MASTERARBEIT

FRAMEWORK FÜR DIE QUALITÄTSSICHERUNG IM RELEASE MANAGEMENT MIT JIRA UND DEVOPS

ausgeführt am

Studiengang

Informationstechnologien und Wirtschaftsinformatik

Von: Christian Egger

Personenkennzeichen: 2010320030

Graz, am 30. März 2022 .....

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

Zu Beginn möchte ich mich bei allen Personen bedanken, die mich während der Erstellung dieser Masterarbeit unterstützt haben. Ein besonderer Dank geht an Herrn Dr. Dipl. Ing. Puchleitner Thomas, der diese Masterarbeit betreut und begutachtet hat. An dieser Stelle möchte ich mich für jegliche Unterstützung in Form von Ideen, Anregungen sowie der konstruktiven Kritik, die bei der Anfertigung dieser Arbeit eine große Hilfestellung dargestellt hat, bedanken.

Weiteres möchte ich mich bei meinen Vorgesetzten im Unternehmen in Person von Herrn Leiner Jakob als Bereichsleiter und Herrn Peischl Mario als Teamleiter bedanken, die es ermöglicht haben, eine einfache Koordination des Studiums und der Arbeit handzuhaben.

Ein besonderer Dank gilt allen Teilnehmern meiner qualitativen Forschung, mithilfe deren Informationen diese Arbeit erst entstehen hat können. Vielen Dank für die Bereitstellung der Erfahrungen sowie des Wissens über dieses Thema.

Außerdem möchte ich mich bei den Korrekturleserin bedanken, die mich bei der Suche nach Fehlern unterstützt haben.

Zu guter Letzt bedanke ich mich von ganzen Herzen bei meiner Familie und meinen Freunden, die mich auf meinem gesamten Studienweg unterstützt haben und mir dadurch sehr geholfen haben.

Auch meinen Studienkollegen und auch den Vortragenden gehört ein Dank ausgesprochen. Einerseits den Lehrenden für jeglichen Inhalt, der mir bis heute beigebracht wurde und andererseits auch den Studienkollegen, mit denen im Studium interessante und wertvolle Erfahrungen ausgetauscht worden sind.

KURZFASSUNG

Release Management spielt in der Softwareentwicklung eine große Rolle. Der Prozess von einer Anfrage zu einer Änderung bis zur Ausrollung der Änderung auf das Live-System behandelt alle Themen, die in dem Bereich des Release Managements fallen. Dass dieser Prozess einen großen Einfluss auf die Qualität der Software und den Erfolg des Unternehmens hat, liegt demzufolge auf der Hand. Aufgrund der Konkurrenzsituation am Markt streben IT-Unternehmen nach dem bestmöglichen Prozess.

An diesem Punkt kommen Tools wie Jira aus der Atlassian Suite oder Azure DevOps aus dem Microsoft-Portfolio ins Spiel. Diese Tools unterstützen Unternehmen in den Bereichen Issue-Tracking, Taskplanung, sowie weitere Themen, die im Release Management gebraucht werden. Nichtsdestotrotz treten bei vielen IT-Unternehmen nach wie vor Probleme auf, die für die Qualität des Release Managements negative Auswirkungen haben und verbessert werden müssen.

In dieser Arbeit wird mithilfe einer Literaturrecherche über die Probleme in Verbindung mit Toolanalysen sowie einer qualitativen Forschung mit Experten aus dem Bereich des Release Management und in der Softwareentwicklung erforscht, ob die gefundenen Probleme im Kontext des Tools auftreten oder, ob diese durch weitere Handlungsempfehlungen gelöst werden können. Aufgrund der Tatsache, dass die Softwareentwicklung in den letzten Jahren eine Entwicklung eingeschlagen hat, in der der Softwareentwicklungsprozess sich in den meisten Unternehmen mehr in Richtung eines agilen Ansatzes geändert hat, wird in folgender Arbeit die Auswirkung dieses Trends auf das Release Management analysiert.

Zu guter Letzt werden einer Gruppe an Unternehmen Handlungsempfehlungen erstellt, die bei Eintritt bestimmter Probleme verwendet werden können, um auch in Verbindung mit dem Tool für eine höhere Qualität im Release Management zu sorgen.

ABSTRACT

High-quality release management is key to a successful software product. Concepts and strategies for software have changed. Terms like agile and DevOps have had an enormous impact and closing the gap between operational and development has already been achieved in some cases. Azure DevOps or Jira as a core Atlassian Suite software product are widely used in the industry. Nevertheless, some struggle to manage software development while others already enjoy more rapid production of better software. Although the supporting tools are already powerful productivity and collaboration tools in software-release development. IT companies still face many challenging and unexpected barriers during a deployment. Solutions can be time-intense and challenging. This thesis explores the evolution of software development and describes the challenges IT companies face. Combined with an analysis of the supporting tools and interviews with experts, a guide for mid-sized IT companies is produced that offers measures to analyse current processes and recommend changes to increase release management quality.

INHALTSVERZEICHNIS

1 EINLEITUNG

Die Qualität einer Software steht eng im Zusammenhang mit einem guten Release Management. Projekte werden immer größer und Schnittstellen häufen sich rapide. Selbst Experten fällt es immer schwerer, einen Überblick über alle Komponenten einer Applikation zu bewahren. Aus diesem Grund rückt das Release Management in den letzten Jahren immer mehr in den Vordergrund.

Mittlerweile gibt es in der Praxis Tools, die Unternehmen bei Problembehebungen im Release Management unterstützen versuchen. Vom australischen Unternehmen Atlassian hat sich in den letzten Jahren eine Service-Plattform im IT-Sektor etabliert, die mit ihren Funktionen das Release Management unterstützt. Im Fokus des Release Managements liegen von der Atlassian Suite die Softwareprodukte rund um Jira, Confluence und Bitbucket, welche im gemeinsamen Einsatz bereits einen großen Bereich der Releaseplanung und -durchführung abdecken.

DevOps, ein Begriff aus der Softwareentwicklung, ist in den letzten Jahren stark in den Vordergrund gerückt und hat die Anforderungen an das Release Management erhöht. Die Verbindung organisatorischer Arbeit mit der Softwareentwicklung, welche im Release Management entscheidende Faktoren für den Erfolg des Produkts darstellen, wird in dieser Bezeichnung dargestellt. Von Microsoft ist für diesen Bereich Azure DevOps entwickelt worden, was auch in vielen Unternehmen Verwendung findet, um vor allem im Release Management zu unterstützen.

Nichtsdestotrotz gibt es noch Probleme im Release Management, die vor allem in Hinsicht auf die steigenden Qualitätserwartungen einer Software behoben werden müssen, um mit der Konkurrenz im IT-Sektor mithalten zu können. Die Ursachen liegen hier in verschiedenen Bereichen, die in dieser Arbeit analysiert werden. In diesem Zusammenhang werden Unternehmen analysiert, die die erwähnten Tools rund um die Atlassian Suite und Azure DevOps verwenden, um Probleme im Release Management zu erkennen. Außerdem werden Ansätze erforscht, die diese Probleme mit der Unterstützung dieser Tools beheben können. Basierend darauf wird am Ende dieser Arbeit die Forschungsfrage *„Wie setzen mittelständische IT-Unternehmen die Tools Jira und Azure DevOps ein, um die Qualität des Release Managements zu steigern?“* beantwortet und dargestellt.

In der Arbeit wird auf eine Übersetzung der englischsprachigen Terminologie der Softwareentwicklung verzechtet und es werden die Original-Begriffe der Fachliteratur zu finden sein. Weiteres zu erwähnen ist, dass in der vorliegenden Arbeit auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet wird. Es wird das generische Maskulinum verwendet, wobei die Geschlechter gleichermaßen gemeint sind.

1.1 Vorwort

Trotz der bereits vorhandenen Tools zur Unterstützung des Release Managements haben viele Unternehmen noch Probleme bei der Durchführung eines Releases. Vielerlei Begriffe wie Agiles Projektmanagement, Continuous Delivery and Continuous Integration entwickelten sich in den letzten Jahren zu zentralen Punkten, die heute im IT-Sektor im Fokus stehen. Diese Themen stehen eng im Zusammenhang mit dem Release Management und tragen einen großen Anteil an der Entwicklung neuer Softwareprodukte bei.

Einige Unternehmen haben bereits Prozessänderungen abgeschlossen und Teams auf die steigenden Anforderungen an Softwaretools vorbereitet. Viele weitere Unternehmen sind bereits auf den Zug aufgesprungen und befinden sich inmitten dieser Umstellung. Während der Einführung, sowie bei der Durchführung eines Prozesses für den Release stoßen Teams trotzdem immer noch auf Probleme, die die Qualität der Software mindern und zeitlich sowie finanziell einen negativen Einfluss auf das Projekt haben. Interviews mit Experten aus diesem Bereich, die in verschiedenen Stadien der Umsetzung und Einführung des Release Managements stehen, geben näheren Aufschluss über die Meilensteine und Probleme des Release Managements und bieten Informationen zur Nutzung der bereits vorhandenen Tools, die in diesem Gebiet eingesetzt werden können. Diese qualitative Forschung der Arbeit bietet den zukünftigen Lesern einen Leitfaden, der als Basis für das Release Management in Zusammenhang mit den vorhandenen Tools dient.

1.2 Themeneinleitung Release Management

Release Management spielt in der heutigen Zeit vor allem in der IT-Branche eine große Rolle. Softwareprodukte wachsen mit der Zeit und Schnittstellen zu weiteren Komponenten häufen sich in regelmäßigen Abständen. Nichtsdestotrotz ist die Qualität nach wie vor essenziell für den Erfolg einer Software. Um diese Komplexität zu meistern, ist ein hochqualitatives Release Management notwendig (Michlmayr et al. 2005). Vor allem das Thema rund um die Automatisierung wird seit einigen Jahren immer relevanter. In der Praxis haben es Unternehmen jedoch nach wie vor schwer, ihre Releases erfolgreich zu planen und durchzuführen. Auf dem Markt gibt es bereits Tools, die konkret Unterstützungen anbieten, um genau diese Probleme zu vermeiden. Trotzdem haben Unternehmen noch Schwierigkeiten das Release Management auf das gewünschte Niveau zu bringen. Warum diese Probleme auftauchen und auf welche Herausforderungen Unternehmen sowohl bei der Einführung eines Release Managements als auch bei der praktischen Umsetzung stoßen, wird im Rahmen dieser Arbeit erläutert.

2 THEORETISCHER RAHMEN

Der Best-Practice-Leitfaden ITIL (Information Technology Infrastructure Library), der in der Praxis oft als Orientierung dient, definiert den Release und Deployment Prozess als die „Planung und Kontrolle der Entwicklung und Veröffentlichung eines Releases, ohne die Integrität der funktionierenden Software zu gefährden.“ Das Release Management ist also dafür verantwortlich, dass bei allen Änderungen die Funktionalität des bereits vorhandenen Systems sichergestellt wird (Shanmugasundaram and Sojini 2018).

Bei der erwähnten ITIL handelt sich um eine speziell für IT-Unternehmen erstellte Sammlung von Best Practice-Prozessen und Erfahrungsberichten, die von vielen IT-Organisationen entwickelt wurde. Unternehmen können sich an den in ITIL definierten Prozess orientieren und diese auch optimieren. Weiteres regen die Empfehlungen zum Umsetzen die Denkprozesse der Führungskräfte an, da verschiedene und auch funktionierende Vorgehensweisen im Unternehmen erklärt sind. Nicht zu vergessen ist außerdem der Punkt, dass in ITIL eine allgemeine Terminologie definiert ist, damit in der gesamten IT-Branche eine einheitliche Sprache verwendet wird.

Die Komplexität der Softwareprodukte hat in den letzten Jahren stark zugenommen und ein Eingriff in die am Produktivsystem verwendete Softaware muss mithilfe des Release Managements organisiert werden. Um nicht jede Systemänderung in einem eigenen Prozess zu geben, erstellt man ein Paket aus mehreren einzelnen Änderungen, die das gleiche System betreffen. In der Praxis spricht man hier von einem Paket aus „Changes“, ein sogenannter Release (Amir et al. 2011). Laut ITIL wird bereits bei der Änderung oder Hinzufügung einer Konfiguration von einem Release gesprochen. Das Thema rund um Konfigurationen wird in Kapitel 2.1.2 detaillierter beschrieben.

2.1 Aufgaben des Release Managements

Die Hauptaufgabe des Release Managements besteht aus der Planung eines Deployment mit neuen Features auf das Produktivsystem, ohne die Sicherheit am laufenden Betrieb zu gefährden. Jeglicher Eingriff auf das Produktivsystem bedeutet ein Deployment, weswegen jede Änderung am laufenden System einen bestimmten Prozess durchlaufen sollte. Dieser dient als Schutz, damit ungeplante Änderungen das laufende System nicht stören. Unter dem Produktivsystem versteht man in der IT ein System, auf dem die tägliche Arbeit der Mitarbeiter stattfindet. Um mögliche Ausfälle zu verhindern, versucht das Release Management jegliche Eingriffe, in Form von Änderungen, Hinzufügungen oder Entfernungen eines Teiles des Systems, so zu planen, dass am Produktivsystem ein fehlerfreier Betrieb möglich ist (Elsässer 2005).

Zusammen mit dem Release Management werden oft das Change Management sowie das Configuration Management erwähnt. Dies führt in der Theorie oft zu Verwirrung, da diese drei Bereiche Ähnlichkeiten aufweisen und auch stark ineinandergreifen. Nichtsdestotrotz gibt es hier klare Definitionen und Abgrenzungen, welche in den nächsten Abschnitten nähergebracht werden (Rasa G., Jagadheesh Kumar, Banu Wahida 2010).

2.1.1 **Change Management**

Ein Paket aus Changes stellt ein Release dar. Das Change Management selbst jedoch ist ein vom Release Management getrennter Bereich. ITIL definiert einen Change als „Zugabe, Änderung oder Entfernung einer Softwarekomponente, die einen Einfluss auf das IT-System hat." Ein Change ist jedoch der Hauptgrund, warum Fehler in einem System auftreten. Keine Änderungen am System zu erledigen, ist jedoch auch keine Lösung, da diese Variante einen noch schlechteren Einfluss auf die Kundenzufriedenheit haben kann (Rance 2011).

Das Change Management befasst sich mit jeglichen Änderungen, die am Produktivsystem durchgeführt werden. Eine detaillierte Dokumentation, ausreichende Testungen sowie eine Risikoabschätzung und ein passender Plan, den Change einzuspielen, führen zu einem hochqualitativen Change Management. Die Umsetzung im Unternehmen ist jedoch weit komplexer und es würde mehr Änderungen am laufenden System mit sich bringen, ein solch detailliertes Change Management im laufenden Betrieb einzuführen. Aus diesem Grund greift hier das Release Management ein (Hughes 2017). Die Bündelung mehrerer Changes zu einem Release hilft dem Unternehmen, aus mehreren Eingriffen auf das Produktivsystem einen zu machen.

Weiteres ergibt es aus der Projektperspektive oft keinen Sinn, Änderungen, die konkret eine bestimmte Komponente des Produktivsystems betreffen, einzeln zu betrachten und mehrmals getrennt voneinander deployen. Wie bereits erwähnt, gilt das Produktivsystem als ein sensibler Bestandteil des laufenden Betriebes eines Unternehmens und mit jedem Deployment besteht die Gefahr, dieses zu stören. Aus diesem Grund verbindet man Changes zu einem Paket, um auch die Frequenz der Deployment zu minimieren, da diese so bereits sehr hoch sein kann (Lienemann 2006).

2.1.2 **Configuration Management**

Eine weitere sehr wichtige Komponente im Release Management ist das Configuration Management. Softwareprodukte wachsen mit der Zeit immer mehr und die Schnittstellenanzahl wird immer höher. Damit Änderungen am laufenden System bereits bei der Planung analysiert werden können und Auswirkungen auf das System erforscht werden können, unterstützt hier das Configuration Management. Dieses verfolgt und kontrolliert den gesamten Lebenszyklus der Software mit all ihren Schnittstellen. Vor allem die hohe Komplexität vieler Softwareprodukte benötigt ein Configuration Management, da die Zusammenhänge der Komponenten innerhalb der Systemlandschaft selbst Experten erschwert, den Überblick zu bewahren. Aus diesem

Grund unterstützen vor allem Visualisierung die zuständigen Personen, um das Wachstum und die Weiterentwicklung einer Software mitzuverfolgen (Saleem and Burney 2019).

