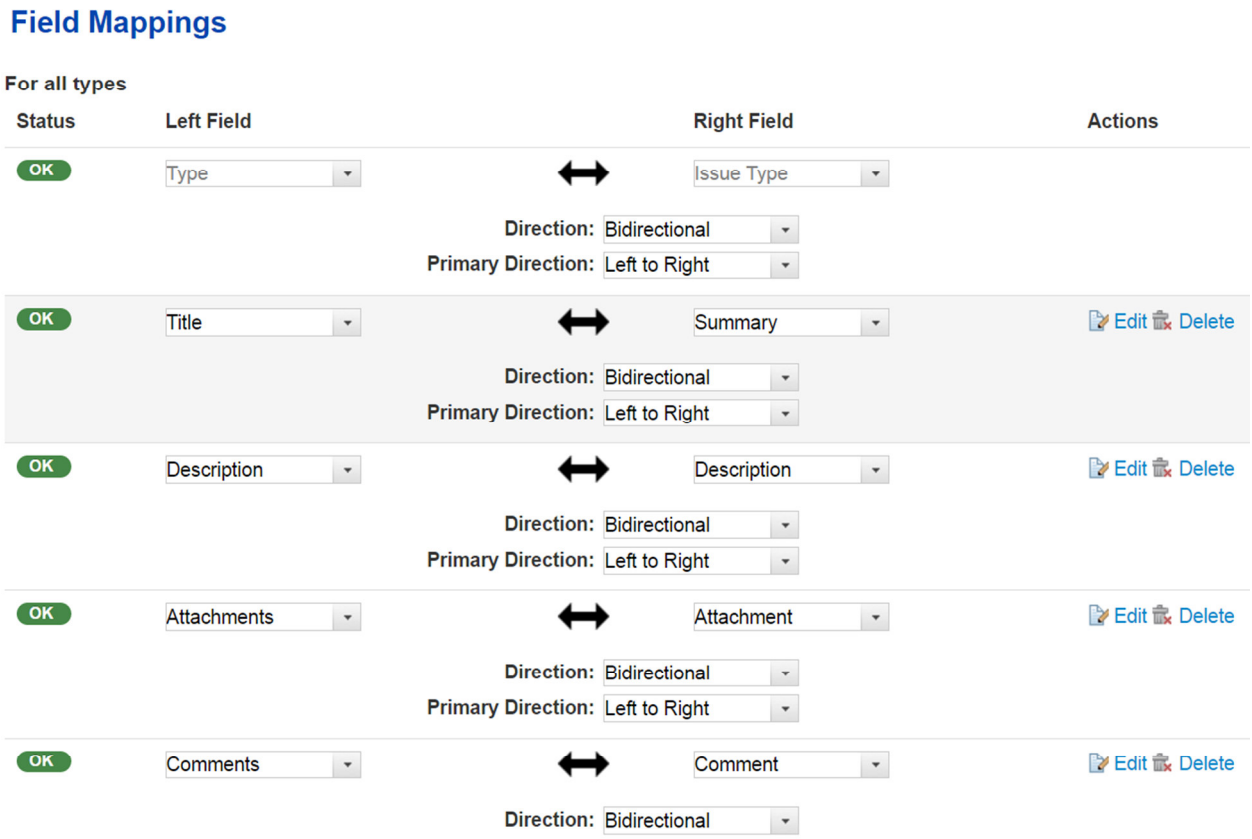


Abb. 34: Field Mapping Einstellungen in Polarion, Quelle: Eigene Darstellung.

Des Weiteren zeigt Abb. 34 einen Knopf *Edit*. Mit diesen kann jedes Feld noch genau zugewiesen werden. Beispielsweise kann eine bestimmte Priorität in Polarion, einer bestimmten Priorität in Jira zugewiesen werden.

Sind diese Einstellungen alle mit dem Status OK bzw. die Verbindung mit einem grünen Haken versehen, wird das Synchronisieren mit dem Knopf *Execute* ausgeführt (siehe Abb. 35).

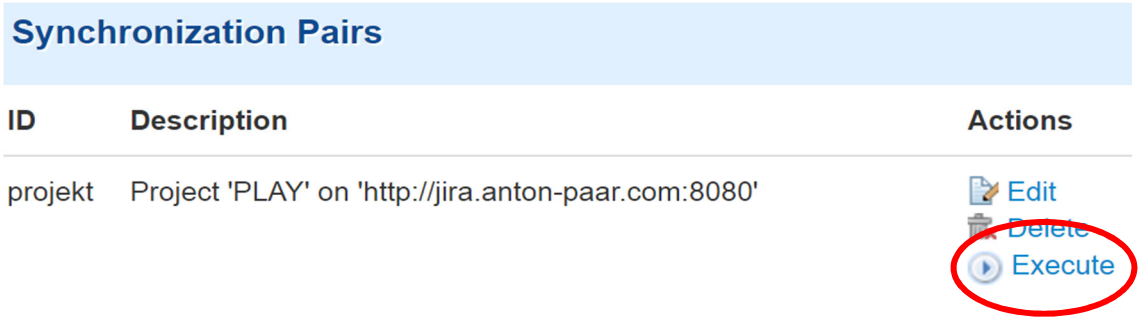


Abb. 35: Synchronisation Polarion zu Jira, Quelle: Eigene Darstellung.

Nach Drücken dieses Knopfes sind die System Requirements des Polarion Projektes samt allen angegebenen Feldern in Jira als Epic vorhanden. Das Ausführen der Synchronisation findet derzeit mit dem Drücken des *Execute* Knopfes statt. Es besteht auch die Möglichkeit, dass nach einem freiwählbaren Zeitintervall eine Synchronisation ausgeführt wird. Dies ist derzeit aber nicht aktiv.

Auch nachherige Änderungen der synchronisierten Work-Items in Polarion bzw. Epics in Jira (siehe Abb. 36 und Abb. 37) sind kein Problem. Wird beispielsweise ein Text einer Beschreibung oder ein Status geändert, wird nach der Synchronisation die letzte Änderung in beide Richtungen übernommen. Probleme gibt es nur, wenn im Beschreibungstext Bilder vorhanden sind, welche nach Jira synchronisiert wurden. Diese sind als Verweise nach Polarion in Jira integriert und werden nach einer Rück-Synchronisation gelöscht. Abhilfe schafft ein Bild im Anhang (egal ob in Polarion oder Jira). Diese werden immer in beide Richtungen richtig synchronisiert. Wie erkennbar ist, ist die Formatierung nicht gleich. Die Texte sind zwar richtig übernommen, aber Aufzählungspunkte werden nicht richtig synchronisiert. Das stört zwar beim Lesen, ist aber nicht weiter ein Hindernis.

Description

Die Firmware muss beim MCR dahingehend angepasst werden:

- Neue Temperatur vorhanden
- Hierarchie der vorhandenen Temperaturen im MCR
    - 1. TMS
      2. TD2
      3. TD-Master
      4. ext. Pt100
      5. COM-Thermostat
      6. TD-Slave
      7. TC
- Programmieren der Messkörper
- Neues Datenformat am Messkörper --> Daten des Messkörper, Regelparameter

Abb. 36: Anforderung in Polarion, Quelle: Eigene Darstellung.

Details

Type:	Epic	Status:	OPEN (View Workflow)
Priority:	Blocker	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Labels:	PM-MCR		
Epic Name:	MCR Firmware Anpassung für TS-Messkörper		
Epic Status:	To Do		
Epic Color:	ghx-label-2		

Description

Die Firmware muss beim MCR dahingehend angepasst werden:
Neue Temperatur vorhandenHierarchie der vorhandenen Temperaturen im MCR1. TMS
2. TD2
3. TD-Master
4. ext. Pt100
5. COM-Thermostat
6. TD-Slave
7. TCProgrammieren der MesskörperNeues Datenformat am Messkörper --> Daten des Messkörper, Regelparameter

Abb. 37: Anforderung in Jira, Quelle Eigene Darstellung.

9.4 Ermittlung der Requirements in Polarion

Das Ermitteln der Anforderungen wurde im realen Projekt schon vor dieser Masterarbeit gemeinsam mit dem PM und dem Entwicklerteam durchgeführt. Deshalb wurde aus den schon bestehenden Dokumenten Rahmenvorgabe und Spezifikation, so wie im Kapitel 4.1.2 unter Dokumentenarchäologie erwähnt, die Use-Cases, die User-Requirements bis hin zu den System Requirements herausgefunden und in Polarion erzeugt. Abb. 38 zeigt einen Teil der erzeugten und untergeordneten Work-Items. Hier sind auch die zusätzlichen Felder *Typ* für Verifikation oder Validierung (für Test-Cases) und *Actor* zur Angabe der ausführenden Person (für Use-Cases und User-Requirements) zu erkennen.

ID	Title	Status	Typ	Actor
TSM-220	Ermittlung der torsionalen Nachgiebigkeit			WissenschafterIn
TSM-252	Messkörper Typen			WissenschafterIn
TSM-295	Polysterenmessung		Validierung	
TSM-280	L-CP25-x/TD/TS Absoluttemperaturabweichung, Gradientenmessung		Verifikation	
TSM-279	L-PP50/TD/TS Absoluttemperaturabweichung, Gradientenmessung		Verifikation	
TSM-278	L-PP25/TD/TS Absoluttemperaturabweichung, Gradientenmessung		Verifikation	
TSM-276	Hardware Entwicklung (Temp.-Sensorplatine)			
TSM-28	Überprüfung der Platinentemperatur		Verifikation	
TSM-274	Umsetzung des Fertigungsprozesses			
TSM-29	Rückmeldungen und Test der Fertigung des LIF		Validierung	
TSM-28	Rückmeldungen und Test der Fertigung der Messkörper		Validierung	
TSM-28	LIF Test		Verifikation	
TSM-269	Regelparameter für die TS-Messkörper			

Abb. 38: Auszug der Work-Items im TS-Messkörper Projekt, Quelle: Eigene Darstellung.

Wenn eine Anforderung erstellt wird, sind Felder vorgegeben, welche ausgefüllt werden müssen und das Work-Item bekommt schon bei dessen Erstellung eine eindeutige Nummer. Angelehnt an die Beschreibung von Use-Cases aus Kapitel 5.6.2 und der Tab. 1 sind bei jedem Work-Item insbesondere bei Use-Cases ein Name des Use-Cases, eine Priorität (*must have, should have* oder *nice to have*), Status, Beschreibung sowie Akteur notwendig und Attachments bis hin zu Links zu anderen Work-Items möglich. Genau diese Punkte werden beim Erstellen ausgefüllt und sollen von einer Zweitperson später überprüft werden, damit die Qualität der Anforderungen gehalten wird (siehe Kapitel 8.3).

Wie schon erwähnt und im Kapitel 3.2.3 gefordert, werden zusammenhängende Work-Items via Link verbunden. Abb. 39 zeigt die Verlinkungen eines User-Requirements mit den anderen Work-Items und dem Dokument, in dem es vorkommt. In diesem Beispiel ist das User-Requirement von drei Use-Cases abgeleitet (Bezeichnung *refines*). Des Weiteren verfeinern drei System Requirements wiederum dieses User-Requirement (Bezeichnung *is refined by*).

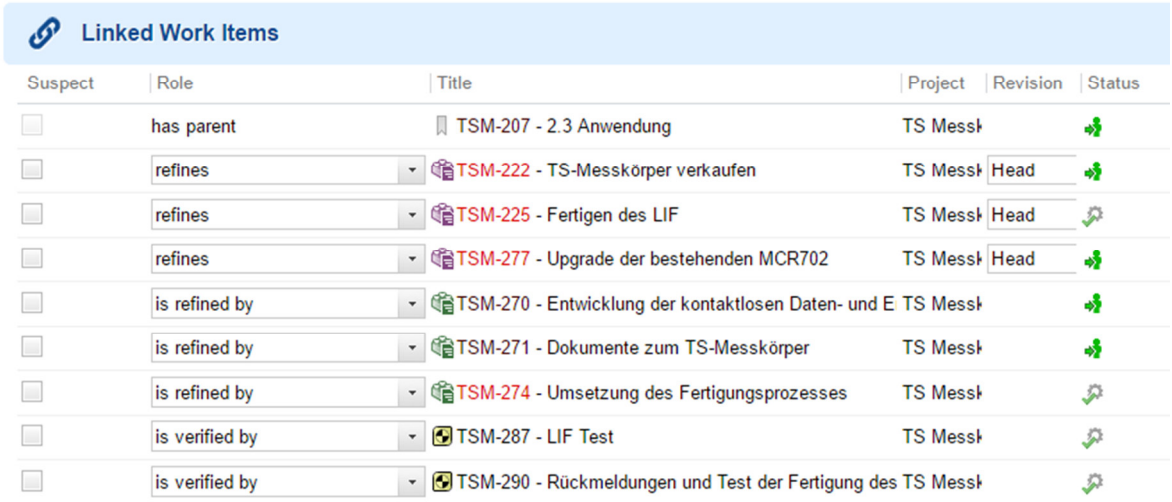


Abb. 39: Mögliche Verlinkungen eines User-Requirements, Quelle: Eigene Darstellung.

Überprüft wird diese Anforderung von zwei Tests (Bezeichnung *is verified by*). Mit der Bezeichnung *has parent* ist der Link in das Dokument Framework Specification angegeben. Das Work-Item ist daher unter Punkt 2.3 Anwendungen im Dokument erwähnt. An diesem Beispiel macht sich der Vorteil des RM bemerkbar, da hier eindeutig ersichtlich ist, warum es diese Anforderung gibt, wie sie technisch umgesetzt und wie sie getestet wird.

Für dieses Entwicklungsprojekt sind schlussendlich 30 Work-Items erstellt worden, welche von 21 Tests geprüft werden.

9.5 Test-Cases mit Test-Runs

Testfälle werden als Test-Case samt Beschreibung (zu verwendendes Material, Toleranz etc.) erstellt, welche wiederum bei einer Verifikation zu System Requirements und bei einer Validierung Use-Cases oder User-Requirements zugewiesen werden. Die Zuweisung geschieht via Link mit dem Zusatz *verifies*. Wichtig zu erwähnen ist dabei, dass ein Test-Case auch mehrere Anforderungen überprüfen kann.

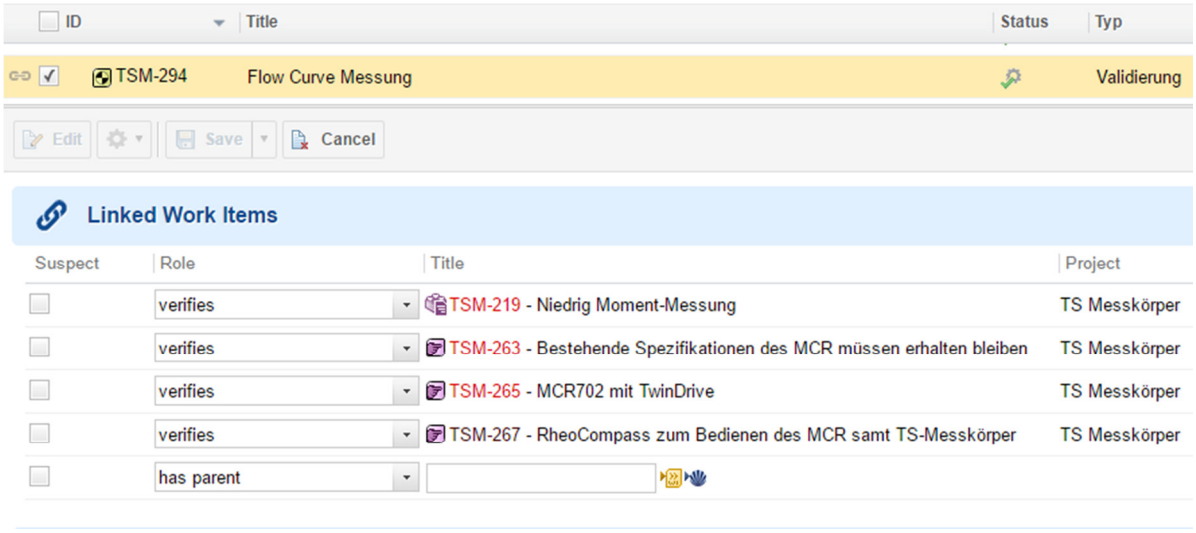


Abb. 40: Link bei einem Test-Case, Quelle: Eigene Darstellung.

Abb. 40 zeigt einen Ausschnitt des Test-Cases TSM-294 zur Validierung eines Use-Cases (TSM-219) und zwei User-Requirements (TSM-263 und TSM-265). Bei jedem Test wird im Feld *Typ* angegeben, ob es sich um eine Verifikation oder um eine Validierung handelt. Des Weiteren werden in der Beschreibung des Test-Cases die nötigen Hilfsmittel, Testschritte zur Durchführung des Tests und die Toleranzen für eine Erfolgreiche Durchführung angegeben.

Sind alle nötigen Test-Cases erstellt, werden diese in einen Test-Run zusammen geführt (siehe Abb. 41). Beim Hinzufügen kann nun nach dem Typ gefiltert werden, um beispielsweise einen Test-Run für eine Verifikation und einen für eine Validierung zu erstellen. Solange der Test-Run nicht gestartet wird, handelt es sich nur um eine Liste der durchzuführenden Tests, welche beschrieben wurden. Dadurch wird ein *Verifikations-* und *Validierungsplan* erstellt, welcher als *.pdf exportiert und für Reporting-Zwecke zur Verfügung steht.