In einer sogenannten Configuration Management-Datenbank (CMDB) werden notwendige Informationen zum Produktivsystem gespeichert. Sie dient vor allem im Release Management dazu, Auswirkungen der Änderungen einer Komponente abschätzen zu können. Es ist daher absolut notwendig, dass der Stand in der Datenbank den Ist-Zustand des laufenden Systems darstellt. Zu jedem Programm muss daher eine Dokumentation aller Schnittstellen vorliegen. Durch eine entsprechende Datenaufbereitung in der CMDB kann im Verlauf des Release-Prozesses die Frage nach den Auswirkungen eines Deployment auf weitere Komponenten schnell und genau beantwortet werden (Elsässer 2005).

2.2 Release Management in der IT

In den letzten Jahren ist vor allem ein Softwaremodel in den Fokus gerückt: Software-as-a-Service (SaaS). Es handelt sich hierbei um eine kontinuierliche Weiterentwicklung einer Software, die für den Endbenutzer meist über einen Webservice verfügbar ist (Barqawi et al. 2016). Vor allem in diesem Modell ist die Verfügbarkeit des Service eines der Hauptkriterien für eine hohe Kundenzufriedenheit. Ein Ausfall des Systems wäre fatal und kann in bestimmten Bereichen des Unternehmens zu einem hohen finanziellen Verlust führen.

Umso wichtiger ist es deshalb durch ein Release Management das Risiko eines Fehlers oder eines Ausfalls des Systems trotz notwendiger Eingriffe so minimal wie möglich zu halten. Mit dieser Entwicklung haben sich wichtige Methoden in den Vordergrund gespielt, die als Unterstützung dienen, all jene Ziele zu erreichen, für die das Release Management eingesetzt wird. Konkret handelt es sich hier um die agile Softwareentwicklung sowie das „Continuous-Implementation“- Modell von DevOps (Sikender 2019a). Ausschlaggebend für diese Entwicklung ist vor allem die Qualitätserwartung an Softwares.

2.2.1 **Ansätze und Methoden im Release Management**

Die Anforderungen an Softwares steigen durch den Trend der Digitalisierung stark. Neben sicherheitstechnischen Aspekten müssen Unternehmen die ständig wechselnde Softwareumgebung im Auge behalten, ohne die Performance oder Stabilität der Software zu gefährden. Um mit diesen Gefahren umzugehen, werden bereits in vielen Unternehmen Methoden angewendet, die für die Sicherstellung der Qualität einer Software dienen.

Amazon hat die neuen Anforderungen an Software früh erkannt und bereits eine der neuesten Methoden, Continuous Integration und Continuous Delivery (CI/CD), in ihren Prozess implementiert (Stacy, D., Prudnikov, M., Khan, A., & Shen, X. 2017). Mit dieser Änderung waren sie Vorreiter einer Methode, die heute in vielen IT-Unternehmen verwendet wird oder in der Implementierungsphase ist. Die Vorteile davon werden in dem detaillierteren Abschnitt über CI/CD erklärt.

In der Literatur gibt es viele Ansätze, wie ein Release Management und auch der Software-Lebenszyklus beschrieben werden kann (Samer 2016). Je nachdem welcher dieser Ansätze verfolgt wird, unterscheiden sich die einzelnen Phasen und Schritte im Durchlauf eines Releases. Eine grobe Übersicht verschiedener Ansätze wird in folgendem Absatz bereitgestellt.

2.2.1.1. Ad-hoc Methode

Der Ad-hoc Ansatz besteht aus hauptsächlich manueller Arbeit und einem sehr geringen Automatisierungsgrad. In vielen Unternehmen kommt diese Methode noch zum Einsatz. Der Produktmanager entscheidet in diesem Fall selbst über die Ausführung des Releases. Dadurch entstehen jedoch subjektive Projektziele und ist daher auch nur für relativ kleine Projekte oder interne Projekte durchsetzbar.

2.2.1.2. Inkrementeller Ansatz

In diesem Prozess wird die Software von Beginn an in mehrere Releases geteilt. Neue Funktionalitäten sowie Fehlerbehebungen werden in den sequenziellen Releases eingespielt und der Kunde kann bereits sehr früh in der Entwicklungsphase an dem weiteren Verlauf teilhaben und gegebenenfalls eingreifen. Außerdem kommt dieser früher in Kontakt mit dem Produkt und kann die Software bereits in den Unternehmensprozess einbinden. Um für Kunden und Entwicklerseite diesen Ansatz erfolgreich zu verfolgen, ist es zu Beginn des Projektstartes notwendig zu definieren, welche Funktionalitäten und Änderungen in welchem Release übergeben werden. Dies ist der große Nachteil des inkrementellen Ansatzes. Da die Kundenziele und die Projektziele oft in einem Konflikt zueinanderstehen, entstehen hier in der Planung oft Unstimmigkeiten zwischen den Parteien (Khan 2020).

2.2.1.3. Agile Softwareentwicklung

Vor allem in der IT-Branche ist eine Methode mittlerweile stark verbreitet und nahezu jedem ein Begriff. Die Rede ist hier vom Agilen Manifest. Angefangen mit den 12 Prinzipien aus dem Agilen Manifest entwickelten sich diese Richtlinien zu einem festen Bestandteil zur modernen Softwareentwicklung und stellten einen Grundstein für viele neue Methoden (Beck 2001). Der Kunde spielt in diesem Ansatz die wichtigste Rolle. Kontinuierliche Abstimmungen mit diesem sollen einerseits dem Entwicklungsteam die Anforderungen stets bewusst machen, um so wenig Interpretationsspielraum wie möglich zu bieten. Andererseits werden durch die regelmäßigen Zusammentreffen dem Kunden stets Einblicke in den aktuellen Stand des Projektes gewährleistet, damit auch dieser von den neuesten Entwicklung Bescheid weiß. Gegebenenfalls kann man im Falle einer Missinterpretation oder einer fehlenden Anforderung den Fehler frühzeitig gemeinsam mit dem Kunden beheben.

Ähnlich zum inkrementellen Ansatz wird das Gesamtprojekt in mehrere Releases aufgeteilt. Diese werden „User Stories“ genannt. Auch hier bedarf es einen sehr hohen Kommunikationsaufwand, vor allem mit dem Kunden. In den fixen Terminen werden den

Kunden im Laufe des Projektes stets eine funktionierende Software mit den Neuerungen präsentiert. Durch die Beteiligung des Kunden weist diese Methode nachweislich eine höhere Kundenzufriedenheit auf und durch das mögliche Eingreifen der Kunden oder auch des Projektteams ist auch der finanzielle Aufwand geringer als eine spätere Änderung im Projekt umzusetzen.

2.2.1.4. DevOps

Eng in Verbindung mit der agilen Softwareentwicklung steht der Begriff DevOps. Schnelle und vor allem flexible Entwicklung steht im Vordergrund, sowie einen fluiden Prozess zu erstellen, der die Entwicklung, das Ausrollen und jegliche operative Hintergrundtätigkeiten beinhaltet (Ebert 2016). Diese operativen Aufgaben sind üblicherweise abteilungsübergreifend miteinander verknüpft, was in der Praxis zu Problemen führen kann, welche durch den DevOps-Ansatz vermieden werden sollen.

Wie bereits beschrieben, können die Ziele der agilen Softwareentwicklung gut mit den des DevOps verbunden werden. Während im Agilen Manifest die 12 Prinzipien als Leitfaden gelten, stehen in DevOps bereits Technologien im Fokus, die den Prozess unterstützen. Die Zusammenarbeit der operativen Abteilung und dem Entwicklungsteam soll durch die Unterstützung diverser Tools automatisiert werden und damit den Projektprozess beschleunigen, sowie den Austausch mit den Stakeholdern erleichtern. Der bereits in der frühen Entwicklungsphase automatisierte Austausch zwischen dem Entwicklungsteam und den operativen Bereichen soll das Projektzielverständnis sowie die Release-Planung fördern. In den letzten Jahren wurde hier klar dahin gearbeitet, den kompletten Softwarelebenszyklus zu automatisieren, sodass auch mehrere Releases pro Tag problemlos möglich sind (Dhaya 2021).

2.2.1.5. Continuous Delivery

Basierend auf die Agile Entwicklungsmethodik arbeitet die Methode der Continuous Delivery mit ähnlichen Prinzipien. In regelmäßigen und kurzen Abständen werden dem Kunden Teile der Software übergeben. Zusätzlich besteht dieser Ansatz aus zusätzlichen Methodiken, die gezielt die Nachteile des Agilen Ansatzes, wie die Kommunikation und der Prozess, ausgleichen. Dadurch sind hochqualitative Releases auch in viel kürzeren Abständen möglich. Laufende Produktübergaben erlauben es, die richtige Software mit einer erhöhten Kundenzufriedenheit zielorientiert zu entwickeln und erfolgreich auf den Markt zu bringen.

Dieser Ansatz verfolgt unter anderem ein schnelles Deployment einer Änderung. Dies soll jedoch nicht heißen, dass diese Änderungen Prozessteile, die in den anderen Ansätzen vorhanden sind, überspringen. Es ist hier der Fall, dass bereits ein hoher Grad an Automatisierung es ermöglicht, einen schnelleren Prozess durchzuführen. In weiterer Folge wird versucht, die Änderungen klein zu halten, sodass viele Deployments mit kleinen Änderungen zum Schluss das gesamte Feature darstellen. Durch die minimale Größenordnung der Änderung ist zusätzlich das Risiko eines Fehlers im Deployment geringer (Miikka 2018).

2.2.2 **Automatisierung im Release Management**

Wie bereits erwähnt, spielt auch die Automatisierung im Release Management eine sehr große Rolle. Ziel dieser ist es, jegliche Tasks im Prozess zu standardisieren und in weiterer Folge zu automatisieren, die regelmäßig in gleicher Form im Prozess ausgeführt werden müssen. Oft geht es hier um Interaktionen zwischen Menschen und den Kommunikationsaustausch sowie in der Entwicklungsperspektive das Testen und ein Deployment. Als Resultat entsteht am Ende ein einheitlicher Prozess, der immer das gleiche Ergebnis liefert und somit auch eine gewisse Qualität garantiert. Wie bereits erwähnt, benötigt es vor allem im Release Management einen hohen Grad an Kommunikation, oft auch zwischen mehreren Bereichen. Mit der Größe der Projekte wächst auch die Komplexität dieser Interaktionen und ohne technische Hilfsmittel sind diese nicht mehr zu meistern (Senepathi et al. 2018).

Heute werden in diesen Bereichen des Prozesses Tools eingesetzt, die die betroffenen Personen unterstützen. Vor allem in Hinblick darauf, dass die Qualität des Produktes immer im Vordergrund steht, versuchen die Projektteams so effizient wie möglich das Projekt durchzuführen. Aus diesem Grund forcieren die Tools Teile des Prozesses zu automatisieren. Das Hauptaugenmerk für Automatisierung liegt vor allem auf manuelle Tasks sowie Übergaben. Ein weiterer großer Vorteil aus Sicht der Organisation ist durch das Unterlassen der menschlichen Interaktion, dass Änderungen am Personal nicht ausschlaggebend für den Prozess und vor allem nicht für die Qualität des Releases sind.

Das Thema Versionierung ist ein weiterer großer Punkt in der Automatisierung und spielt in der Entwicklung und im Deployment eine wichtige Rolle. Versionierung ist in der Praxis schwierig handzuhaben, da es vielerlei Definitionen zur richtigen Versionierung gibt und oft subjektiv entschieden wird, ob das neue Softwareprodukt eine neue Version ist. Weiteres wird bei der Wartung von mehreren Versionen einer Software der Aufwand immer höher und hindert das Entwicklungsteam an der Weiterentwicklung eines neuen Produktes. Durch die Verwendung einer automatisierten Versionierung ist es möglich, schnell und einfach Releases zu deployen, sowie bei unerwarteten Fehlern auf einen bekannten Stand der Software zurückzugehen. Tools für die Versionierung ermöglichen das Testen aller Komponenten des Releases bereits vor der Ausrollung der Änderungen. Zusätzlich benötigen auch Schnittstellen ausreichende Informationen, da vor allem diese von Anpassungen im Produkt Bescheid wissen sollen, da eine geänderte Schnittstelle zu einem Bruch am gesamten System führen und dadurch große Auswirkungen im Betrieb haben kann.

2.3 Phasen im Release Management

Das Release Management kann also als ein Prozess gesehen werden. Angefangen mit der Aufnahme der Anforderungen von einem Kunden über die Planung der Software und dem Release bis zum Deployment der Änderungen. Wie in Abbildung 1 zu sehen, kann dieser Prozess in drei große Phasen geteilt werden. In den Release Goals werden die Projektanforderungen erhoben. Hier spielt das Requierement Engineering eine große Rolle, mit

dem die Kundenwünsche bereits vor dem Start so genau wie möglich dokumentiert werden. Bereits in dieser Phase des Prozesses wird das Ziel des Releases definiert.

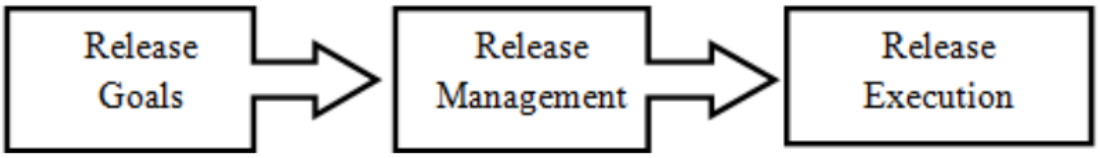

Abbildung 1: Überblick Release Management Prozess (Amir et al. 2011)

In weiterer Folge starten die Umsetzung und Entwicklung des Releases. Ein hoher Grad an Standardisierung und Automatisierung soll hier eingesetzt werden, um auch bereits während der Entwicklung eine hohe Qualität sicherzustellen. Schlussendlich werden in der Release Execution, die im Release Management umgesetzten Änderungen auf das Produktivsystem ausgerollt.

Dies stellt einen vereinfachten Prozess im Release Management dar. Ein solcher wird im nächsten Abschnitt detaillierter mit den beteiligten Rollen und einzelnen Phasen beschrieben. Dazu muss erwähnt werden, dass je genauer der Prozess im Release Management abgebildet wird, desto subjektiver wird dieser. Ab einem bestimmten Detaillierungsgrad kann der beschriebene Prozess nicht mehr in jedem Unternehmen umgesetzt werden und spiegelt demzufolge nicht die Realität wider.

Release Management ist im Detail ein sehr komplexer Prozess mit vielen Tasks, die von Personen aus verschiedenen Bereichen erledigt werden müssen. Demzufolge ist es wichtig vor der Definierung eines Prozesses die Rollen zu bestimmen. Nicht außer Acht gelassen werden darf jedoch die Rolle des Kunden im Release Management. Wie bereits in den verschiedenen Methoden und Ansätzen des Release Managements beschrieben, ist der Kunde am Prozess beteiligt. Um den Komplexitätsgrad der Beschreibung nicht zu überlasten, ergibt es aus diesem Grund Sinn, das Release Management mit ihren Rollen in die zwei Perspektiven zu teilen: die Akzeptanz- beziehungsweise Kunden- und die Verkäuferseite (Kajko-Mattsson M. 2005).

In der Arbeit von (Kajko-Mattsson M. 2005) wird ein Prozess definiert, der basierend auf die vordefinierten Rollen ausführbar ist. Je nachdem, welche Rollen tatsächlich in der Praxis vorhanden sind, können Prozesse anders aussehen. Hier wurde also ein Modell erstellt, das auf definierte Rollen zugeschnitten wurde und auf eine Recherche des Modells von Microsoft und Oracle baut. In Abbildung 2 ist der in sieben Phasen aufgeteilte Prozess visualisiert dargestellt und wiederum in die Kunden- und Verkäuferperspektiven getrennt worden.