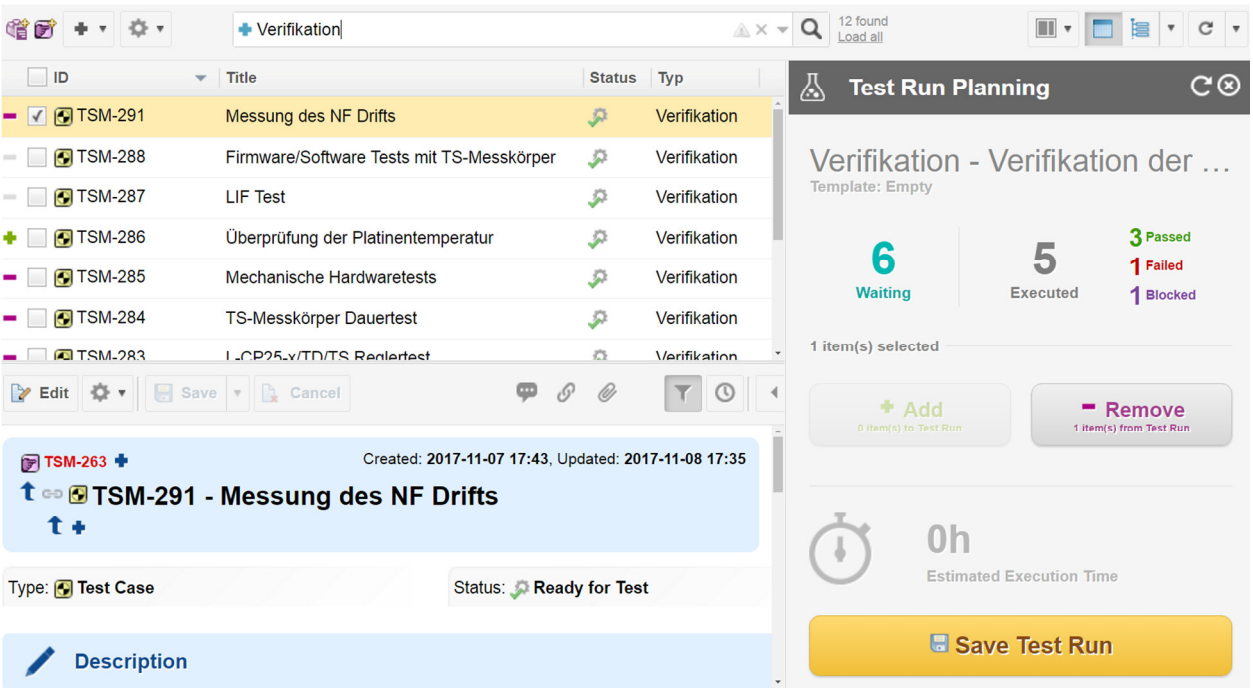


Abb. 41: Erstellung eines Test-Runs in Polarion, Quelle: Eigene Darstellung.

Bei der Zusammenstellung des Test-Runs in Abb. 41 wird ein Test-Run für die Verifikation erstellt. Die angezeigten Test-Cases können mit einem Plus oder Minus Symbol hinzugefügt oder wieder entfernt werden. In dieser Abbildung ist auch ersichtlich, dass der ausgewählte Test-Case beschrieben, überprüft und das Statussymbol für *Ready for Test* erhalten haben, sodass eine Aufnahme in den Test-Run gerechtfertigt ist. Der am unteren Ende links ersichtliche Bereich ist der Test an sich, welcher zu diesem Zeitpunkt auch eingesehen werden kann.

Wird der Test-Run ausgeführt, sind die vorher definierten Testschritte zu sehen. Diese sollen nun durchgeführt werden.

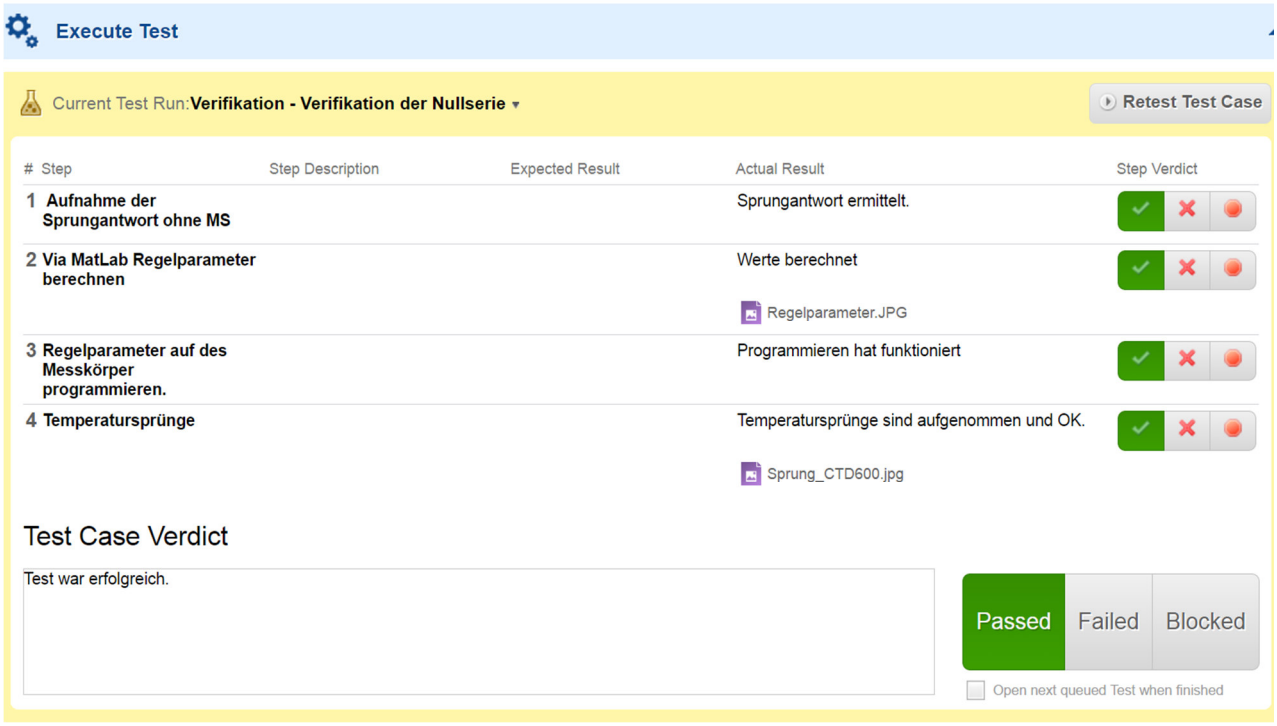
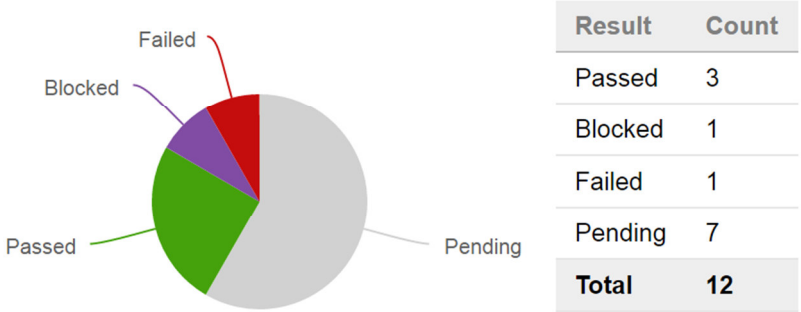


Abb. 42: Ergebnis eines ausgeführten Tests, Quelle: Eigene Darstellung.

Als Beispiel eines erfolgreich durchgeführten Verifikationstests zeigt Abb. 42 die ausgefüllten Ergebnisse eines Reglertests. Die Beschreibung der Testschritte und das erwartete Ergebnis sind schon bei der Erstellung des Tests ausgefüllt worden, aber aus Datenschutzgründen in dieser Abbildung entfernt worden. Nur das aktuelle Ergebnis wird bei der Ausführung noch eingegeben und mit *Passed*, *Fail* oder *Blocked* gekennzeichnet.

Mit der Software Polarion ist die Möglichkeit gegeben, dass Test-Runs mehrmals ausgeführt werden können. Das bedeutet, dass der gleiche Test-Run beispielsweise mit Prototypen in der Entwicklungsphase und mit Nullseriengeräten in der Überleitungsphase durchgeführt werden kann. Dazu kann jedes Testergebnis eines Test-Runs auch später eingesehen werden, sodass mögliche Unterschiede im Testergebnis miteinander verglichen werden können. Generell werden während und auch nach einem Test-Run alle Ergebnisse aller Tests gemeinsam in der Auswertung des Test-Runs angezeigt. Dies kann zur Steuerung des Projektergebnisses und für Reporting verwendet werden.



Result	Count
Passed	3
Blocked	1
Failed	1
Pending	7
Total	**12**

Abb. 43: Anfangsübersicht eines Test-Runs, Quelle: Eigene Darstellung.

Als erste Übersicht des Test-Runs wird eine Einteilung aller *Passed, Failed, Blocked* und noch nicht ausgeführten Test unter *Pending* gezeigt (siehe Abb. 43). Danach wird eine Auflistung aller Tests samt Beschreibung, Ergebnisse der Tests und allen Anhängen, welche zum Test hinzugefügt wurden erstellt. Diese spiegeln beispielsweise exakt die Ergebnisse der Angaben wie in Abb. 42 wider. Auch dieses Ergebnis kann als *.pdf exportiert, weiterverwendet oder abgelegt werden. Letztendlich können alle Status der Work-Items auf einmal nach dem angegeben Workflow verändert werden.