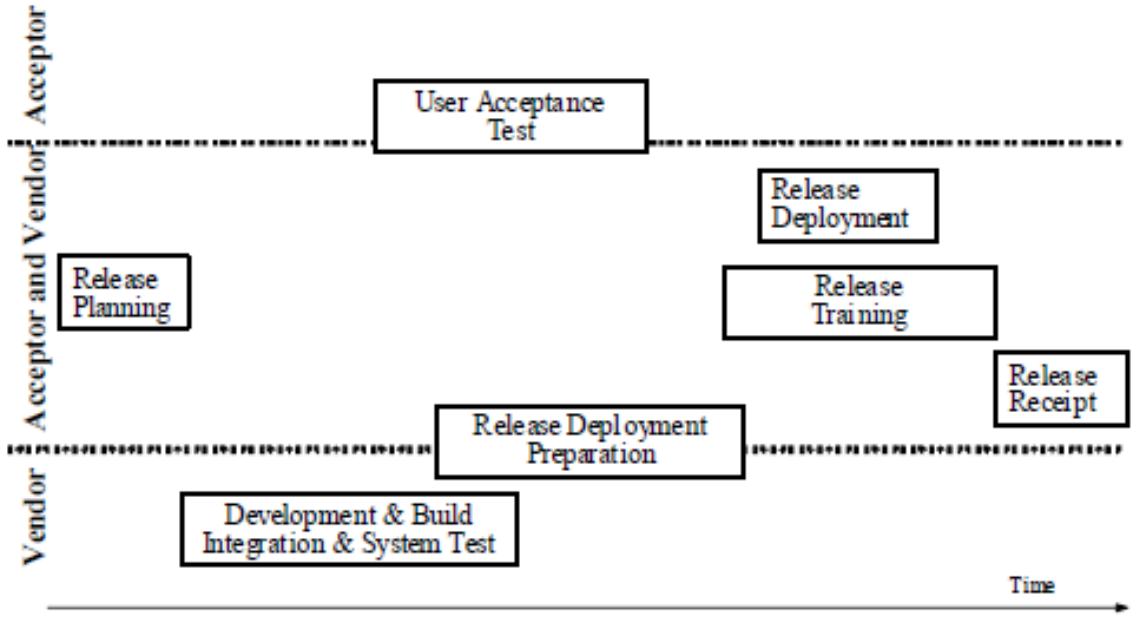

Abbildung 2: Release Management Prozess mit Perspektiventrennung aus (Kajko-Mattsson M. 2005)

2.3.1 **Release-Planung**

Wie bereits in der Einführung erwähnt, spielt die Planung des Releases eine entscheidende Rolle im Erfolg der Software. Demzufolge beginnt der Release-Prozess mit einer bedeutenden Aufgabe, die wie in der Abbildung 2 zu sehen ist, die Verkäufer- und Kundenseite betrifft und in starker Zusammenarbeit ausgearbeitet werden muss. Diese Phase wird vom Release Project Leader geführt, welcher für die Kunden den Change Request analysiert und eine Ressourcenschätzung sowie Budgetvorkalkulierung durchführt. Basierend auf ein erstelltes Dokument, in dem ein Angebot für den geplanten Release aufbereitet wird, folgen weitere detailliertere Planungen, weswegen hier bereits erkennbar ist, dass der Angebotsentwurf im weiteren Verlauf ausschlaggebend für die Qualität ist.

Im nächsten Schritt wird dem Release Advisory Board der Release-Plan vorgelegt, welcher auf den Inhalt und Ressourcen geprüft wird und im Optimalfall genehmigt wird. Im Anschluss stellen der Release Manager und der Release Project Leader auf der entsprechenden Seite ein Team zusammen, dass an diesem Release arbeitet. Aus diesem Schritt ist erkennbar, dass die Rollen in einem Release nicht an fixe Personen gebunden sind. Die Teams in den Releases können daher immer unterschiedlich besetzt werden.

Bereits in der Release Planung kommt es in den Unternehmen schon zu vielen Unterschieden. Dies liegt vor allem daran, dass die Planung hauptsächlich noch ad hoc geschieht und noch keine Modelle und einheitliche Methoden verwendet werden. Die Planung wird also nach wie vor trotz einer hohen Anzahl an Features von den Personen selbst durchgeführt, was zu erheblichen Unterschieden in den Unternehmen führt (Ruhe and Saliu 2005).

2.3.2 **Development & Build Integration & System Test (DBIST)**

Diese Phase besteht hauptsächlich aus Tasks für das Entwicklungsteam. Nachdem die Features entwickelt wurden, erstellt der Build Integrator eine neue Version der Software und ladet diese in das vorhandene System. Dieser Schritt muss detailliert beschrieben werden, damit klar ersichtlich ist, welche Features die Tests passen und welche nicht und warum. Diese System- und Akzeptanztests werden im Anschluss mit den Ergebnissen dem Release Project Leader vorgestellt, damit der Fortschritt des Releases überprüft und der Prozess laut Plan weitergeführt werden kann. Nachdem der neue Build in das System eingebaut ist, startet das System Test Team mit den Funktionstest der neuen Features. Die Erstellung eines Testplans beinhaltet auch das Überprüfen weiterer Faktoren wie Sicherheit und Performance.

2.3.3 **User Acceptance Test**

Auf der Kundenseite bereiten das Operation Team und die Endnutzer einen Testplan vor, um die angeforderten Funktionalitäten zu testen. Als Vorbereitung erstellt das Konfigurationsteam eine Testumgebung, die dem Produktivsystem gleicht, damit die Tests so aussagekräftig wie möglich sind. Im Testplan sind Akzeptanzkriterien definiert, welche konkret getestet werden und erfüllt sein müssen. Die Ergebnisse werden wiederum dem Release Manager berichtet, woraufhin über den weiteren Verlauf entschieden wird.

Entscheidend ist hier das Ergebnis der Akzeptanztests, welches „Akzeptiert“, „Abgelehnt“ oder „Mit Vorbehalt akzeptiert“ sein kann. Im Optimalfall kann mit dem Status „Akzeptiert“ in die nächste Phase übergegangen werden. In den anderen Fällen wird zurück in die DBIST-Phase gesprungen und die Änderungen an den Features werden überarbeitet.

Bei der Übermittlung der Ergebnisse von den Akzeptanztests wird mit dem Release Project Leader entschieden, wie der Prozess fortschreiten soll. Wie bereits erwähnt, wird bei der Akzeptierung der Tests mit der nächsten Phase, der Deployment-Vorbereitung begonnen. Wenn Teile des Build nicht akzeptiert wurden, werden diese dem Entwicklungsteam zurückgegeben, die eine detaillierte Beschreibung vorgelegen bekommen, welche Anforderungen nicht erfüllt wurden und bauen diese Anforderungen wiederum in den neuen Build ein. Wenn der gesamte Build im Akzeptanztest durchgefallen ist, erfordert es meist sogar den Sprung zurück in die Release Planungsphase, um erneut zu erläutern, was die Systemanforderungen sind.

2.3.4 **Deployment-Vorbereitung**

Nachdem der neue Build von der Kundenseite akzeptiert wurde, startet die Vorbereitung für das Deployment. Dafür bereiten der Release Project Leader mit den Entwicklern auf der Verkäuferseite einen Backout-Plan vor, im Falle, dass beim Deployment ein Fehler auftritt. Dieser muss wiederum von dem System Test Team getestet werden. Im nächsten Schritt wird der Release Documentation Engineer beauftragt, das Handbuch für die Kunden zu erstellen. In

dieser Anleitung sollen die neuen Funktionen, Benutzerhandbuch, eine Auflistung der Änderungen, eine Installationsanleitung, sowie alle weiteren Medien beinhalten sein.

Falls ein Release mehrere Kunden hat, muss dieses Handbuch für jeden Kunden geschrieben werden. Wenn die sogenannten Release Notes finalisiert sind, wird das Release Package zusammengestellt. Dort sind alle Releasenotes sowie die neue Software und gegebenenfalls automatisierte Installationsskripte eingebunden. Das Release Package wird im Anschluss an den Release Manager übergeben, welcher dieses überprüft und entscheidet, ob die Produktivumgebung auf der Kundenseite auch bereit für das Deployment ist. Wenn diese Entscheidung positiv ausfällt, wird das Release Package dem Release Advisory Board vorgelegt, welcher das Deployment genehmigt.

2.3.5 **Release Deployment**

Wenn der Release auch vom Release Advisory Board genehmigt wurde, startet das System Administration Team mit der Installation des neuen Releases in der Produktivumgebung. Wenn bei der Installation Probleme auftreten, wird der vorbereitet Backout-Plan zur Hand genommen und umgesetzt, während alle beteiligten Rollen über den Stand des Prozesses informiert werden (Rasa G., Dr. RSD Wahida Banu 2019).

2.4 Probleme im Release Management

In den vorigen Kapiteln wurde viel über das Release Management und diverse Definitionen geschrieben. Trotz der klaren Definitionen, die aus den Forschungen entstanden sind, haben Unternehmen heute noch immer große Probleme in der tatsächlichen Ausführung eines Releases. Vor allem das Thema rund um die Automatisierung ist für viele Unternehmen noch nicht umsetzbar und der Grad an manueller Arbeit in einem Release ist noch hoch.

In folgendem Abschnitt werden verschiedene Probleme beschrieben, die Unternehmen bei der Umsetzung eines Releases haben.

2.4.1 **Probleme im Projektfortschritt**

In Softwareprojekten spielen sogenannte „Major Features“ eine große Rolle. Diese dienen als eine Art Meilensteine und müssen erstellt sein, um im Projektfortschritt weiterarbeiten zu können (Michlmayr et al. 2007). Probleme bei der Fertigstellung eines der Major Features führt daher zu einem Stillstand aller weiteren Tasks.

Ähnlich verhält es sich bei Abhängigkeiten. In vielen Softwares gibt es Abhängigkeiten zu weiteren Projekten. Auch dort kann es vorkommen, dass diese Komponente noch nicht die Version hat, die für ein Deployment notwendig ist. Aus diesem Grund kann es hier zu weiteren Wartezeiten kommen, da das Entwicklungsteam selbst keinen Einfluss auf die Entwicklung eines anderen Projektes hat.

2.4.2 **Formulierung der Release-Anforderungen**

Wie bereits in den Phasen beschrieben, hängt der Erfolg eines Releases stark von der Release-Planung ab. Hier gilt es, die Anforderungen der Kunden so gut wie möglich zu formulieren. Dies stellt bekanntlich eine der größten Herausforderungen der Softwareentwicklung dar. Mit dem Agilen Ansatz versucht das Projektmanagement dagegenzuwirken, indem das Projekt in kleineren Teilen geliefert wird und Unstimmigkeiten frühzeitig bemerkbar werden. Abbildung 3 zeigt ein in der Softwareentwicklung bekanntes Bild, das sinnbildlich für die Problematik in der Formulierung von Anforderungen steht und die Notwendigkeit von Agilen Arbeiten aufzeigt.

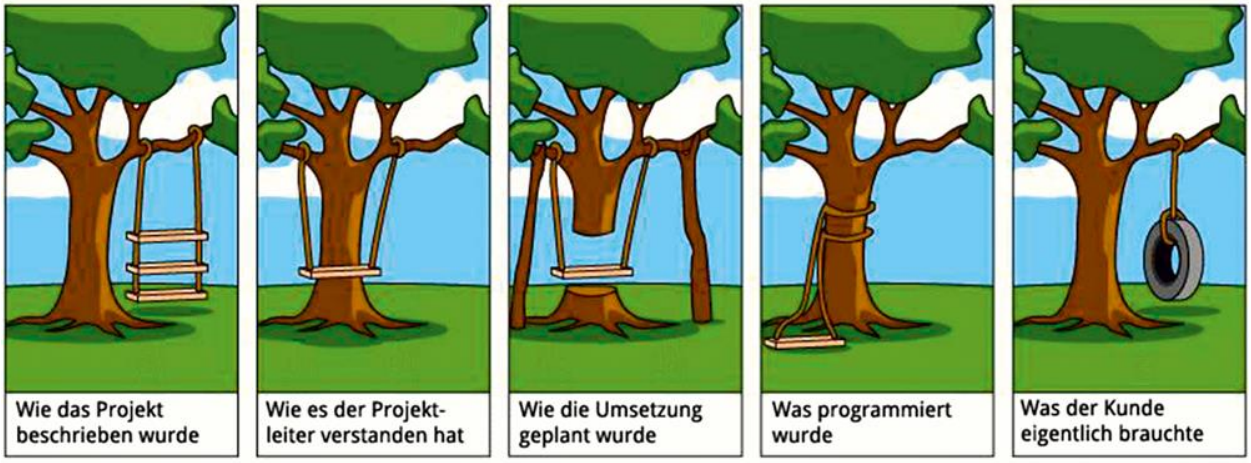

Abbildung 3: Beispiel Softwarebeschreibung - SCRUM

Nichtsdestotrotz bleibt die Formulierung kleiner Projektteile eine Herausforderung, die es zu meistern gibt. Vor allem in der Planungsphase, wo es zu viel Interaktion zwischen Kunden und Verkäufer kommt, besteht die Gefahr, dass Anforderungen nicht ausreichend beschrieben werden.

2.4.3 **Testen der Software**

Dieser Punkt muss in zwei Bereiche geteilt werden, da es einerseits zu Problemen kommen kann, die auf das Testen an sich zurückzuführen sind und Problemen, da die Softwareprodukte einen sehr hohen Grad an Komplexität aufweisen, sodass einzelne Features nicht getrennt voneinander getestet werden können.

Das Testen an sich nimmt auf der Kunden- sowie auf der Verkäuferseite eine große Rolle ein. Während auf der Entwicklerseite Integrations- sowie Unittests zur Sicherstellung der Qualität dienen, werden auf der Kundenseite Akzeptanztests vorbereitet, die erfüllt werden müssen. Bekannterweise wird das Testen der Software auf beiden Seiten nicht die notwendige Priorität zugeteilt, was bei der Ausrollung des Releases dann im laufenden Betrieb zu Nacharbeiten führt. Auch die nicht ausreichende Formulierung der Akzeptanzkriterien ist Teil des Problems.

2.4.4 **Infrastruktur**

Auf den Servern eines Unternehmens laufen mehrere Applikationen. Für die Überwachung und Wartung dieser Server gibt es meistens ein eigenes Team, das getrennt von Projektentwicklungen diese Aufgaben durchführt. Nichtsdestotrotz muss das Infrastrukturteam zu Zeitpunkten in einem Projekt in Abstimmung mit dem Entwicklerteam Aufgaben erledigen, die für den Release notwendig sind.

Die Infrastrukturserver können jedoch jederzeit Probleme bereiten, sodass die Priorität der Aufgaben des Infrastrukturteams geändert wird. In solchen Fällen kommt es für das Projektteam zu einem Stillstand und einer Abhängigkeit zu dem anderen Team. Wenn es im Projektteam keine treibende Kraft gibt, die in Abstimmung mit den Administratoren die nächsten Schritte priorisiert, kann der Projektstillstand mehrere Tage dauern.

2.4.5 **Automatisierung**

Trotz der hohen Anzahl an Möglichkeiten, wie Automatisierung in einem Release eingebunden werden kann, ist der Grad an manueller Arbeit in vielen Unternehmen dennoch sehr hoch. Die Gründe dafür sind vielseitig (Wikström 2019).

Einerseits gibt es vor allem bei Kunden, wo die Software für finanzkritische Themen eingesetzt werden, ein sehr hohes Risiko. Denn es kann ein hoher finanzieller Schaden bei einem neuen Release entstehen, wenn beim Deployment ein Fehler auftritt. Aus diesem Grund ist in Verträgen teilweise noch hinterlegt, dass ein Roll-out auf das Produktivsystem nicht ohne die Zustimmung des Kunden durchgeführt werden darf. Dies hat zur Folge, dass eine Implementierung eines Continuous Deployment nicht durchzuführen ist.

In weiterer Folge treten im Bereich der Versionierung Schiefstände auf, da weitere Kunden auf eine neue Version bestehen. Somit wird der Wartungsaufwand verschiedener Versionen einer Software höher und zeitintensiver. Die Gefahr, dass bei weiteren Deployments Probleme auftreten, steigt dadurch enorm (Chen 2017).

In vielen Unternehmen wird eine Software bereits mit einem konfigurierbaren Konzept erstellt. Aus diesem Grund können verschiedene Kunden die gleiche Software benutzen, jedoch auf spezielle Anwendungsfälle konfigurieren. In der Einführung eines automatisierten Prozesses kann diese Eigenschaft jedoch zu Schwierigkeiten führen. Entscheidungen über die Konfiguration können nicht immer bereits im Vorfeld geklärt oder abgeschätzt werden und abgeschätzt werden. Man weißt davor nicht, welche Auswirkungen diese auf die produktive Umgebung haben kann (Wikström 2019).

2.4.6 **Prozessänderung**

Ein weiteres Problem, das in vielen Unternehmen auftritt und auch in der Literatur nicht vollends geklärt ist, behandelt das Thema rund um die Prozessadaptierung im Zuge der Steigerung der Effektivität im Release Management. Wie bereits in vorigen Kapiteln erwähnt, etablierte sich in

der Softwareentwicklung der DevOps Ansatz. Hierbei handelt es sich jedoch nicht um eine Methode, die im Unternehmen eingeführt werden kann. Es beinhaltet eine Umstrukturierung des Teams und des gesamten Entwicklungsprozesses (Joby 2019).

Dieser Umbau führt in vielen Unternehmen jedoch zu Problemen. Einerseits sind hierfür notwendige Skills bei dem Aufbau notwendig, die nicht standardmäßig im Personal aufgrund der Spezifität des Themas vorhanden sind. Andererseits muss der Prozess in die Unternehmensstruktur passen. Bis das neu geformte Team diesen Prozess kennen gelernt hat und vor allem die Verantwortlichkeiten in den Bereichen der Entwicklung definiert sind, kommt es zu Zeitverzögerungen bei der Entwicklung selbst. Auch zu erwähnen ist, dass das Produkt, das geliefert wird, in der Prozessänderung involviert ist. Sei es eine architektonische Änderung in der Software oder nur die Regelmäßigkeit in den Updates. Die Software muss auch auf diesem Prozess abgestimmt sein (Leite et al. 2020).

2.5 Tools im Release Management

Als Unterstützung zur Erhöhung der Qualität eines Releases gibt es Tools, die in vielen Unternehmen eingesetzt werden. Mithilfe dieser Tools werden Teile des Prozesses digitalisiert und sollen somit die Effektivität im Prozess erhöhen. So werden teils einfache Benachrichtigungen oder sogar ein komplettes Deployment in diesen Tools automatisiert. Nach wie vor gibt es jedoch auch dort Probleme, die Tools richtig anzuwenden, sodass der Prozess davon profitiert und nicht darunter leidet. Im folgenden Abschnitt werden zwei Tools näher vorgestellt, welche von bekannten Anbietern entwickelt worden sind. Einerseits Azure DevOps von Microsoft und Produkte aus der Atlassian Suite. Beide zählen in der IT-Branche zu den Big-Playern und unterstützen das Release Team bei der Durchführung eines Releases.

2.5.1 **Microsoft Azure DevOps**

Bei Azure DevOps handelt es sich um eine Plattform, die die Kollaboration innerhalb eines Softwareproduktes unterstützen soll. Der Azure DevOps Server ist vormals unter Team Foundation Server (TFMS) bekannt gewesen und mit der Version ab 2019 in Azure DevOps Server umbenannt worden. Der Begriff DevOps wurde bereits in vorherigen Kapiteln nähergebracht und steht für die Kombination aus Entwicklung (Development) und operativen Tätigkeiten (Operation). Microsoft bietet mit DevOps viele Features wie Versionskontrolle und Benutzerverwaltungen an, die vor allem im Release Management verwendet werden können.

Azure DevOps verwendet das Feature „Release Pipeline“ in vielen ihrer Produkte, die über Lizenzen im Unternehmen eingebunden werden können. Mit diesem Tool soll vor allem das regelmäßige Ausrollen der Software erleichtert werden und zu einem großen Teil automatisiert werden. Wie in (Umme et al. 2020) beschrieben, hat bereits ein halbautomatisierter Prozess einen Vorteil gegenüber des rein manuellen Prozesses.

In Abbildung 4 ist der Release-Prozess in Azure DevOps erläutert und wie ein solcher im laufenden Betrieb aussehen kann. Die Hauptrolle in diesem Prozess spielt ein sogenannter

„Agent“, der in DevOps die Tasks in einem Deployment ausführen kann. Release Pipelines können im Vorfeld konfiguriert werden und entsprechende Freigaben verwenden, sodass Unternehmensrichtlinien eingehalten werden können und die entsprechende Sicherheit gewährleistet ist.

Beginnend mit der Anfrage eines Deployment, die vom User gestartet wird, überprüft die Pipeline, ob für das Deployment eine Genehmigung erforderlich ist. Dies wird aus der erwähnten Konfiguration gelesen. Im nächsten Schritt wird das Deployment in eine Queue geschrieben, in der ein Agent zugreift und den Task abarbeitet. Da in DevOps Agenten konfiguriert werden können, greift nicht der nächste freie Agent zu und holt den Task, sondern es wird ein entsprechend konfigurierter als Abnehmer des Tasks verwendet.

Im nächsten Schritt lädt der Agent alle Artefakte, die in der Software benötigt werden in seinen Speicher. Ein Artefakt steht in diesem Kontext für ein Paket aus Dateien, die als externe Referenz dienen und Abhängigkeiten darstellen und für den Build des Programms notwendig sind.

Darauffolgend startet der Agent alle Schritte, die im Deployment-Skript hinterlegt sind und lädt alle erstellten Artefakte auf das definierte System. Während des gesamten Prozesses werden alle Schritte mitdokumentiert und in Log-Dateien in DevOps geladen. Zu guter Letzt kann eine Post-Deployment-Bestätigung eingestellt werden, woraufhin eine zuständige Person das Deployment auf die nächste Stage bestätigen kann (Roopesh 2021).

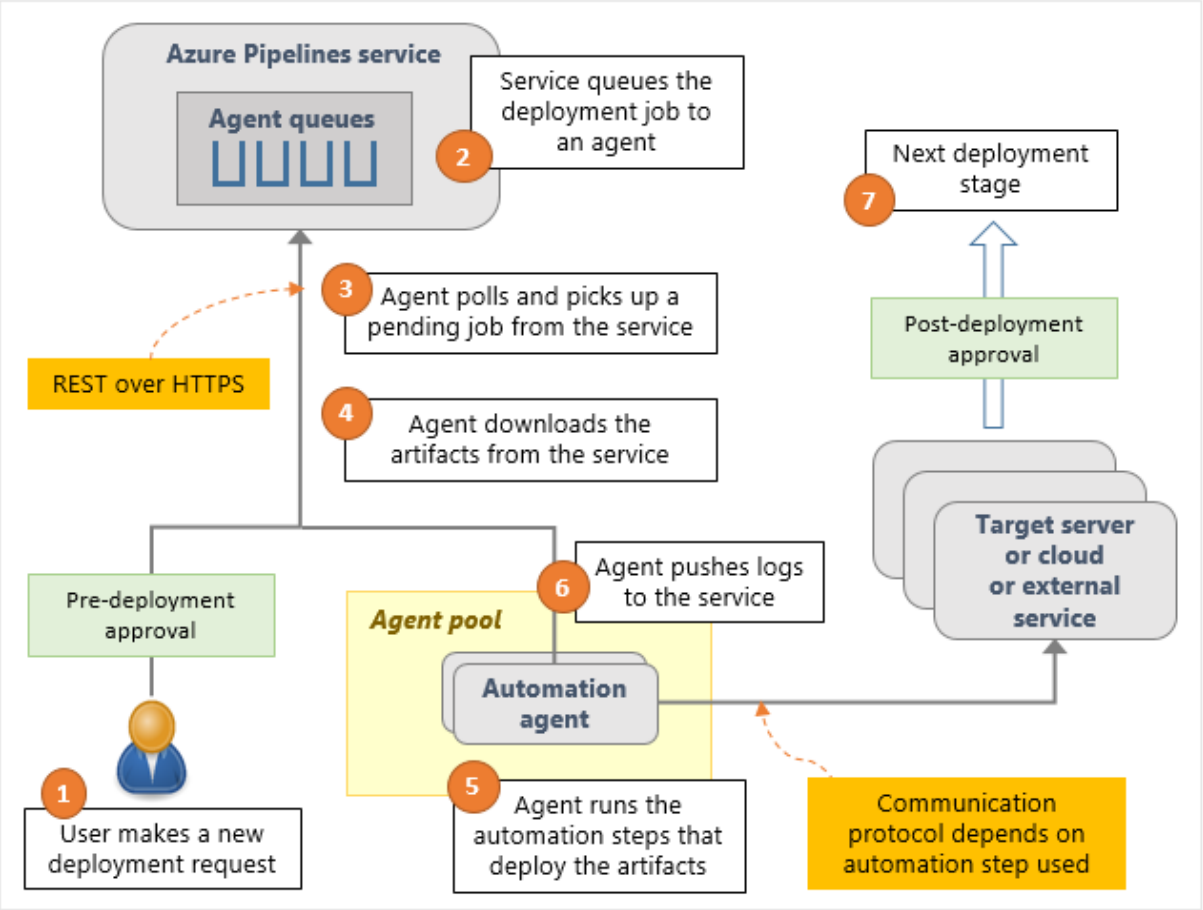

Abbildung 4: Azure DevOps Pipeline (RoopeshNair 2021)

2.5.2 **Atlassian Suite**

Das Australische Softwareunternehmen Atlassian entwickelte in den letzten Jahren Software-Tools, die Unternehmen im operativen Geschäft unterstützen sollen. Im Bereich des Release Managements spielen drei Produkte eine sehr große Rolle, die unterschiedliche Aufgaben in den Prozessen unterstützen und differenziert aber auch in Verbindung zusammen eingesetzt werden können. Vor allem bei der gemeinsamen Nutzung der drei Tools können Vorteile in vielerlei Hinsicht entstehen, die die Qualität des Release Managements erheblich steigern.

2.5.2.1. Confluence

Als erstes Produkt der Atlassian Suite wird in folgenden Absatz Confluence nähergebracht. Confluence stellt einen Bereich zu Verfügung, in dem mehrere Benutzer gleichzeitig Seiten bearbeiten können. Vor allem in Hinblick auf Dokumentation und Wissensansammlungen eignet sich dieses Tool, da es einen zentralen Speicherort für vielerlei Anwendungsfälle darstellt. Die Möglichkeit Ideen auszuarbeiten, Teams und Aufgaben zu strukturieren und zu teilen bietet Teams jeglicher Art Vorteile, die die tägliche Arbeit unterstützen und erleichtern.

Das Anpassen und Erstellen von Vorlagen, die für Confluence Seiten verwendet werden, gehört zu den mächtigsten Features, die diese Software zu bieten hat. Aus diesem Grund können verschiedene Teamstrukturen und Unternehmensrichtlinien in Dokumentationen eingebunden werden, ohne dass der Anpassungsaufwand zu groß wird. Ein weiteres wichtige Feature, dass die Benutzung von Confluence unterstützt, ist die Einbindung weiterer Softwareprodukte von Atlassian die in den nächsten Abschnitten beschrieben werden (Atlassian 2021b).

2.5.2.2. Jira

Wie bereits in früheren Abschnitten beschrieben, rückte in den letzten Jahren in der IT das agile Arbeitsmanagement stark in den Vordergrund. Mit Jira veröffentlichte Atlassian 2002 eine Webanwendung, die Teams in diesem Bereich nutzen. Features wie Taskboards und integrierte Workflows unterstützen vor allem das Anforderungsmanagement, sowie den Helpdesk in ihren Prozessen.

Während Azure DevOps vor allem Softwareentwicklern in der DBIST-Phase eine große Unterstützung darstellt, ist Jira hauptsächlich für organisatorische Tasks eine Hilfe. Durch die Möglichkeit als Unternehmen selbst Release-Pläne zu erstellen und diese mit vorgefertigten Dokumenten-Templates zu ergänzen, bietet Jira viele Optionen, das Tool in das interne Release Management einzubinden. Diese Templates führen dazu, dass in den manuellen Tasks so wenig Information wie möglich vergessen wird (Atlassian 2021c).

2.5.2.3. Bitbucket

Als Unterstützung für die Entwickler kann in Jira Bitbucket als Add-On integriert werden. Dabei handelt es sich um ein Code-Versionstool, das eine ähnliche Funktionalität zu Azure DevOps bietet. Auch hier können sogenannte Pipelines erstellt werden, die vor allem im Bereich Continuous Integration und Continuous Deployment (CI/CD) eine wichtige Rolle spielen und eine Möglichkeit zur Automatisierung von Tests und Deployment darstellen (Atlassian 2021a).

3 PRAXISERFORSCHUNG

Nach der theoretischen Aufarbeitung und der Einleitung in das sehr breit aufgestellte Thema rund um das Release Management folgt nun die praktischere Einführung. Der Release-Prozess kann in jedem Unternehmen unterschiedlich aussehen und durchgeführt werden. Gründe hierfür liegen in verschiedenen Bereichen. Während einerseits der Kunde in vielen Teilprozessen im Release mitarbeitet und dadurch den Prozess mitbestimmen kann, besitzt nicht jedes Unternehmen die gleichen Möglichkeiten im personellen, zeitlichen sowie im technischen Bereich.

Aufgrund dieser Ursachen wird im folgenden Teil mithilfe einer Definition der Zielunternehmensgruppen eine Basis für die Praxiserforschung gelegt. In weiterer Folge werden die Tools Azure Devops und Jira noch detaillierter in ihren Anwendungsmöglichkeiten analysiert, somit basierend auf deren Ergebnissen die Hypothesen formuliert werden können und der Interviewleitfaden erstellt werden kann.

3.1 Definition mittelständisches IT-Unternehmen

Die Forschungsfrage *„Wie setzen mittelständische IT-Unternehmen Jira und Azure DevOps ein, um die Qualität im Release Management zu steigern?“* bezieht sich auf mittelständische IT-Unternehmen. Aus diesem Grund werden in diesem Abschnitt Analysegrenzen definiert, damit in der Forschungsmethodik keine falschen Informationen in die Auswertungen einfließen.

	Mitarbeiter	**Umsatz**	**Bilanzsumme**	**Eigenständigkeit**
Kleinstunternehmen	Bis 9	≤ 2 Mio. Euro	≤ 2 Mio. Euro	i.A. Kapitalanteile oder Stimmrechte im Fremdbesitz ≤ 25 Prozent (*)
Kleinunternehmen	Bis 49	≤10 Mio. Euro	≤10 Mio. Euro	
Mittlere Unternehmen	Bis 249	≤ 50 Mio. Euro	≤ 43 Mio. Euro	
Großunternehmen	Ab 250	> 50 Mio. Euro	> 43 Mio. Euro	

Tabelle 1: Klassifikation Unternehmensgrößen (WKO 2022)

Zu Beginn werden die Unternehmensgrößen nach der offiziellen Auslegung der Wirtschaftskammer definiert. In dieser Definition werden Unternehmen nach den in Tabelle zu sehenden Kriterien unterteilt (WKO 2022). Mit der Definition Mittlere Unternehmen sind die Unternehmensgrenzen bei 50 bis 249 Mitarbeitern. Um jedoch in dieser Arbeit die relevanten Informationen zu bekommen, wird für diese Arbeit die Grenze von 50 bis 1000 Arbeitern festgelegt, da bis zu dieser Unternehmensgröße die Release Prozesse verglichen werden können und das Produkt selbst mehr für die Unterschiede im Prozess verantwortlich ist.

Bei Azure DevOps und Jira handelt es sich um zwei Big-Player im Bereich der Planung und Entwicklung von Softwareprodukten, weswegen die Kosten für diese Tools eine sehr große Rolle spielen. Um den Rahmen dieser Arbeit jedoch nicht zu sprengen, wird dieser Punkt, welches Tool geeigneter für jemanden ist, nicht berücksichtigt und auch nicht behandelt.

3.2 Verwendung der Tools im Unternehmen

In dieser Arbeit wird der Fokus auf zwei der bekanntesten Tools gelegt. Es handelt sich hier um die bereits im Kapitel 2.5 beschriebenen Tools Jira und Azure Devops. Viele Unternehmen sind bereits in den letzten Jahren vor der Wahl gestanden, welches Tool am besten für den bereits vorhandenen Prozess geeignet ist. Bei dieser Entscheidung spielen Faktoren wie bereits vorhandenen Nutzungswissen, sowie vor allem aus der Unternehmenssicht die relevanten Lizenzgebühren und viele weitere Argumente eine große Rolle.