9.6 Dokumente des Requirement-Management aus Polarion

Wie schon im Kapitel 7.2.1 erwähnt, gibt es die Möglichkeit in der Software Polarion sogenannten Live-Documents zu erzeugen. In diesem Projekt sollen die Dokumente Rahmenvorgabe (Framework Specification) und die technische Spezifikation (Specification) anhand der Erkenntnisse aus dem Kapitel 7.1.4 erstellt werden. Zusätzlich werden die Work-Items, Use-Case und User-Requirements in die Rahmenvorgabe und System Requirements in das Spezifikationsdokument verlinkt. Das hat den Vorteil, dass im Dokument eine gesammelte Liste aller Anforderungen samt Beschreibung vorhanden ist. Bei Reporting-Tätigkeiten ist es beispielsweise notwendig nur mehr die Rahmenvorgabe zu präsentieren, um das aktuelle Vorhaben zu erklären, welche als *.pdf erstellt und auch gespeichert werden kann.

Die Inhalte der erstellten Dokumente sind in Bezug auf Anforderungen lt. dem Leitfaden aus Kapitel 8 wie folgt aufgeteilt:

- Rahmenvorgabe
    - Produktbeschreibung
    - Hauptanwendung
    - USP
    - Qualitätsmerkmale
    - Kompatibilität und Synergien mit anderen Baugruppen
    - Einbindung der vorhandenen Work-Items Use-Case und User Requirements
    - Use-Case Diagramm und Anforderungen lt. Kano-Modell aus Confluence
    - Nicht-Ziele
- Spezifikationsdokument
    - Mess- und Funktionsprinzip
    - Technische Spezifikation (Toleranzangaben der Temperaturmessung etc.)
    - Firmware Spezifikationen
    - Software Spezifikationen
    - Kontextabgrenzung, Zielmodell, Wirkkettenmodell und Sequenzdiagramm aus Confluence
    - Einbindung der Work-Items System Requirement

Die Erstellung der Dokumente in Polarion ist in Abb. 44 ersichtlich und grundsätzlich ähnlich anderen Schreibprogrammen. Es können genauso Überschriften als Kapitel, Bilder eingefügt, Work-Items und deren Text eingeblendet sowie auch die Links zu den Diagrammen in Confluence eingefügt werden. Als Nachteil ist aufgefallen, dass nicht ersichtlich ist, wann eine Seite endet (Endlos-Modus) und die Verwen-

dung von Formatvorlagen nicht möglich ist. Dies erschwert die Handhabung und Formatierung der Dokumente.

Aus Datenschutzgründen wurden die Texte des Dokuments in der Abb. 44 geschwärzt.

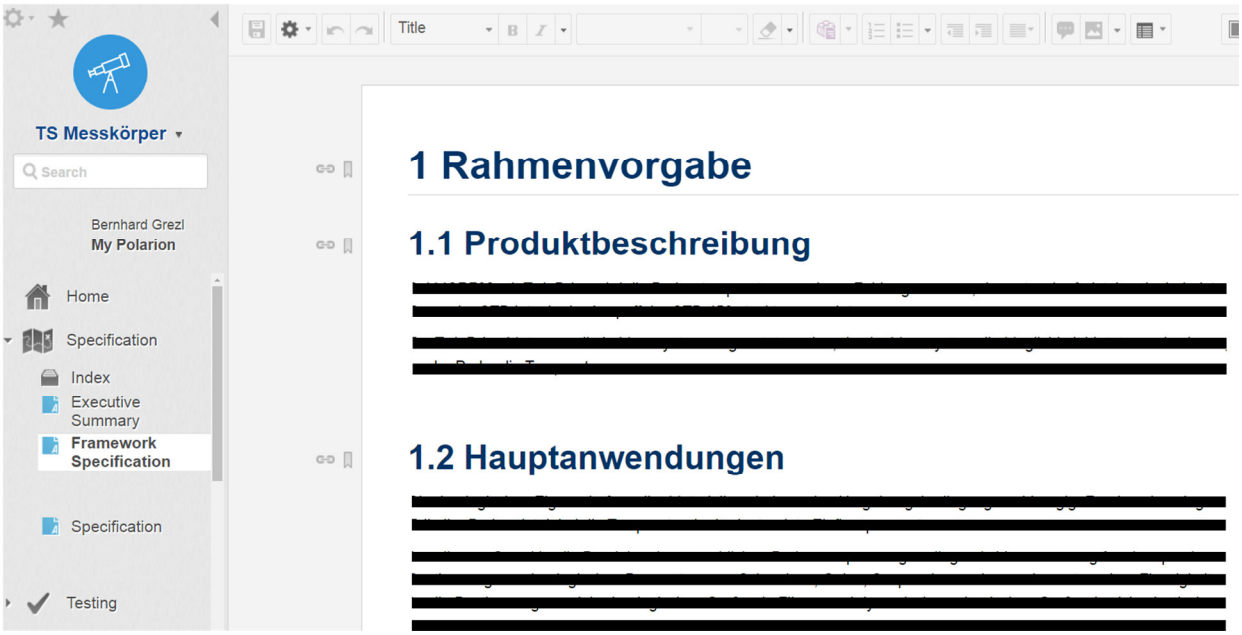


Abb. 44: Erstellung der Rahmenvorgabe in Polarion, Quelle: Eigene Darstellung.

9.7 Projektfortschritt in Polarion

Im Kapitel 4.7 ist schon erwähnt worden, dass das Anzeigen des Projektfortschritts anhand von Anforderungen möglich ist. Das bestätigt beispielsweise auch schon die Auswerte- bzw. Anzeigemöglichkeit bei einem Test-Run aus Abb. 43, welche den aktuellen Stand der Tests von Anforderungen widerspiegelt.

Polarion bietet eine Vielzahl an vorgefertigten Auswertungen als Widget (siehe Abb. 45) an, welche einen Überblick des aktuellen Projektfortschritts bieten. Es kann eine Webseite in Polarion generiert werden, in der diese Widgets implementiert werden können. Je nach Wunsch können verschiedenste Auswertungen angezeigt werden.

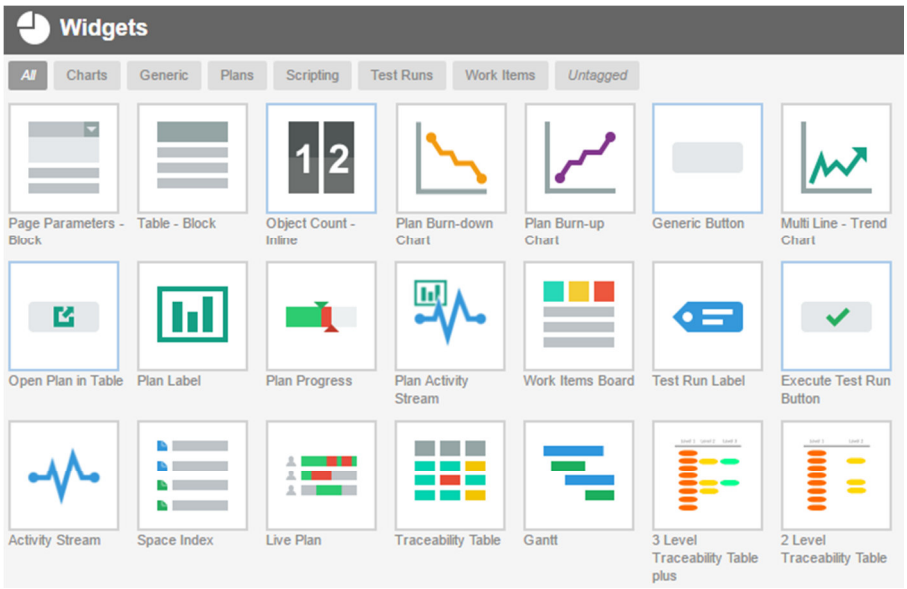


Abb. 45: Auszug der Widgets zur Auswertung in Polarion, Quelle: Eigene Darstellung.

Auswertungen, wie beispielsweise ein Burndown-Graph, Burnup-Graph, Anzeige des Planungsfortschritts uvm. sind dabei möglich.

Eine gute Übersicht bringt hier ein Burndown-Graph, welcher den Fortschritt übersichtlich in einem Diagramm anzeigen würde. Wie schon anfangs im Kapitel 9 erwähnt, ist das Projekt parallel zum echten Projekt erstellt worden. Daher kann die Abarbeitung der einzelnen Implementierungen nicht über Wochen oder Monate dargestellt werden. Da es aber weitere Auswertungsmöglichkeiten gibt, wurde mit dem *Traceability Table* eine Auswertung der Status der Use-Cases samt untergeordneten Work-Items erstellt.

ID-Title	Link Role	Status
TSM-223 - Fertigung der Messkörper		
TSM-266 - RheoServiceTool zum Programmieren der Messkörper	is refined by	Open
TSM-274 - Umsetzung des Fertigungsprozesses	is refined by	Ready for Test
TSM-268 - Konstruktion der TS-Messkörper	is refined by	Ready for Test
TSM-274 - Umsetzung des Fertigungsprozesses	is refined by	Ready for Test

Abb. 46: Ausschnitt der Status Auswertung der Use-Cases via Traceability Table, Quelle: Eigene Darstellung.

Der Ausschnitt des *Traceability Table* in Abb. 46 spiegelt die Auswertung der Status der Use-Cases und somit den aktuellen Implementierungsstand wider. Es ist eindeutig ersichtlich, welche Anforderungen noch offen sind (Status *Open*) und welche schon erfolgreich implementiert wurden (Status *Ready for Test*). Auch mit dieser Auswertung lässt sich der aktuelle Stand des Projekts leicht darstellen und gibt eine gute Einschätzungsmöglichkeit über die noch zu erledigenden Tätigkeiten. Wie alle Webseiten in Polarion, kann auch diese Auswertung als *.pdf erstellt werden und für Reporting weiter verwendet oder als Dokumentation abgelegt werden.

9.8 Fazit der Anwendung des Requirements-Managements

Das Erstellen der Diagramme in Confluence funktioniert mit dem vorhandenen UML unterstützenden Plug-In sehr gut. Das Verlinken nach Jira ist aufgrund dergleichen Herstellerfirma beider Softwareprogramme besser möglich als nach Polarion. Ein Link von Polarion nach Confluence ist nur als klassischer Weblink möglich, sodass auf die andere Webseite gesprungen wird und keine direkte Sicht auf das aktuelle Diagramm möglich ist. Trotzdem bieten die Modelle und Diagramme einen sehr guten Grundstein für die Ermittlung der Anforderungen. Bei Workshops mit dem Entwicklungsteam oder Erklärungen bei Stakeholder lässt sich die Thematik damit schneller erklären.

Die Anforderungen wurden in der Entwicklungsphase in Polarion aufgenommen, was einen großen Aufwand bedeutet, da viele Einzelheiten in verschiedenen Dokumenten vorhanden waren. Wenn das RM bereits von Anfang an eingesetzt wird, wachsen die Anforderungen und Dokumente mit dem aktuellen Projektstand mit und der Aufwand wird über einen längeren Zeitraum gestreckt. Dennoch wurde der aktuelle Implementierungsstand des Entwicklungsprojekts aufgezeigt und das Testen von Anforderungen in der Entwicklungsphase durchgeführt. Schon zu diesem Zeitpunkt ist es eine wesentliche Erleichterung einen Überblick über alle Tests und Implementierungen zu generieren, da nur mehr ein zentrales System gesichtet werden muss, ohne in verschiedenen Programmen oder Dokumenten zu suchen. Als Nachteil wird festgehalten, dass die Software Polarion teilweise umständlich funktioniert. Beispielsweise ist das Verlinken von Work-Items nicht gut gelöst, da immer nur ein Link nach dem anderen erstellt werden kann. Das mehrmalige Eingeben und Suchen des passenden weiteren Work-Item ist meist sehr mühsam und kostet sehr viel Zeit.

Das Erstellen der Live-Documents in Polarion ist grundsätzlich ähnlich wie bei Standardsoftwareprogrammen geregelt. Mit dem Vorteil, dass die Anforderungen als Work-Item in das Dokument gebracht werden können. Dadurch sind die Beschreibung des Projekts sowie die Auflistung aller Anforderungen zentral vorhanden. Mühsam gestaltet sich das Integrieren von Anforderungen in das Dokument, da nicht mehrere Anforderungen zu gleich in das Dokument eingefügt werden können, sondern nur eine Anforderung nach der anderen. Durch diesen Umstand und dem Fehlen von Formatvorlagen steigt auch hier der Aufwand für das Handling des Dokumentenerstellens.

Grundsätzlich ist der Einsatz des Requirements-Management-Leitfaden im Entwicklungsprojekt zufriedenstellen verlaufen. Insbesondere die Abwicklung der Tests ist positiv hervorzuheben. Aus diesem Grund wird in der folgenden Überleitungsphase das RM beibehalten und die abschließenden Validierungs- und Verifikationstests damit durchgeführt.

10 WIRTSCHAFTLICHKEIT DES REQUIREMENTS-MANAGEMENTS

Als wirtschaftliche Darstellung des RM ist eine Gegenüberstellung von Kosten erstellt worden. Damit soll im ersten Schritt gezeigt werden, was RM in einem Projekt der Firma Anton Paar GmbH kostet. Im zweiten Schritt werden Kosten aufgezeigt, welche entstehen, wenn beim genannten EPr eine Anforderung übersehen wird und welche Kosten durch den Mehraufwand entstehen können. Schlussendlich werden die beiden Aufwände verglichen und eine Schlussfolgerung daraus gezogen.

10.1 Kosten des Requirements-Managements in einem Entwicklungsprojekt

Die Kosten des RM beziehen sich auf die Kosten der Verwendung von RM in Kombination mit der Software Polarion. Jira und Confluence, die schon im Unternehmen eingesetzt werden und demnach keine zusätzlichen Kosten darstellen, die bei einem Entwicklungsprojekt anfallen. Lediglich der Zeitaufwand zur Erstellung der Diagramme kommt bei Confluence hinzu.

Für die Software Polarion werden verschiedene Lizenzen benötigt. Dabei wird nach dem Level der Benutzung (siehe Kapitel 7.2.1) und der Möglichkeit der Verteilung (named, floating) unterschieden. Je nach anfallender Lizenz sind dabei unterschiedliche Preise beim Hersteller zu bezahlen. Für die Firma Anton Paar GmbH sind die Lizenzen ALM, QA und RQ gekauft worden.

Mit der Formel 11.1 werden die Anschaffungskosten (AK) berechnet. Dazu wird die Summe der anfallenden Lizenzen mit dem jeweiligen Preis der Lizenz gebildet.

$$AK = \sum LA * LP \qquad (11.1)$$

AK/€	Anschaffungskosten
LA/Stück	Lizenzanzahl
LP/€	Lizenzpreis

In Tab. 6 sind zur Errechnung der AK der Software Polarion die Lizenzarten, die gekaufte Lizenzanzahl (LA) und Lizenzpreis (LP) aufgelistet. Die Summe der errechneten Kosten sind die gesamten Anschaffungskosten.

Es wird davon ausgegangen, dass die Anzahl der gekauften Lizenzen für die Verwendung aller Projekte der Firma Anton Paar GmbH ausreichen. Die Kalkulation müsste erneuert werden, wenn weitere Lizenzen zugekauft werden.

Lizenzart	Anzahl der gekauften Lizenzen	Kosten pro Lizenz	Errechnete Kosten
	In Stück	In €	In €
ALM named	2	2.760	**5.520,00**
ALM floating	2	9.640	**19.280,00**
QA named	1	2.040	**2.040,00**
QA floating	2	7.080	**14.160,00**
RQ named	4	1.620	**6.480,00**
RQ floating	7	5.620	**39.340,00**
		Anschaffungskosten:	**86.820,00**

Tab. 6: Anschaffungskosten der Software Polarion, Quelle: Eigene Darstellung.

Um daraus anteilige Software Kosten (ASK) pro Entwicklungsprojekt zu errechnen, werden eine Nutzungsdauer (ND) von 10 Jahren und 17 jährlich neu startenden Entwicklungsprojekte, welche RM mit Polarion einsetzen könnten, angegeben. Damit die Software immer einsatzbereit ist und nötige Anpassungen durchgeführt werden können, sind Ressourcen von 15 % der Anschaffungskosten pro Jahr in der Firma bereit gestellt. Mit der Formel 11.2 können nun anteilige Softwarekosten für jedes Entwicklungsprojekt (EPr) errechnet werden.

$$ASK = \frac{\frac{AK}{ND} + AK * 0{,}15}{EPr} = \frac{\frac{86820\ €}{10\ Jahre} + 86820\ € * 0{,}15}{17\ Projekte} = 1276{,}76\ {}^{€}/_{Projekt}$$

(11.2)

ASK/€Projekt^{-1}	Anteilige Softwarekosten
ND/Jahre	Nutzungsdauer
EPr/Stück	Entwicklungsprojekte

Um die zusätzlichen Aufwände für des RM im verwendeten EPr zu berechnen wurden die nötigen Arbeitszeiten für die jeweiligen Tätigkeiten ermittelt. Dabei sind Zeiten beim Erstellen der Anforderungen aufgezeichnet oder auch geschätzt worden. Die angegebenen Zeiten beziehen sich auf den Aufwand des angewandten Entwicklungsprojekts. Bei umfangreicheren Projekten steigt der Aufwand dementsprechend an, da mehr Anforderungen vorhanden sind.

Des Weiteren sind die internen Stundensätze der Rollen, welche normalerweise die Tätigkeiten durchführen, verwendet worden, um realistische Aufwände für ein RM im Entwicklungsprojekt zu errechnen.

Mit der Formel 11.3 werden alle Kosten der Aufwände für das RM des TS-Messkörperprojekts errechnet. Die anfallenden Tätigkeiten in Stunden werden dabei mit dem Stundensatz multipliziert und schlussendlich wird daraus eine Summe gebildet, um die Requirements-Management Kosten (RMK) zu erhalten.

$$RMK = \sum TK * StdS \qquad (11.3)$$

RMK/€ Requirements-Management Kosten

TK/h Tätigkeiten

StdS/€h^{-1} Stundensatz

In der Tab. 7 sind alle Tätigkeiten mit den Stundensätzen aufgelistet und die Kosten mit der Formel 11.3 berechnet. Einerseits sind es Kosten, die durch die Beschreibung der UC und der User-Requirements (U-RQ) entstehen. Andererseits kommt zusätzlicher Aufwand auch noch durch das Handling der Software (Einstellungen der Software, Zuweisungen, Auswertungen etc.) hinzu. Das Beschreiben der Dokumente Rahmenvorgabe und Spezifikationsdokument sind nicht als zusätzlicher Aufwand angeführt, da dieser Aufwand schon jetzt besteht. Es sind grundsätzlich keine Unterschiede beim Erstellen des Dokuments in Word oder wie bei RM in Polarion vorhanden.