Zum heutigen Zeitpunkt, wo Unternehmen bereits auf mehrere Jahre der Verwendung des Tools zurückblicken können, gibt es jedoch im Release Management noch immer Probleme, die im Prozess auftreten und auch eine fehlerfreie Release-Durchführung erschweren. Die Hersteller Atlassian für Jira sowie Microsoft für Azure DevOps bieten den Nutzern ein ausführliches Handbuch an, mithilfe denen die Verwendung im eigenen Unternehmen unterstützt werden können.

In weiterer Folge wurden auch bereits Bücher veröffentlicht, die einen Leitfaden für die Integration von den Tools bilden sollen. Als Beispiel dient hier das Buch *Agile Project Management with Azure DevOps* (Rossberg 2019). Als großes Problem stellt sich jedoch immer wieder heraus, ein neues Tool in einem bereits implementierten Prozess einzubinden. Auch hierfür bieten die Entwickler von den Atlassian und Microsoft Möglichkeiten für Anpassungen zur Verwendung der Tools. Nichtsdestotrotz gibt es dort Grenzen der Software, die eine Anpassung des eigenen Prozesses notwendig machen oder auch einen effektiven Einsatz des Tools nicht möglich machen. Im folgenden Abschnitt werden nun die Möglichkeiten zur Verwendung der Tools bezogen auf die in Kapitel 2 beschriebenen Anforderungen sowie den Phasen des Release Managements beschrieben.

Aufgrund der bereits bekannten Thematik rund um die Probleme im Release Management gibt es bereits Arbeiten, die diesen Bereich analysieren und erforschen. Sei es die Nutzung von Azure DevOps oder die Verwendung von der Atlassian Suite oder nur Teile davon, Vorteile bringen diese in den meisten Fällen und wie solch eine Nutzung aussieht, wird in den folgenden Abschnitten erläutert.

3.2.1 **Atlassian Suite**

Atlassian bietet in ihren Produkten den Kunden die Möglichkeit, Schnittstellen selbst zu konfigurieren. Über eine Plugin SDK und eine REST API lassen sich die drei Hauptkomponenten Confluence, Jira und Bitbucket konfigurieren und anpassen. Während das Plugin als Add-on in alle drei Tools über den Plugin-Manager eingebunden werden können,

benötigt es für die REST API Skripte, die jedoch im Vergleich zu den Plugins weit flexibler sind und eine breitere Funktionspalette bieten (Johansson 2017). Basierend auf Interviews hat Johansson 2017 Änderungen an den Schnittstellenkomponenten durchgeführt, die dazu führten, dass manuelle Arbeit, die regelmäßig durchgeführt werden muss, automatisiert wird.

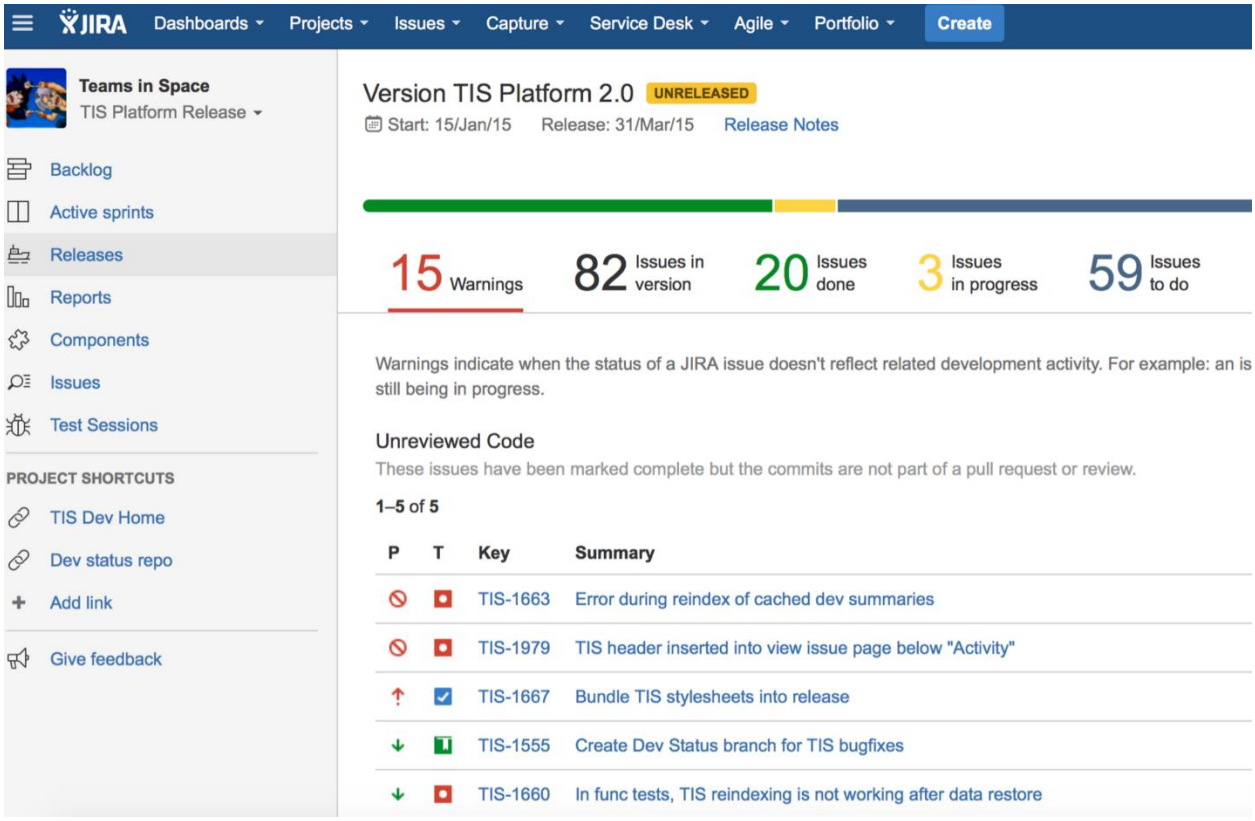

Abbildung 5: Jira Release (Groß 2021)

Vor allem für das Release Management bietet Atlassian eine standardisierte Oberfläche, die Informationen zu dem aktuellen Release visuell darstellt. Wie in Abbildung 5 zu sehen ist, befindet sich der Release derzeit in Entwicklung, was in dem Tag „*UNRELEASED*“ neben dem Header zu sehen ist. Derzeit gibt es 82 Issues, zu denen Tasks aufbereitet müssen, um diese zu lösen. Nachdem alle 82 Issues den Status „Done“ haben, kann die neue Version veröffentlicht werden. Dank implementierter Benachrichtigungsfunktionen werden auch automatisch betroffene Personen, die am Release beteiligt sind, über wichtige Änderungen und Informationen benachrichtigt und sind somit immer am aktuellen Stand. Vor allem für Release Manager ist diese Oberfläche hilfreich, da dort der aktuelle Status in der Entwicklung ersichtbar ist (Daly 2021).

3.2.2 **Azure DevOps**

Neben der Atlassian Suite gehört auch aus dem Microsoft-Portfolio Azure DevOps zu den bekanntesten Tools. Hierbei handelt es sich um ein Produkt, das vor allem die in der Softwareentwicklung verwendete Methode der Continuous Integration und Continuous Delivery (CI/CD) unterstützt. Im Vergleich zur Atlassian Suite liegt der Hauptaufgabenbereich hier auf die

Softwareentwicklung. Azure DevOps bietet neben der Codeverwaltung auch Taskboards für die Planung Agiler Projekte an und deckt dadurch den gesamten Softwarelebenszyklus ab.

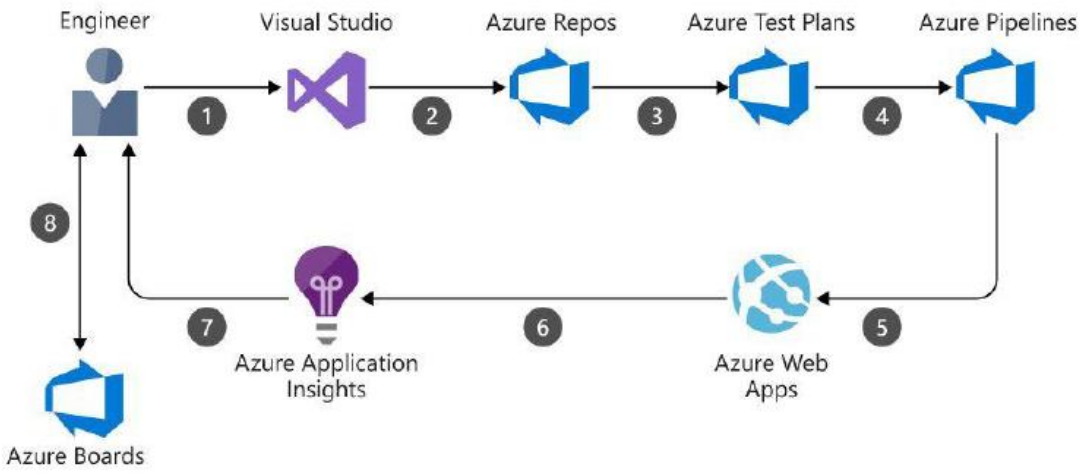

Abbildung 6: CI/CD in Azure DevOps für eine .NET Applikation (Dileepkumar and Mathew 2021)

Vor allem für Unternehmen, die in einer Microsoft-Umgebung arbeiten und entwickeln, bietet Azure DevOps große Vorteile im Release Management. Wie in Abbildung 6 zu sehen, gibt es viele Schritte im CI/CD-Prozess, die über die Azure DevOps-Funktionen abgedeckt werden können. Die von der Plattform vorgeschlagenen Release Pipelines, die bereits in früheren Kapitel vorgestellt wurden, bieten dem Entwickler hilfreiche Schritte zur Einbindung von Automatisierungsschritten in der Durchführung von Releases (Dileepkumar and Mathew 2021).

3.3 Verwendung der Tools im Prozess

Die erwähnten Tools Azure DevOps und Jira unterstützen im Prozess das Team, um effizienter an den Release zu arbeiten. Hierbei spielt es in diesem Abschnitt keine Rolle, welches Tool im Unternehmen genutzt wird. Es werden Möglichkeiten beider Tools dargestellt, ohne das eine Wertung abgegeben wird, welches Tool besser geeignet ist. Der Fokus in diesem Teil der Arbeit liegt auf den Bereichen, in denen die genannten Tools unterstützen können. Von dem Start des Releases in der Release Planung bis zum Release Deployment am Ende gibt es Bereiche, wo diese ihre Stärken besitzen und dadurch dem Unternehmen einen Mehrwert bringen.

3.3.1 **Planung**

Beginnend mit der Release Planung können mithilfe der Boards in Azure DevOps und Jira Tasks aufbereitet werden. Die Agilen Methoden wie SCRUM oder Kanban nutzen diese Boards zur Veranschaulichung der Tasks, die im Release für ein Feature zu erledigen sind. Zusätzlich lassen sich diese Boards in den Tools anpassen, sodass auch unternehmensspezifische Workflows abgebildet werden können.

In weiterer Folge können mit diesem Feature auch langfristige Planungen erstellt werden. Vor allem bei der Nutzung von SCRUM, wo die Planung über Sprints stattfindet, können Projektpläne mithilfe der Boards aufbereitet werden und auch Verantwortlichkeiten definiert werden. Abbildung 7 zeigt ein Board aus Azure DevOps, das eine visuelle Darstellung der Tasks zeigt, an die das Team derzeit arbeitet.

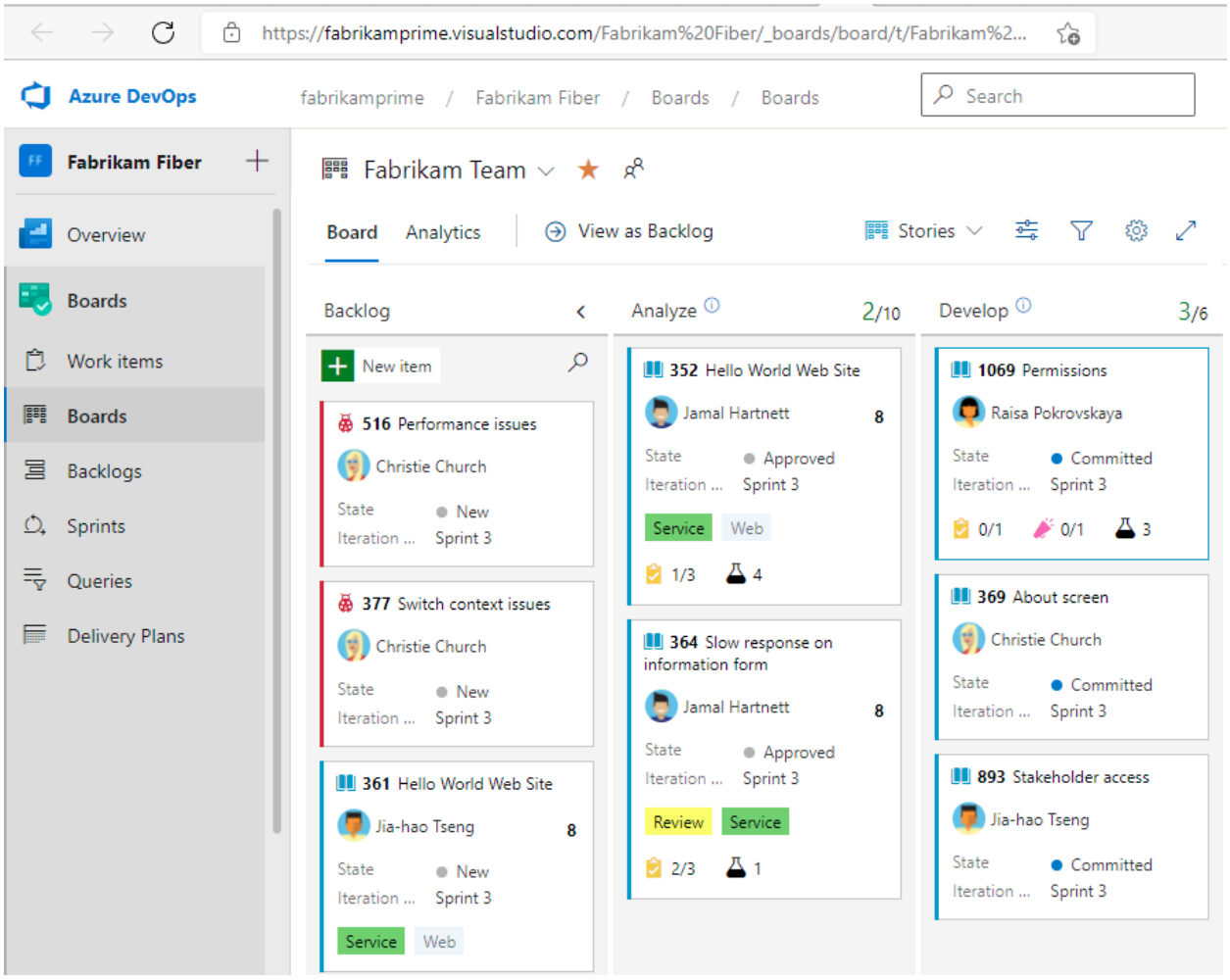

Abbildung 7: Azure DevOps Board (Microsoft 2022)

Auch Jira bietet als eines der Kernfeatures Boards, wie diese in Abbildung 7 zu sehen sind, an. Neben der Planung unterstützen die Boards vor allem in Meetings, um rasch einen Überblick zu bekommen, in welchem Status das Projekt sich gerade befindet und auch was die nächsten Schritte sind. Boards können auf verschiedenen Ebenen betrachtet werden. In Abbildung 8 ist das Jira Board zu sehen, in der die Taskebene betrachtet wird und die verschiedenen Status gezeigt werden. Anhand dieser Boards kann die Planung unterstützt werden und Überlastungen sowie freie Kapazitäten identifiziert und behandelt werden (Atlassian 2022).