Tätigkeit	Personenrolle	Zeitaufwand	Stundensatz	Errechnete Kosten
		In Stunden	In €/Stunde	In €
Erstellung der Diagramme und Modelle	PL	4,3	57,44	**246,99**
Erstellung der Diagramme und Modelle	SEn	15,3	70,79	**1.083,09**
Workshop für UC und U-RQ	PL	2,5	57,44	**143,60**
Workshop für UC und U-RQ	PM	2,5	53,58	**133,95**
Beschreibung der UC und U-RQ	PM	7,25	53,58	**388,46**
Beschreibung der UC und U-RQ	PS	5	53,28	**266,40**
Workshop für System RQ	PL	2,5	57,44	**143,60**
Workshop für System RQ	SEn	2,5	70,79	**176,98**
Beschreibung der System RQ	SEn	35,5	70,79	**2.513,05**
Beschreibung der Validierungstests	PS	4,52	53,28	**240,83**
Beschreibung der Verifikationstests	PE	8,4	43,46	**365,06**
Erstellung und Beschreibung der notwendigen Tasks zur Abarbeitung	PL	3,5	57,44	**201,04**
Handling Polarion	PL	25	57,44	**1.436,00**
Handling Polarion	SEn	25	70,79	**1.769,75**
			RMK:	**9108,80**

Tab. 7: Kosten des RM im Projekt, Quelle: Eigene Darstellung.

Um die Gesamten RMK inklusive der Kosten der Software zu errechnen, werden die Kosten anhand der Formel 11.4 addiert.

$$GK = RMK + ASK = 9108{,}80\,€ + 1276{,}76\,€ = 10.385{,}56\,€$$

(11.4) *GK*/€ Gesamtkosten des RM im Projekt

Das Ergebnis der Gesamtkosten (GK) mit 10.385,56 € spiegelt den finanziellen Aufwand des RM im TS-Messkörper Projekt wider. Da es sich bei diesem Projekt um eine Entwicklung eines Zubehörs für ein Messgerät handelt, ist verglichen mit einer Hauptgeräte-Entwicklung ein kleines Projekt (Aus Datenschutzgründen werden keine Gesamtkosten des TS-Messkörperprojekts und anderen Projekten der Firma Anton Paar GmbH in dieser Masterarbeit angegeben). Dennoch wird vermerkt, dass die Kosten des RM je nach Größe und Umfangs des Projekts noch weiter steigen können.

10.2 Kosten des Mehraufwandes bei einer übersehenen Anforderung

Bei fehlendem RM mit Polarion bei einem Entwicklungsprojekt, ist das Risiko einer übersehenden Anforderung höher, da die Anforderungen nicht für das ganze Team ersichtlich sind und daher kann es zu Fehlern in der Entwicklung kommen.

Um die Kosten einer übersehenen Anforderung aufzuzeigen, wird angenommen, dass die Anforderung der Einhaltung der Platinentemperatur (< 85 °C) im TS-Messkörper übersehen wird. Entdeckt wurde der Fehler bei Tests mit dem normalerweise fertigen TS-Messkörper. Dieser Fehler hat eine Reihe von Tätigkeiten zur Folge, die durchgeführt werden müssen. Zusätzlich müssen neue Prototypen gefertigt werden, um damit die neue Lösung zu verifizieren.

Die möglichen entstehenden Kosten der längeren Time-To-Market Zeit wird absichtlich vernachlässigt, da nicht beziffert werden kann, ob durch dieses Zubehör wirklich sofort mehr verkauft werden würde. Es handelt sich nur um eine Technologieerweiterung eines schon bestehenden Produkts. Deshalb liegt der Fokus nur auf die zusätzlichen Kosten des Aufwandes.

Um den Entwicklungsaufwand (EA) der neuen Lösung zu berechnen wird die Formel 11.5 verwendet, welche die Summe der Aufwände der Stunden mit den Stundensatz berechnet.

$$EA = \sum TK * StdS \quad (11.5)$$

EA/€ Entwicklungsaufwand

TK/h Tätigkeiten

StdS/€h^{-1} Stundensatz

Die Tab. 8 listet nun alle Tätigkeiten der Entw. , FertigungsmitarbeiterInnen (Fert.M), SpezialistInnen (SP) und Tätigkeiten des PE auf, welche für die Lösung der Anforderung benötigt werden.

Tätigkeit	Perso-nenrolle	Zeitaufwand	Stundensatz	Errechnete Kosten
		In Stunden	In €/Stunde	In €
Konstruktion				
Lösungsfindung	Entw	8	57,44	**459,52**
Erneuter Konstruktionsaufwand	Entw	14	57,44	**804,16**
Neue Zeichnungen	Entw	5	57,44	**287,20**
Konstruktion neuer Vorrichtungen	Entw	5	57,44	**287,20**
Layout				
Layout anpassen	SP	4	69,48	**277,92**
Dokumente erstellen	SP	4	69,48	**277,92**
Zusätzliche Tätigkeiten				
CNC Programmierung der neuen Teile	Fert.M	16	57,41	**918,56**
Neue Regelparameter für alle Temperier-kammern	SP	45	69,48	**3126,60**
Nochmalige Testwiederholung	PE	50	43,46	**2.173,00**
Umsetzung des neuen Fertigungsprozesses	PE	32	43,46	**1.390,72**
			EA:	**10.002,80**

Tab. 8: Kosten der Aufwände bei einer Neukonstruktion, Quelle: Eigene Darstellung.

Da neue Prototypen gefertigt werden müssen, kommen zu den Aufwänden noch Materialkosten (MK) der Prototypen, Vorrichtungen, Platinen und die Kosten für die Verschrottung der alten Platinen hinzu. Berechnet werden diese Kosten mit der Formel 11.6.

$$MGK = \sum MK * Stk \qquad (11.6)$$

MGK/€ Materialgesamtkosten

MK/h Materialkosten

Stk/Stück Stück

In der Tab. 9 sind alle Materialkosten aufgelistet, welche zur neuen Konstruktion der TS-Messkörper benötigt werden, damit die übersehene Anforderung implementiert ist und in Serie gefertigt werden kann.

Material	Materialkosten	Benötigten Stück	Errechnete Kosten
	In €/Stück	In Stück	In €
20 Stk. neue Vorrichtungen	54	20	**1.080,00**
5 Stk. neue Prototypen	394	5	**1.970,00**
Neue Platinen	41,91	12	**502,92**
Alte Platinen verschrotten	41,91	43	**1.802,13**
		MGK:	**5.355,05**

Tab. 9: Materialkosten der neuen Konstruktion, Quelle: Eigene Darstellung.

Es werden fünf neue Messkörper gefertigt, um den Nachweis der Lösung zu erbringen und eine gewisse Serienstreuung zu erkennen. Die 20 neuen Vorrichtungen sind für die spätere Serienfertigung und an die Losgrößen angepasst. Alte Platinen, die auf Lager sind, sind auf Grund der neuen Konstruktion nicht mehr zu verwenden und müssen verschrottet, bzw. neue Platinen gekauft werden. Die Mindestbestellmenge von 12 Stk. sind für die Tests notwendig.

Werden nun alle Kosten zur Implementierung der übersehenen Anforderung mit der Formel 11.7 addiert, werden die Gesamtaufwandskosten (GAK) errechnet, welche bei der Lösung der übersehenen Anforderung entstehen.

$$GAK = EA + MGK = 10.002{,}80\ € + 5.355{,}05\ € = 15.357{,}85\ €$$

(11.7) *GAK/€* Gesamtaufwandskosten

Die GAK in Höhe von 15.357,85 € spiegeln die Kosten der Auswirkung wider, welche entstehen können, wenn die Anforderung zur Einhaltung der Platinentemperatur übersehen wird.

10.3 Interpretation der wirtschaftlichen Ergebnisse

Ein Vergleich der Kosten des RM von 10.385,56 € im TS-Messkörperprojekt und den Kosten einer neuen Konstruktion auf Grund einer übersehenen Anforderung von 15.357,85 € wird so interpretiert, dass ein RM zwar fixe Kosten und somit einen Mehraufwand im Projekt verursacht, dadurch aber das Risiko einer späteren, teureren Neukonstruktion vermieden wird. Zusätzlich fällt das RM günstiger als die komplette Nacharbeit aus.

11 ZUSAMMENFASSUNG UND AUSBLICK

Die Ist-Stands-Analyse der Dokumente im aktuellen PEP hat die schon vorhandenen Grundzüge des RM im PEP gezeigt. Dafür sind Empfehlungen zur Umgestaltung zusammengestellt worden, wie die Dokumente bei einer Verwendung von RM ausgeführt werden können. Die Analyse der bestehenden Projekte, welche die Software *Polarion* genutzt haben, hat gezeigt, dass keine gleiche Verwendung der Software stattfindet. Durch die noch fehlende standardisierte Vorgabe ist die Software durch ihre individuelle Gestaltungsmöglichkeit sehr unterschiedlich eingesetzt worden. Dabei ist auch aufgefallen, dass eine mechatronische Produktentwicklung andere Anforderungen, wie ein reines Software- respektive Firmware-Entwicklungsprojekt an eine RM-Software hat.

Aufgrund der theoretischen Grundlage dieser Masterarbeit und der Erkenntnisse der Analysen ist ein Requirements-Management-Leitfaden entwickelt worden, welcher das RM bei einer mechatronischen Produktentwicklung vorgibt. Das im Leitfaden entwickelte Requirements V-Modell ist als Informationsplakat gedacht und bietet einen Überblick über die Zusammenhänge von Anforderungen, Abarbeitungstasks, allen Tests bis hin zur Verwendung von allen Software-Werkzeugen (Polarion, Jira und Confluence) samt Modellen und Diagrammen. Durch die Definition der Anforderungskategorien und der hierarchischen Verknüpfungsmöglichkeit ist jede Anforderung auf ihren Ursprung bzw. dessen Nutzen rückführbar. Somit wird sichergestellt, dass das Produkt immer für einen speziellen Kundennutzen entwickelt wird.

Der Einsatz des Requirements-Leitfadens in einem laufenden Produktentwicklungsprojekt hat gezeigt, dass der Requirements-Management-Leitfaden für eine mechatronische Produktentwicklung funktioniert und *Polarion* in Kombination mit schon länger bestehenden Software-Werkzeugen dafür verwendet werden kann. Durch den Einsatz von *Polarion* ist eine äußerst gute Übersicht aller Anforderungen immer gegeben und auch der Projektfortschritt lässt sich analysieren. Des Weiteren können die nötigen PEP Dokumente gleich direkt in dieser Software erzeugt werden, die auch exportiert und archiviert werden können. Als Nachteil werden die schlechte Formatierungsmöglichkeit anhand der fehlenden Formatvorlagen und das manchmal umständliche Handling gesehen. Modelle und Diagramme in der Software *Confluence* unterstützen das ganze Entwicklungsprojekt und helfen Stakeholder mit wenig Aufwand einen Überblick der betroffenen Komponenten, Anwendungen und Implementierungen zu geben.

Die wirtschaftliche Darstellung zeigt, dass RM einen Mehraufwand und somit zusätzliche Kosten verursacht. Die Gegenüberstellung des Aufwandes bei einer übersehenen Anforderung zeigt dennoch, dass das RM im Vergleich dazu jedoch günstiger ausfällt.

Durch die strukturierte Ermittlung und Handhabung der Anforderungen in einem mechatronischen Produktentwicklungsprojekt wird das Risiko einer fehlerhaften Entwicklung minimiert. In Kombination mit Software-Werkzeugen kann durch die Erstellung von Modellen und Diagrammen eine Hilfestellung und Übersichtsgenerierung geschaffen werden, die rein textuell aufwändiger wäre. Der Überblick über alle Anforderungen, insbesondere der aktuelle Status aller Implementierungen, ist durch die Verwendung von *Polarion* sehr gut gegeben. Darüber hinaus ist das Handling der Tests sehr übersichtlich. Durch die Möglichkeit der mehrmaligen Durchführung der Test-Runs können Testergebnisse in verschiedenen Entwick-

lungsphasen verglichen werden, die wiederum Aufschlüsse über beispielsweise Konstruktionsänderungen geben. Der Mehraufwand des RM sollte investiert werden, da die gewonnenen Zusatzinformationen aller Anforderungen einen Mehrwert über die Laufzeit eines Entwicklungsprojekts gibt.

Der entwickelte Requirements-Management-Leitfaden stellt eine hohe Qualität der Anforderungen dar. Zusätzlich sind das Management der Anforderungen während der ganzen Projektlaufzeit und vor allem die kundenorientierte Produktentwicklung dadurch gesichert.

In weiterer Folge werden die Ergebnisse der Analysen und der Masterarbeit den Verantwortlichen der Firma Anton Paar GmbH vorgelegt. Diese Personen werden danach entscheiden, welche Änderungen im PEP für den Einsatz des RM durchgeführt werden. Des Weiteren soll diese Masterarbeit als Informationsquelle für Personen, welche sich mit Produktentwicklung beschäftigen und sich mit RM noch nicht stark auseinandergesetzt haben, dienen.

Die Einführung von einer standardisierten Verwendung von *Polarion* soll mit Zuhilfenahme des entwickelten Requirements V-Modells stattfinden. Mit der Zusammenarbeit von erfahrenen Projektleitern kann so ein Template zur allgemeinen Verwendung erstellt werden.

Derzeit sind einige Produktentwicklungsprojekte in der Spezifikationsphase. Bei einem davon wird bereits versucht *Polarion* in Bezug auf das Requirements V-Modell einzusetzen. Dieser Versuch in einem größeren Entwicklungsprojekt wird dem Entwicklungsleiter in weiterer Folge vorgelegt und danach wird entschieden, ob diese Variante über das ganze Projekt hinweg weiter verfolgt werden soll.

LITERATURVERZEICHNIS

Gedruckte Werke (15)

Department of Defense (Hrsg.) (1969): *MIL-STD-499 - System Engineering Management*

Verein Deutscher Ingenieure (Hrsg.) (2004): *VDI 2206: Entwicklungsmethodik für mechatronische Systeme.*

Schweizerische Normen-Vereinigung (SNV) (Hrsg.) (2015): *EN ISO 9001: Qualitätsmanagementsysteme - Anforderungen*

ISO copyright office; Institute of Electrical and Electronics Engineers, Inc. (Hrsg.) (2011): *ISO/IEC/IEEE 24765:2010(E): Systems and software engineering - Vocabulary*

Alt, Oliver (2012): *Modellbasierte Systementwicklung mit SysML*, Carl Hanser Fachbuchverlag, München

Blanchard, Benjamin S.; Blyler, John E. (2016): *System Engineering Management* , 5 Auflage, John Wiley & Sons, Inc. , New Jersey

Eigner, Martin; Roubanov, Daniil; Zafirov, Radoslav (2014): *Modellbasierte virtuelle Produktentwicklung* , Springe-Verlag Berlin Heidelberg , Berlin

Graner, Marc (2015): *Methodeneinsatz in der Produktentwicklung: Bessere Produkte, schnellere Enwicklung, höhere Gewinnmargen*, Springer Fachmedien Wiesbaden, Frankfurt am Main

Herrmann, Andrea ; Knauss, Eric; Weißbach, Rüdiger (2013): *Requirements Engineering und Projektmanagement*, Springer-Verlag Berlin Heidelberg, Berlin

Hruschka, Peter (2014): *Business Analysis und Requirements Engineering: Produkte und Prozesse nachhaltig verbessern*, Carl Hanser Verlag München, München

Leffingwell, Dean (2011): *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*, Pearson Education, Inc. , Boston

Pohl, Klaus; Rupp, Chris (2015): *Basiswissen Requirements Engineering: Aus- und Weiterbildung zum >> Certified Professional for Requirements Engineering*<<, 4 Auflage, dpunkt.verlag GmbH, Heidelberg

Rupp, Chris (2014): *Requirements-Engineering und -Management: Aus der Praxis von klassisch bis agil*, 6 Auflage, Carl Hanser Verlag München, Nürnberg

Stachowiak, Herbert (1973): *Allgemeine Modelltheorie*, Springer-Verlag, Wien

Weilkiens, Tim; Lamm, Jesko G.; Roth, Stephan; Walker, Markus (2016): *Model-Based System Architecture*, John Wiley & Sons, Inc. , New Jersey

Online-Quellen (7)

Anton Paar GmbH (2017): *Anton Paar*
https://www.anton-paar.com/at-de/ueber-uns/ [Stand: 21.08.2017]

Anton Paar GmbH (2017): *Rheometer*
https://www.anton-paar.com/at-de/produkte/gruppe/rheometer/ [Stand: 15.09.2017]

Anton Paar GmbH (2017): *World of Rheology*
http://www.world-of-rheology.com/de/ [Stand: 15.09.2017]

Anton Paar GmbH (2017): *Rheometer MCR 702 TwinDrive*
https://www.anton-paar.com/corp-en/products/details/rheometer-mcr-702-twindrive/ [Stand: 16.10.2017]

Atlassian (2017): *Jira Software*
https://de.atlassian.com/software/jira [Stand: 08.10.2017]

Atlassian (2017): *Confluence*
https://de.atlassian.com/software/confluence [Stand: 08.10.2017]

Siemens Industry Software GmbH (2017): *Polarion ALM*
https://polarion.plm.automation.siemens.com/products/polarion-alm [Stand: 06.10.2017]

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

ADC	Analog Digital Converter
AK	Anschaffungskosten
ASK	anteilige Software Kosten
CTD	Convection Temperature Device
EA	Entwicklungsaufwand
EC-Motor	Electronically Commutated Motor
Entw	EntwicklerIn
EP	Externer Partner
EPr	Entwicklungsprojekt
FD	Firmware Developer
Fert.M	FertigungsmitarbeiterIn
GAK	Gesamtaufwandskosten
GK	Gesamtkosten
HIL	Hardware-in-the-Loop
LA	Lizenzanzahl
LP	Lizenzpreis
MGK	Materialgesamtkosten
MK	Materialkosten
ND	Nutzungsdauer
PAG	ProjektauftraggeberIn
PE	Product Engineer
PEP	Produktentwicklungsprozess
PL	ProjektleiterIn
PM	Product ManagerIn
PO	Product Owner
PS	Product Specialist
R&D	Research and Development
RE	Requirements-Engineering
RFID	Radio Frequency Identification

RM	Requirements-Management
RMK	Requirements Management Kosten
RQ	Requirement
SD	Software Developer
SE	Systems Engineering
SEn	System Engineer
SIL	Software-in-the-Loop
ST	Software TesterIn
Stk	Stück
SUC	System-Use-Case
SUCD	System-Use-Case-Diagramm
SysML	Systems Modeling Language
TD	Technical Documentation
TE	Test-Engineer
TS	Temperatursensor
UC	Use-Case
UCD	Use-Case-Diagramm
UML	Unified Modeling Language
U-RQ	User-Requirement
USP	Unique Selling Proposition

ANHANG 1: SEQUENZDIAGRAMM DES STANDARD-MESSKÖRPERS

Detailliertes Sequenzdiagramm der Messkörpererkennung eines Standard-Messkörpers aus Kapitels 9.2.5.