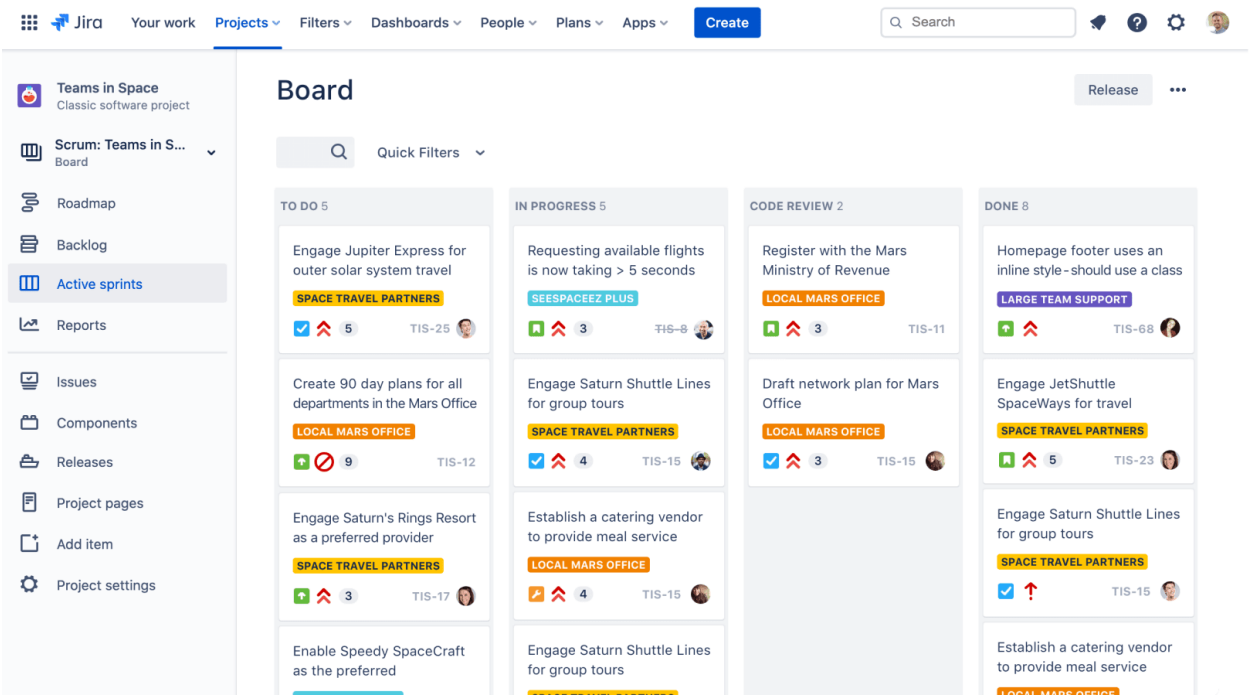

Abbildung 8: Jira Board (Atlassian 2022)

3.3.2 **Tracking**

In Verbindung mit der Planung steht das Tracking der Tasks. In den Tools wird oft über *Issue Tracking* gesprochen. Dieser Begriff steht für das Nachverfolgung aller Tasks, Features und User Stories, die in der agilen Softwareentwicklung Kategorien darstellen. Diese Kategorien werden in weiterer Folge als ein Work Item deklariert. Dieser Begriff kommt aus dem Azure DevOps-Syntax und fasst die erwähnten Kategorien zusammen. In den Tools besteht die Möglichkeit Verknüpfungen zwischen diesen Objekten herzustellen. Ein Test kann einem Task, einer User Story oder einem Feature zugewiesen werden. Tasks können User Stories als Parent haben und diese wiederum Features als Parent.

Mit diesen Verknüpfungen, die in Abbildung 9 am rechten Rand zu sehen sind, sind Informationen aus dem Projekt auf einem Blick zu erkennen. Dadurch wird einerseits die Planung unterstützt, indem die noch offenen Tasks zu in einem Projekt veranschaulicht werden. Andererseits verfolgt das Board alle Tasks in einem Projekt mit und es ist auf einem Blick ist erkennbar, wie der aktuelle Stand im Projekt steht. Dies dient vor allem der internen Transparenz gegenüber der Projektleitung oder auch Geschäftsführung. Andererseits wird dadurch die Transparenz gegenüber dem Kunden gewährt, da vor allem für sie die Information relevant ist, wie der aktuelle Status des Projektes aussieht.

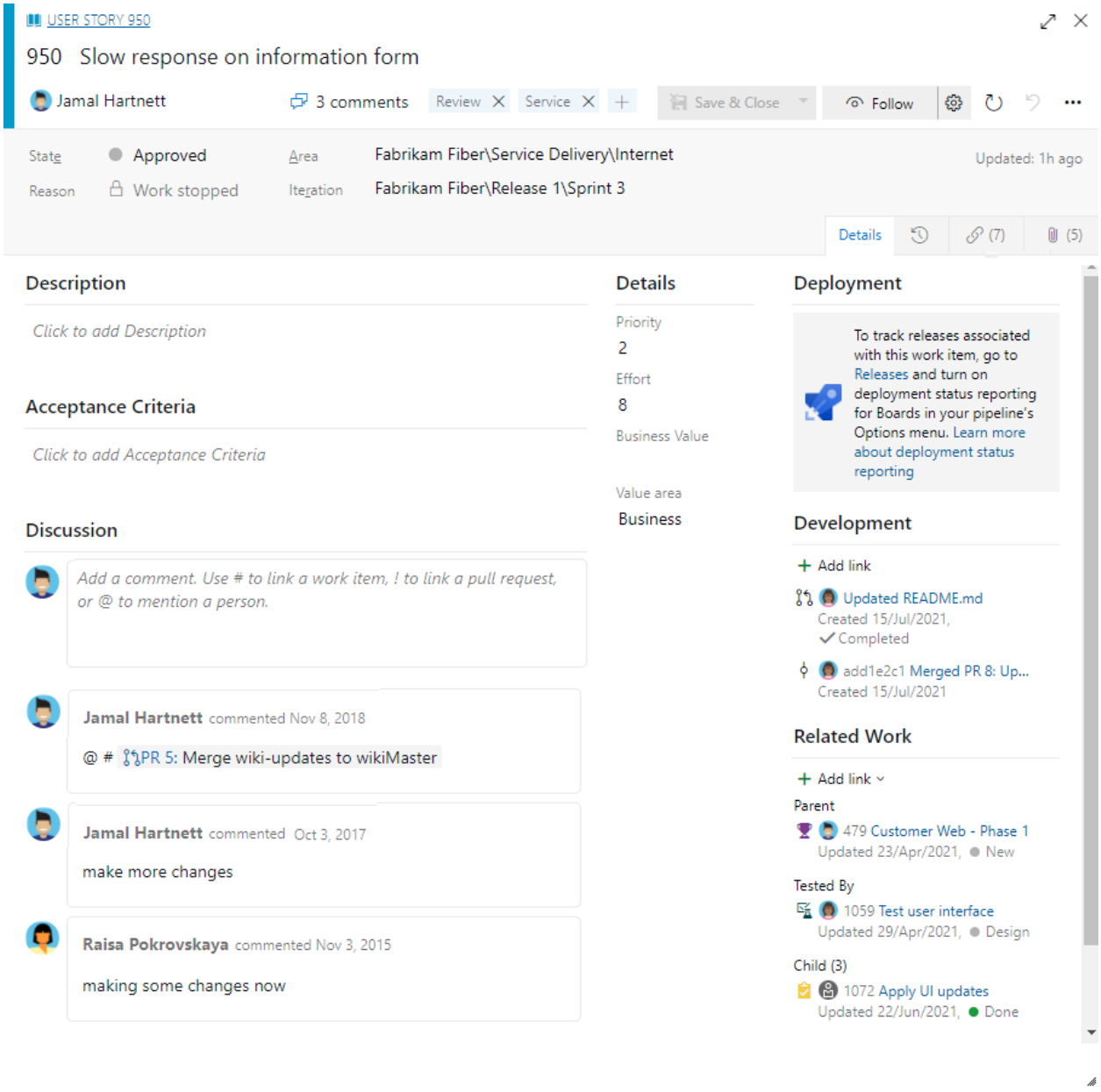

Abbildung 9: Azure DevOps Task Übersicht (Microsoft 2022)

Zusätzlich sind in den Tools Auswertungen integriert, die es ermöglichen, den Projektverlauf zu verfolgen und dadurch einen Trend zu identifizieren. Sogenannte Meilensteine in Projekten können dadurch Aufschluss geben, ob das Lieferdatum erreichbar ist oder ob Maßnahmen gesetzt werden müssen. Abbildung 10 zeigt eine mögliche Auswertung dieser Zusammenfassung eines Sprints. Diese können vom Benutzer angepasst werden, sodass diese die für das Team notwendige Informationen darstellt (Microsoft 2022).

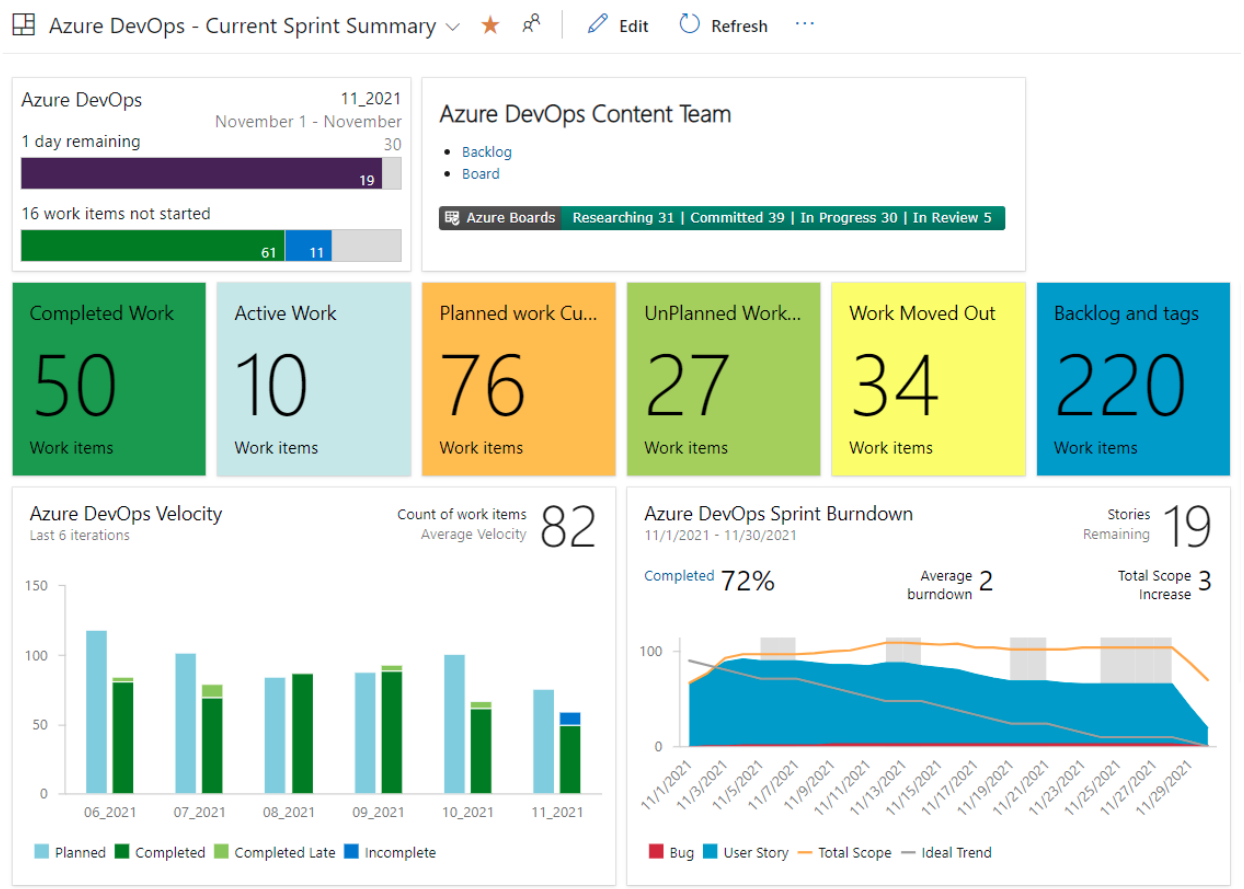

Abbildung 10: Azure DevOps Dashboard (Microsoft 2022)

In Abbildung 9 ist links neben dem Titel der User Story die Zahl 950 zu sehen. Diese dient im Tool als eindeutige Identifikationsnummer. Mithilfe dieser ID ist es möglich im ganzen System auf diese User Story zu referenzieren. Einerseits in der verfügbaren Kommentarfunktion, die es bei jedem Objekt im Tool gibt. Andererseits können hiermit Änderungen im Softwarecode mit dieser User Story verlinkt werden, in dem diese in einem Commit referenziert werden. Näheres dazu wird im Abschnitt der Softwareverwaltung erklärt.

3.3.3 Testen der Software

Ein essenzielles Thema im Release Management ist das Testen der Software. Unabhängig davon, um welche Testkategorien es sich handelt, sollen diese dokumentiert sein und schnell verfügbar sein. Auch bei diesem Punkt stellt das Tool Funktionalitäten zur Verfügung, die das Testen im Release abdecken.

Während in Azure DevOps die Testpläne eine breite Funktionspalette für Tests zu Verfügung stellen, sind für Jira verschiedene Add-Ons integrierbar, die abhängig von der Notwendigkeit im Unternehmen angebunden werden können. Einerseits spielt das automatisierte Testen in der Softwareentwicklung bereits eine große Rolle. Andererseits sind nach wie vor manuelle Tests notwendig. Diese finden größten Teils auf der Kundenseite statt und müssen dementsprechend dokumentiert sein. Im Release Management spricht man hier von den User Acceptance Tests. Tools wie Azure DevOps und XRay, das ein Jira Add-On darstellt, bieten den IT-Unternehmen manuelle Tests in die projektspezifische Teststrategie einzubinden. Dies erfolgt durch ein

wiederum adaptierbares Framework in den Tools, das die Kundentests und interne Überprüfungen unterstützt, eine detaillierte Testdokumentation zu erstellen.

Neben den manuellen Testungen spielen die bereits erwähnten automatisierten Tests eine große Rolle in der Softwareentwicklung. Der hohe Grad der Komplexität der Softwareprodukte erhöht die Notwenigkeit der Tests enorm. Dass automatisierte Tests hierfür benötigt werden, um in der Lage zu sein, Software in einer am Markt üblichen Dauer zu entwickeln, wurde bereits in früheren Kapiteln erläutert. Auch in diesem Bereich bieten die Tools bereits integrierte Supportfeatures an. Während in Jira wiederum XRay auch für die automatisierten Tests verwendet werden kann, wird in Azure DevOps die Pipeline-Funktionalität eingesetzt, die auch im Bereich des CI/CD eine große Rolle spielt.

3.3.4 **CI/CD**

Neben dem Testen wird im Zuge der Automatisierung in der Softwareentwicklung zusätzlich oft über das automatisierte Deployment gesprochen. In einem nach ITIL-definierten Release Management verfolgt das Entwicklerteam den Ansatz, Änderungen auf dem Entwicklungssystem durchzuführen. Diese werden über die Testumgebung und die Stagingumgebung am Ende auf das Produktivsystem deployed. Bereits hier ist erkennbar, dass es sich um einen wiederholenden Prozess handelt. Aufgrund der Tatsache, dass die Änderungen durch die Umgebungen durchgeschliffen werden und getestet werden, unterstützt die bereits erwähnte Pipeline in Azure DevOps diesen Prozess zu automatisieren. Über Konfigurationen ist es dort möglich, Artefakte von einem System auf das nächste System zu deployen.

Aus dem Atlassian-Portfolio kommt hier Bitbucket ins Spiel. Bitbucket bietet den Softwareentwicklern eine ähnliche Funktionspalette an. Hier gelten die Bitbucket Pipelines als Unterstützung für ein automatisiertes Deployment. Mithilfe dieser Pipelines können Deployment automatisiert durchgeführt werden. Basierend auf die Konfigurationen können verschiedene Zielplattformen bestimmt werden, wohin der Code deployed werden soll. In diesen Pipelines können allerlei Tasks erstellt werden, die bei einem Deployment auszuführen sind. Beispielsweise kann hier ein unternehmensspezifischer Release Prozess konfiguriert werden, sodass erst auf die Produktivumgebung ausgerollt werden darf, wenn alle Issues geschlossen worden sind.

Mithilfe dieser Konfigurationen können in den Pipelines sehr flexibel Prozesse hinterlegt werden, die an bestimmten Bedingungen gebunden sind. Wie zum Beispiel eine Pipeline in Azure DevOps konfiguriert ist, wurde bereits in Kapitel 2.5 gezeigt.

3.3.5 **Source Code Verwaltung**

Zu guter Letzt ist vor allem für die Softwareentwicklung ein System notwendig, in dem der Code gespeichert wird. GIT, ein System für Versionskontrolle, unterstützt Teams, zusammen an

einem Projekt zu arbeiten. Ein Repository bietet den Speicherort dieser Projekte an. Azure DevOps und Bitbucket bieten mit ihrer GIT-Integration eine breite Palette an Funktionaltäten, um die Kollaboration in einem Softwareentwicklungsteam zu fördern. Mithilfe der GIT-Integration ist es möglich, in der Entwicklung von Projekten, den Release Prozess zu verfolgen und Projektinformationen aus Jira oder Azure DevOps zu verknüpfen.

Im Zusammenspiel mit den Pipelines kann der Automatisierungsgrad erhöht werden. Hierfür bieten die Tools konfigurierbare Trigger, die bestimmen, wann ein Release veröffentlicht werden soll. Als Beispiel kann eine Änderung in einem Softwareteil ein automatisiertes Deployment auslösen. Azure DevOps und Bitbucket bieten unter anderem eine visuelle Oberfläche, die es das Team erlaubt, einfach Code Reviews durchzuführen. Des Weiteren ermöglichen Features wie eine semantische Suche im Code Informationen über Softwareteile zu erlangen, was vor allem bei einer Fehlersuche hilfreich sein kann.

4 METHODIK

Im folgenden Abschnitt werden die in der Arbeit bereits erforschten Informationen aus der Literatur überprüft und mittels wissenschaftlicher Forschung validiert. Die beschriebenen Möglichkeiten der Nutzung von Azure Devops und Jira ermöglichen in der Privatwirtschaft einen vielfältigen Einsatz, der auf die unterschiedlichen Unternehmen anpassbar ist. Aus diesem Grund wird im nächsten Punkt die Methodik beschrieben, sowie begründet, warum diese für die Arbeit gewählt wurde.

4.1 Wahl der Methodik

Für den Forschungsteil wird zwischen einer qualitativen und der quantitativen Methode entschieden. Während bei der quantitativen Methode die Anzahl der Ergebnisse für die Auswertung relevant ist, stehen in der qualitativen Methode die einzelnen Ergebnisse selbst im Vordergrund. Ziel dieser Arbeit ist es, basierend auf den Ergebnissen des Forschungsteil ein Framework zu erstellen, dass für Unternehmen im IT-Sektor eingesetzt werden kann, um das Release Management im Unternehmen zu stärken und zu sichern.

Der Vorteil der qualitativen gegenüber der quantitativen Forschung liegt darin, dass mithilfe von Interviews verschiedene Meinungen eingeholt werden und auch tiefere Einblicke in verschiedene Ansätze erfragt werden können. Aus diesem Grund wird in dieser Arbeit die qualitative Forschungsmethode angewendet, um schließlich aus den Ergebnissen der Interviews die Thesen zu validieren.

4.1.1 **Bestimmung Stichprobe**

In der qualitativen Forschung spielt die Wahl der zu befragenden Experten eine entscheidende Rolle. Um durch mangelndes Wissen das Ergebnis nicht zu verfälschen, ist es notwendig die Gruppe an Personen unter Berücksichtigung folgender Kriterien zu wählen, sodass mit einer geringen Anzahl an Personen so allgemeine Informationen wie möglich gefunden werden kann.

Wie im Titel und in der Forschungsfrage herauszulesen ist, werden in dieser Arbeit die zwei Tools Azure Devops und Jira getrennt voneinander analysiert. Aus diesem Grund muss die Stichprobe auch die Verwendung beider Tools beinhalten. Weiteres ist es notwendig, verschiedene Rollen in Release Prozess zu interviewen. Wie bereits in der Prozessbeschreibung aus Kapitel 2.3 zu lesen ist, gibt es im Release Management verschiedene Aufgabenbereiche, die teilweise getrennt voneinander analysiert werden können und auch Schnittstellen zwischen den Personen und Aufgaben, die stark in Verbindung zueinanderstehen. Unter Berücksichtigung dieser Faktoren wird die Stichprobenauswahl in zwei

Kategorien geteilt. Einerseits geben Informationen von Personen aus dem operativen Sektor Informationen über Probleme im Prozess und in der Schnittstellen Kommunikation zu den Entwicklern und zu den Kunden. Beispielsweise werden hierfür die Rollen des Projektleiters oder aus dem Agilen Umfeld Product Owners oder auch Scrum Masters.

Demgegenüber steht eine Gruppe, die aus Entwicklern besteht. Aus den Antworten aus dieser Gruppe sollen Informationen gewonnen werden, die im Bereich des Testens und auch der Automatisierung relevant sind. Wie im Abschnitt über die Funktionalitäten der Tools zu sehen ist, bestehen Möglichkeiten, um diese Schritte in der Entwicklung einer Software bereits zu Teilen oder auch vollkommen zu automatisieren. Inwieweit diese Möglichkeit dort bereits genutzt wird und Probleme entstehen, sollen die Antworten aus diesen Interviews liefern.

Ein wichtiger Punkt in der Wahl der Stichprobe sind die Kenntnisse dieser Tools. Neben der in den Jahren gesammelten Erfahrung spielt auch die Häufigkeit der Verwendung eine große Rolle. Aus diesem Grund wird ausgeschlossen, dass die Interviewpartner nicht genügend Knowhow in der Nutzung der Tools mitbringen und keine tieferen Einblicke in Problemen bieten können. Des Weiteren ist in diesem Fall wichtig, nicht Personen aus dem gleichen Team zu wählen. Ziel dieser Masterarbeit ist es, ein Framework zu erstellen, das unternehmensunabhängig eingesetzt werden kann und durch eine geringe Varianz in den Prozessen der Experten besteht die Gefahr, eine zu unternehmensspezifische Lösung zu erforschen.

4.1.2 **Begründung Wahl der Methodik**

Wie bereits erwähnt, wird in dieser Arbeit der qualitative Ansatz der Forschungsmethodik gewählt. Aus den Antworten dieser Experteninterviews soll wie im Titel der Arbeit zu erkennen ein Framework entstehen, das für mittelständige IT-Unternehmen als Hilfestellung genutzt werden kann, um die Probleme im Release Management zu lösen. Da hierfür detailliertes Wissen und auch tiefere Einblicke in Probleme benötigt werden, ist der qualitative Ansatz die passende Lösung, da durch die Kontrolle des Interviewleitfadens nähere Details zum Lösungsfindungsprozess gewährt werden.

Vor allem in Hinblick auf die Erstellung des Frameworks und die Beantwortung der Hypothesen bringt diese Forschungsmethode wichtige Vorteile gegenüber der quantitativen Methode. Der angesprochene Interviewleitfaden, der zu den Antworten der Hypothesen führen soll, wird in Kapitel 4.2 noch detaillierter Beschrieben. Um jedoch zu diesem zu kommen, müssen die Ziele des Interviews definiert werden, welche im kommenden Absatz mit Bezug auf den Hypothesen zu lesen sind.

4.1.3 **Aufstellen der Hypothesen**

Für den Forschungsteil werden nun Hypothesen aufgestellt, die mithilfe der Experteninterviews beantwortet werden sollen. Zu Beginn müssen zu den Hypothesen die abhängigen und die unabhängigen Variablen definiert werden. In weiterer Folge werden auch die Hypothesen

kategorisiert. Zur Auswahl stehen hier die ungerichtete und die gerichtete Hypothese. Somit kann auch definiert werden, ob und welchen Einfluss die unabhängige Variable auf die abhängige Variable hat. In dieser Arbeit wird jedoch die gerichtete Hypothese forciert, um nicht nur den Effekt selbst zu überprüfen, sondern auch in welche Richtung sich die abhängige Variable beeinflussen lässt.

4.1.3.1. Hypothese A: Nutzungszeitraum

Für die erste Hypothese spielt die Erfahrung des Tools eine große Rolle. Bei der Einführung einer neuen Software gibt es vor allem zu Beginn noch viele Fragen, die die Benutzung des neuen Produktes betreffen. Erst im Laufe der Zeit und mit der Dauer der Nutzung klären sich diese Fragen und das Wissen über das Tool wirkt sich dann im Prozess des Release Managements aus, womit die Effizienz gesteigert werden kann. Ob diese Effizienz auch langfristig steigt und nicht nach einem Punkt der Erfahrung stagniert, wird in der ersten Hypothese H1 überprüft. Der Nutzungszeitraum stellt in dieser Annahme die unabhängige Variable dar. Wie der Einfluss dieser Variable auf die Dauer des Deployment ist, wird in der folgenden Hypothese erforscht:

„*Je höher der Nutzungszeitraum des Tools, desto kürzer die Dauer eines Deployment.“*

4.1.3.2. Hypothese B: Nutzungsdauer

In der nächsten Hypothese wird mehr Bezug auf die aktive Nutzung des Tools hergestellt. Neben der gesamten Nutzungsdauer von dem Zeitpunkt der Einführung des Programms bis hin zur Erstellung dieser Arbeit spielt auch das Ausmaß der Verwendung eine große Rolle. Während der Release Manager Jira oder Azure DevOps mehr als Prozessüberwachungstool verwendet, ist es für die Entwickler und Tester mehr ein Tasktrackingprogramm, in dem jegliche Tasks in einem Projekt dokumentiert und verfolgt werden können. Nichtsdestotrotz kann die Anzahl der Stunden einen Einfluss auf die Dauer eines Releases haben. Und dieser Einfluss wird in dieser Hypothese überprüft:

„Je mehr Stunden pro Woche das Tool genutzt wird, desto kürzer ist die Dauer eines Releases.“

4.1.3.3. Hypothese C: Prozess

In weiterer Folge wird der im Unternehmen angewendete Prozess detaillierter analysiert. Trotz der Verfügbarkeit von Tools wie Jira und Azure DevOps kommt es vor, dass Probleme im Release Management auftreten. In weiterer Folge versuchen die betroffenen Teams der Ursachen für diese Probleme auf den Grund zu gehen, jedoch stellt sich dieses Erfangen meist als weniger Erfolgreich dar. Teils sind Verbindungen und Abhängigkeiten zwischen Tasks in den Prozessen nicht erkennbar oder Probleme treten nicht in den Kontext des Tools auf. In diesem Fall kann jedoch auch das Hinzufügen oder Ändern eines Tools das Problem nicht

lösen. Der Ursprung des Problems liegt dann eher bei der Wahl der Arbeitsmethode oder auch des gesamten Prozesses. Aus diesem Grund lautet die nächste Hypothese folgendermaßen:

„Funktionierende Release-Prozesse sind toolunabhängig.“

4.1.3.4. Hypothese D: Rollen im Release Management

Eine wichtige Rolle spielt im Release Management die Aufgabenverteilung. In einem funktionierenden Prozess müssen jegliche Schritte klar definiert sein und auch einer verantwortlichen Person zugewiesen sein. Der Weg bis zu diesem funktionierenden Prozess ist jedoch nicht einfach und erfordert eine detaillierte Analyse der im Unternehmen vorhandenen Teamstruktur. Nur mithilfe des Ergebnisses kann dieser erstellt und eingeführt werden. Die Rolle des unterstützenden Tools wird in dieser Analyse jedoch nicht immer vorgesehen. Die Schnittstelle aller Rollen zu ignorieren, kann jedoch Folgen in der Effektivität mit sich bringen. Neben der Software selbst stellt sich auch die Frage, ob es eine zuständige Person geben soll, die die Funktionalität des Tools in den Prozess einbindet. Ob eine erweiterte Rollenverteilung wie diese auch einen Einfluss auf die Qualität des Release Managements hat wird in dieser Hypothese erforscht, die nachfolgend definiert ist:

„Eine erweiterte Rollenverteilung für die Tools verringert die Anzahl an Problemen im Release Prozess.“

4.1.3.5. Hypothese E: Handbücher

Als letzte Hypothese werden die Dokumentationen und Handbücher analysiert. Vor allem von den Anbietern gibt es Leitfäden und auch Bedienungsanleitungen für die Nutzung der Software, welche den Nutzern Funktionalität und den Einsatz des Tools näherbringen soll. In diesen Dokumentationen sind für den Kunden jedoch keine Informationen über den Prozess zu finden, sondern Möglichkeiten der Nutzung des Tools. Nichtsdestotrotz bieten Azure DevOps und Jira vielerlei Optionen, die Funktionen im Tool auf den im Unternehmen implementierten Prozess anzupassen. Dieses Feature kann demzufolge nicht direkt die Probleme im Release Management lösen, jedoch kann der Prozess in der Software so angepasst werden, dass diese Probleme durch die richtige Darstellung im Tool verringert oder vermieden werden können. Ob diese Dokumentation über die Adaptierbarkeit des Tools für die identifizierten Probleme als Lösung gesehen werden kann, wird in der letzten der fünf Hypothesen überprüft, die lautet:

„Die online verfügbaren Dokumentationen über die Programme Azure DevOps und Jira helfen die Probleme im Release Management zu minimieren.“

4.2 Erstellung Interviewleitfaden

Vor der Erstellung des Leitfadens muss entschieden werden, in welcher Form das Interview stattfinden soll. Zur Wahl stehen in diesem Fall mehrere Methoden, die für die unterschiedlichen Problemstellungen passend sind. Für diese Arbeit wird das problemzentrierte Interview

herangezogen. Im Interviewleitfaden wird also das Ziel verfolgt, die Interviewpartner nach Erfahrungen und Empfinden zu befragen und dadurch Aufschlüsse zu einer konkreten Problemstellung zu bekommen.

Da es sich in dieser Arbeit um die Probleme im Release Management handelt, wird der Leitfaden in diese Richtung erstellt, sodass durch die Interviews verschiedene Meinungen zu diesem Thema gesammelt werden können. Dadurch sollen die im vorigen Kapitel aufgestellten Hypothesen bestätigt oder widerlegt werden können. Ziel des Interviews ist es jedoch den Interviewpartner viel Freiraum in der Beantwortung der Fragen zu geben und nur gelegentlich einzugreifen. Der Befragte soll seine Meinungen kundtun und durch die geringe Anzahl der Fragen jedoch in die richtige Richtung gelenkt werden, sodass die Antworten in Bezug zu den Hypothesen stehen.

4.3 Kategorisierung der Fragen

Für die Erstellung des Leitfadens werden zu Beginn Kategorien für die Fragenblöcke gewählt. Die in Abschnitt 4.1.3 erstellten Hypothesen dienen hier bereits als Orientierung, die für die Gruppierungen herangezogen werden können. In diesem Sinn werden basierend auf die Hypothesen Kategorien erstellt, sodass zu allen in den Fragenblöcken Informationen gesammelt werden können. Aus diesem Punkt stammt die erste Gruppierung der Fragenblöcke, die in Tabelle 2 zu sehen ist.

Kategorie
Einleitung
1. Fragenblock: Erfahrung
2. Fragenblock: Nutzung
3. Fragenblock: Prozess
4. Fragenblock: Rollen
5. Fragenblock: Probleme

Tabelle 2: Kategorisierung Interviewfragen

Mithilfe dieser Tabelle können nun Fragen erstellt werden, die zu den Hypothesen Informationen bringen sollen. Hierfür werden die bereits erstellten Hypothesen zu den in Tabelle 2 zu sehenden Kategorien zugewiesen. Mit dieser Zuweisung wird im Interview die Richtung zu den benötigten Informationen abgeleitet. Die Struktur wird im Interview jedoch nicht als fixe Vorlage gesehen. Die Fragen aus den Blöcken können auch in Bezug zu Hypothesen aus einer anderen Kategorie stehen.