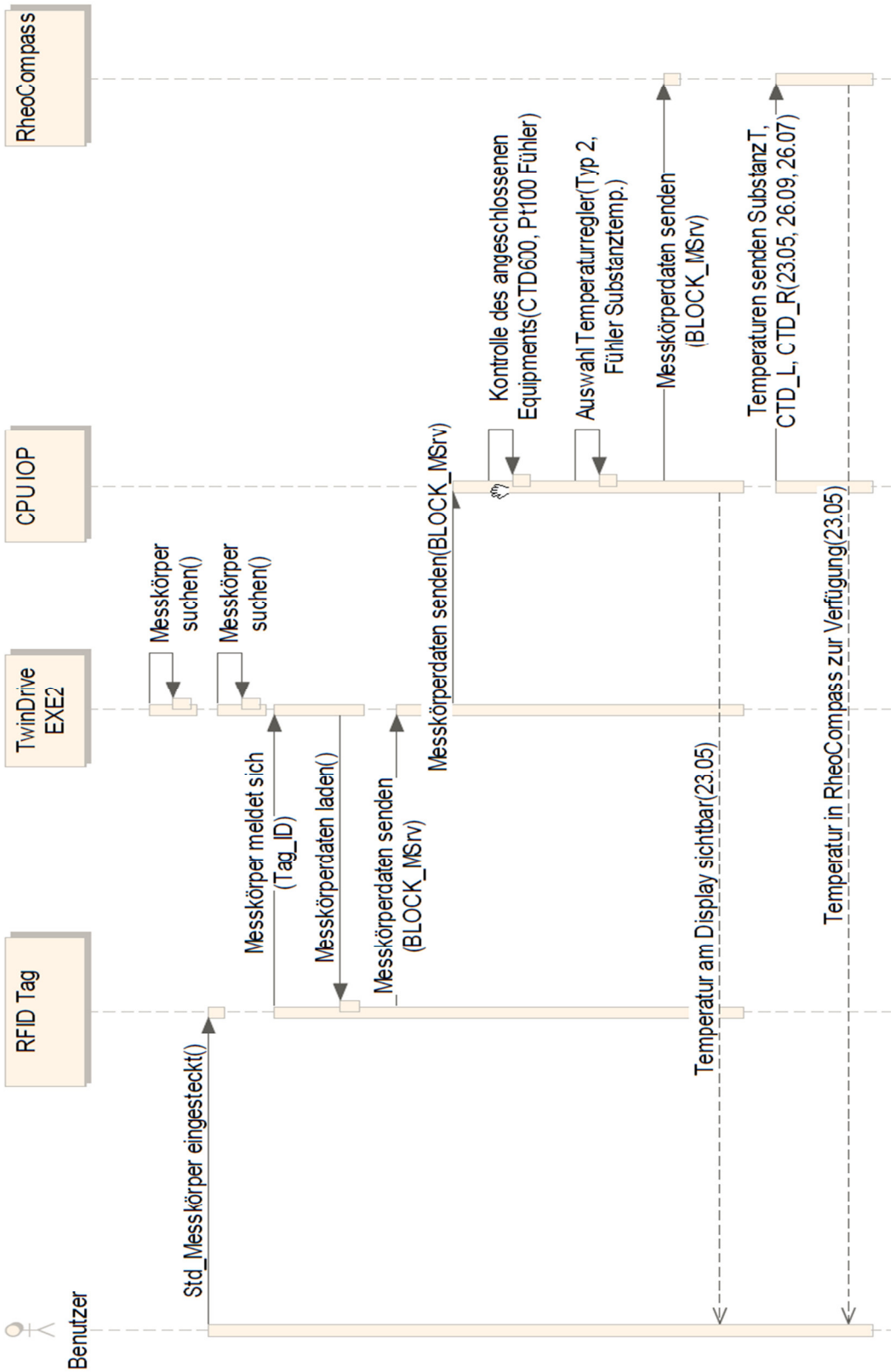


Abb. 47: Detailliertes Sequenzdiagramm des Standard-Messkörpers, Quelle: Eigene Darstellung.

ANHANG 2: SEQUENZDIAGRAMM DES ZU ENTWICKELNDEN MESS-KÖRPERS

Detailliertes Sequenzdiagramm der Messkörpererkennung eines TS-Messkörpers aus Kapitels 9.2.5.

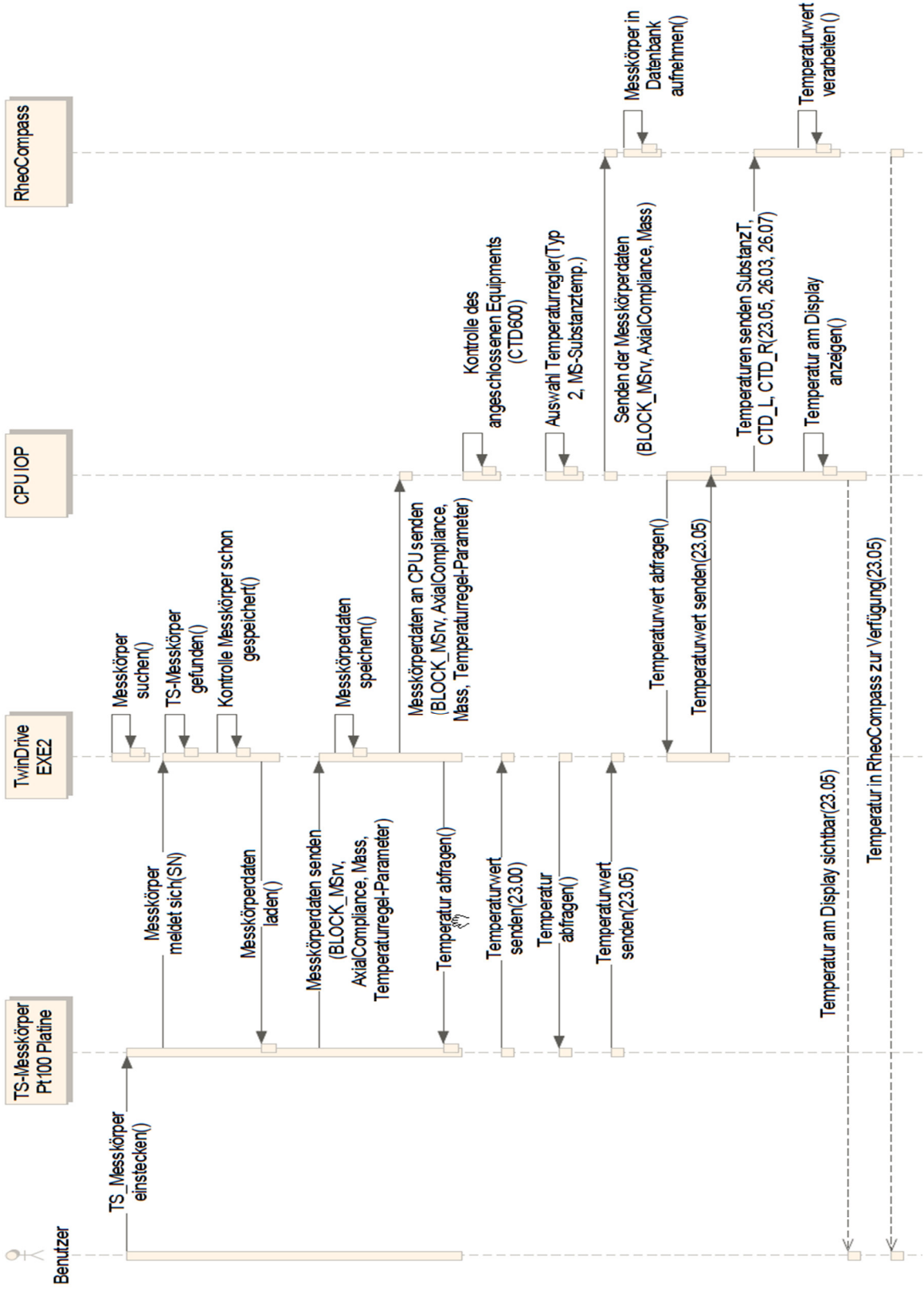


Abb. 48: Detailliertes Sequenzdiagramm des TS-Messkörpers, Quelle: Eigene Darstellung.