Kategorie	**Zuweisung Hypothese**
Einleitung	
1. Fragenblock: Erfahrung	Hypothese 1
2. Fragenblock: Nutzung	Hypothese 2
3. Fragenblock: Prozess	Hypothese 3
4. Fragenblock: Rollen	Hypothese 4
5. Fragenblock: Probleme	Hypothese 5

Tabelle 3: Fragenkategorie Zuweisung Hypothesen

Nach der Zuweisung der Kategorien zu den Hypothesen werden nun konkrete Fragen zusammengestellt, die in den jeweiligen Blöcken Informationen zu den Hypothesen bringen sollen. Durch den problemzentrierten Ansatz im Experteninterview sind die Schlüsselfragen der jeweiligen Blöcke offen gestellt und geben den Befragten viel Spielraum, die Fragen zu beantworten. Der gesamte Interviewleitfaden ist im Anhang schlussendlich beigelegt.

5 ERGEBNISSE

Nach der Durchführung der Experteninterviews werden diese nun analysiert. Im Anhang A – F sind die Transkripte der fünf Interviews abgelegt. Während sich fünf der sechs Interviews konkret an den Interviewleitaden orientierten, versuchte Interview 5 mit einem Experten aus dem Unternehmen Atlassian, Einblicke in die Entwicklung des Tools, das als Unterstützung für das Release Management dient, zu geben. Mithilfe der qualitativen Inhaltsanalyse von Phillipp Mayring (Mayring 2021) wurde der Inhalt der Interviews analysiert und anschließend in Bezug auf die Hypothesen und Forschungsfrage vorgestellt.

5.1 Qualitative Inhaltsanalyse

Als ersten Schritt in der qualitativen Inhaltsanalyse wurden die bereits erwähnten Interviews durchgeführt. Mithilfe des Interviewleitfadens konnte bereits in der Durchführung Bezug auf die Forschungsfrage und die Hypothesen genommen werden. Im zweiten Schritt musste die Wahl der Vorgehensweise getroffen werden. Ziel hier ist es eine der folgenden zwei Methoden zu wählen, um die transkribierten Interviews beziehungsweise die Aussagen der Experten zu kategorisieren. Die erste Möglichkeit, um die Kategorien der Interviewteile zu bestimmen, ist eine spontane Wahl eines Begriffes, der während der Ausarbeitung der Antworten definiert wird. Diese Methode nennt sich induktive Kategorienentwicklung. Das Gegenstück dieser Methode heißt deduktive Kategorienanwendung und ist mit einer weitaus höheren Literaturrecherche verbunden als die induktive Methode, da basierend auf dem theoretischen Rahmen der Arbeit bereits Kategorien definiert sein müssen, welche dann den Aussagen zugewiesen werden.

Aufgrund des sehr breiten Themengebietes rund um das Release Management konnten vor der Auswertung der Interviews keine passenden Kategorien aus der Literatur gebildet werden, die für die Auswertung passend sind. Aus diesem Grund fiel die Entscheidung auf die induktive Kategorienentwicklung.

5.2 Kodierungsleitfaden

Mit der Wahl der passenden Strategie für die Kategorien konnte die Kodierung der Transkripte gestartet werden und ein Kodierungsleitfaden definiert werden. In diesem Schritt wurde mithilfe von Excel eine Tabelle mit allen Antworten der Interviews erstellt. Die Antworten konnten teilweise identisch verwendet werden, jedoch besaßen auch einige Antworten Aussagen zu mehreren Kategorien.

Aufgrund der Empfehlung nach 10 – 50% einen Cut zu machen und die bisher erstellten Kategorien zu analysieren, erfolgte eine kurze Auswertung der Kategorien, wo unter anderem

die Anzahl der Zuweisungen gezählt werden. Hier werden neben dieser Kennzahl auch die Bezeichnungen und eine mögliche Zusammenführung überprüft. Ziel in diesem Schritt ist es, Kategorien definiert zu haben, die einerseits eine ausgeglichene Anzahl an Zuweisung besitzen und andererseits den Inhalt korrekt widerspiegeln, sodass diese auch auf die Forschungsfrage hinweist.

Aus diesem Schritt entstanden in der Kodierung, die in Tabelle 3 abgebildeten Kategorien mit den ergänzten Kürzeln, die in der Kodierungstabelle verwendet wurden.

Kategorie	**Kürzel**	**Beschreibung**
Erfahrung	K1	Berufserfahrung des Befragten
Rolle	K2	Rolle des Befragten
Verwendung	K3	Wie wird das Tool verwendet?
Team	K4	Teamstruktur
Prozess	K5	Beschreibung des Release Prozesses
Tool	K6	Erfahrungen mit dem Tool
Tracking	K7	Verwendung für Arbeitsverfolgung
Arbeitsmethode	K8	Entwicklungsmethode und Arbeitsweise
Kommunikation	K9	Tool als Kommunikationsunterstützung
Automatisierung	K10	Automatisierungen im Prozess
Metriken	K11	Kennzahlen
Deployment	K12	Entwicklungs- und Deploymentprozess
Problembehandlung	K13	Wie werden Probleme analysiert?
Planung	K14	Tool dient zur Planung der Arbeit
Probleme Team	K15	Probleme aufgrund interner Teamstruktur
Probleme Prozess	K16	Probleme aufgrund des Prozesses
Probleme Tool	K17	Probleme für die das Tool verantwortlich ist
Probleme Extern	K18	Probleme mit der Kundenseite
Add-On	K19	Tool muss erweitert werden
Benutzerfreundlichkeit	K20	Erste Kontakte zum Tool und Lernprozess

Tabelle 4: Kategorien Inhaltsanalyse

Nach der Erstellung der Kodierungskategorien konnten im Anschluss die noch ausständigen Interviews kodiert werden und somit die in Tabelle 4 zu findende Kodierung vervollständigt werden und mit der weiterführenden Analyse fortgesetzt werden.

5.3 Vorstellung Ergebnisse

Nach der Erstellung der Kodierungskategorien können nun weitere Analysen gestartet werden. Im ersten Schritt werden nun die Häufigkeiten der Zuweisungen einer Kategorie in einer Tabelle dargestellt.

Kürzel	**Kategorie**	**Häufigkeit**
K5	Prozess	20
K3	Verwendung	19
K10	Automatisierung	19
K6	Tool	19
K8	Arbeitsmethode	17
K4	Team	16
K2	Rolle	14
K7	Tracking	12
K1	Erfahrung	11
K12	Deployment	10
K16	Probleme Prozess	6
K11	Metriken	6
K15	Probleme Team	6
K	Add-On	5
K17	Probleme Tool	5
K13	Problembehandlung	5
K9	Kommunikation	4
K14	Planung	4
K20	Benutzerfreundlichkeit	4
K18	Probleme Extern	2

Tabelle 5: Auswertung qualitative Inhaltsanalyse

In der oben angeführten Tabelle 5 ist auf dem ersten Blick zu erkennen, dass Aussagen zu der Kategorie Prozess am häufigsten getroffen wurden. Neben der genannten Kategorie traten auch hohe Häufigkeiten bei der Automatisierung und bei der Verwendung des Tools auf. Im Gegenzug bekamen externe Probleme, Planung, Kommunikation sowie Benutzerfreundlichkeit keine hohe Aufmerksamkeit.

Aus diesen absoluten Zahlen kann abgelesen werden, mit welchen Themen das Release Management selbst stark in Verbindung steht. Aufgrund der Tatsache, dass in den Interviews jedoch mehr oder weniger der Fokus auf den Prozess selbst und der Anwendung des Tools gelegt wurde, darf nicht daraus geschlossen werden, dass eine falsche oder zu geringe Anwendung des Tools zu Problemen im Release Management führt.

Wie in der Erstellung des Interviewleitfadens bereits erwähnt, wurden die Experten bewusst aus verschiedenen Rollen gewählt. Während einerseits Toolexperten großes Wissen über das Tool, dessen Stärken sowie auch Schwächen preisgegeben haben, fokussierten Entwickler und auch der SCRUM Master mehr den Prozess und die Teamarbeitsweise selbst. Außerdem konnten mit den Interviews der Personen 5 und 6 detailliertere Einblicke aus der Engineering Perspektive erforscht werden. Einerseits die Erfahrung von Person 5 aus dem Großunternehmen und andererseits Person 6 mit dem Einblick in die Entwicklung eines Teams mit organisatorischen Änderungen und Anpassungen. Im nächsten Schritt werden diese unterschiedlichen Perspektiven auf das Thema differenziert und detaillierter beschrieben.

5.4 Perspektiven differenzieren

In diesem Abschnitt werden nun die in Abschnitt 5.3 erforschten Ergebnisse detaillierter und in den verschiedenen Perspektiven getrennt analysiert. Zu Beginn zeigt Tabelle 5 eine Zuordnung der Befragten zu einer bestimmten Rolle im Release Prozess. Basierend auf dieser Zuteilung werden im nächsten Schritt die Häufigkeiten der Kategorien getrennt voneinander aufgezeigt und analysiert. In weiterer Folge werden im Anschluss die Rollen SCRUM Master und Softwareentwickler zu der Prozessperspektive zusammengefasst und die Rolle des Toolexperten mit dem DevOps Engineer als Toolperspektive verbunden.

Interview	**Person**	**Rolle**
A	1	SCRUM Master
B	2	Softwareentwickler
C	3	Toolexperte
D	4	DevOps Engineer
E	5	Development Leader
F	6	Development Leader

Tabelle 6: Interview-Rollen-Zuweisung

5.4.1 **SCRUM Master**

Als erste Analyse wird die Rolle des SCRUM Masters und dessen Aussagen zum Release Management analysiert. Gleich wie im Schritt davor wird hier eine Häufigkeitstabelle erstellt, worin die Anzahl der Zuweisungen einer Kategorie zu sehen ist.

Kürzel	**Kategorie**	**Häufigkeit**
K8	Arbeitsmethode	5
K3	Verwendung	5
K10	Automatisierung	5
K7	Tracking	5
K6	Tool	3
K12	Deployment	3
K11	Metriken	3
K16	Probleme Prozess	3
K9	Kommunikation	3
K1	Erfahrung	2
K2	Rolle	2
K13	Problembehandlung	2
K18	Probleme Extern	2
K4	Team	2
K5	Prozess	2
K14	Planung	1
K15	Probleme Team	1

Tabelle 7: Kodierung SCRUM Master

Diese Tabelle zeigt, dass der Fokus des Gesprächs auf bestimmte Kategorien gerichtet war. Am häufigsten angesprochen wurden die Themen rund um die Arbeitsmethode, Verwendung des Tools, was bisher automatisiert wurde und wie das Tool für das Issue Tracking eingesetzt wird. Dieses Ergebnis ist jedoch nicht überraschend, da der Entwicklungsprozess und auch in dem Fall die Arbeitsweise zu den Hauptaufgabenbereichen des SCRUM Masters zählen (Noll et al. 2017). Weiteres auffällig zu erkennen ist, dass weder die Probleme am Tool aufkamen noch die Benutzerfreundlichkeit in den Arbeitsbereich hineinfiel. Da jedoch das Thema Benutzerfreundlichkeit auch nicht explizit angesprochen wurde, kann dies auch als positiv gewertet werden, da es keine Probleme gab, die das Tool betrafen und eine einfache Verwendung daraus impliziert werden kann. Zu erwähnen ist außerdem, dass in dieser Rolle

alle Kodierungen zu externen Problemen erstellt wurden. Dies kann einerseits der Organisation geschuldet sein, dass in diesem Fall das hohe Maß an Kundenkontakt vorhanden ist. Andererseits ist in dem Interview mit Person A bereits erwähnt worden, dass die Zusammenarbeit mit dem Kunden bereits verbessert wurde. Ansonsten spiegeln die Auswertungen den Gesamtanalyse sehr gut wider.

5.4.2 **Softwareentwickler**

Als nächstes wird die Rolle des Softwareentwicklers und dessen Aussagen analysiert. In Tabelle 8 ist die Kodierung des Interviews zu sehen, was weitere Aufschlüsse über das Release Management und dessen betroffenen Themen gibt. Es ist klar zu erkennen, dass sich in diesem Gespräch der Inhalt sehr stark an den Prozess und an die Verwendung orientiert hat. Unabhängig davon, was der Inhalt in diesen Aussagen konkret war, kann gesagt werden, dass der Prozess selbst im Release Management aus der Softwareentwickler eine sehr große Rolle spielt. Außerdem als positiv zu sehen ist die hohe Anzahl der Kodierung zur Verwendung.

Kürzel	**Kategorie**	**Häufigkeit**
K5	Prozess	7
K3	Verwendung	6
K8	Arbeitsmethode	3
K10	Automatisierung	3
K6	Tool	3
K17	Probleme Tool	3
K4	Team	2
K15	Probleme Team	2
K7	Tracking	1
K12	Deployment	1
K11	Metriken	1
K16	Probleme Prozess	1
K2	Rolle	1
K1	Erfahrung	1
K13	Problembehandlung	1

Tabelle 8: Kodierung Softwareentwickler

Weiteres zu erwähnen ist, dass in diesem Fall die meisten Probleme auf der Seite des Tools zu finden sind. Im Interview wurde hierfür konkret der Fall für ein nicht im Unternehmen

vorhandenes Betriebssystem erwähnt, was für die Probleme gesorgt hat, weswegen diese Aussage mit Vorsicht zu betrachten ist, da dieses Problem am Tool nicht für alle gilt.

5.4.3 **Toolexperte**

In folgenden Abschnitt wird detaillierter auf das Ergebnis der Kodierung von Person C, dem Toolexperten, eingegangen. Die Einteilung stammt von der Rolle des Befragten, die die Person derzeit ausführt. Konkret handelt es sich hier um den Berufstitel Jira- und Confluence Expert. Aus der Kodierung kann man erkennen, dass in dieser Rolle keine Kategorie besonders heraussticht. Die ähnlich häufige Erwähnung in den Bereichen rund um die Verwendung des Tools und der Automatisierung, sowie vor allem der Erfahrung deuten darauf hin, dass der Prozess in diesem Unternehmen bereits klar vorgegeben ist, weswegen dieser selbst nur einmal erwähnt wurde.

Wie auch im Interview selbst bestätigt, wird hier der Ansatz verfolgt, dass das Tool auf den eigenen Prozess angepasst wird, sodass nicht der Prozess an das Tool angepasst wird, sondern das Tool auf den Prozess. Die in diesem Fall sehr hohe Erfahrung in der Nutzung des Tools hilft hier, diese Strategie erfolgreich umzusetzen. Ein weiteres Indiz für diese Methodik ist das häufige Erwähnen der selbst entwickelten Add-Ons.

Kürzel	**Kategorie**	**Häufigkeit**
K1	Erfahrung	4
K3	Verwendung	3
K10	Automatisierung	3
K6	Tool	3
K4	Team	3
K2	Rolle	3
K19	Add-On	3
K20	Benutzerfreundlichkeit	2
K5	Prozess	1
K8	Arbeitsmethode	1
K17	Probleme Tool	1
K11	Metriken	1

Tabelle 9: Kodierung Toolexperte

5.4.4 **DevOps Engineer**

Eine weitere befragte Rolle ist die des DevOps Engineers. Hierbei ist zu erwähnen, dass auch die befragte Person A, die als SCRUM Master fungiert, eine DevOps Engineering Rolle innehat, wobei dieser Aufgabenbereich in die Operational Zeit einfließt und damit getrennt von dieser behandelt wird.

Kürzel	**Kategorie**	**Häufigkeit**
K7	Tracking	5
K8	Arbeitsmethode	4
K3	Verwendung	3
K2	Rolle	3
K5	Prozess	3
K1	Erfahrung	2
K6	Tool	2
K4	Team	2
K10	Automatisierung	1
K20	Benutzerfreundlichkeit	1
K17	Probleme Tool	1
K15	Probleme Team	1
K16	Probleme Prozess	1
K9	Kommunikation	1
K14	Planung	1

Tabelle 10: Kodierung DevOps Engineer

Aus der Tabelle 10, der Kodierung des DevOps Engineers, ist zu lesen, dass der Fokus auf das Tracking sowie die Arbeitsmethode lag. Die Differenzierung und Erwähnung verschiedener Probleme weisen auf eine breite Nutzung des Tools hin.

5.4.5 **Entwicklungsleiter**

Als Letztes wird nun die Kodierung der Entwicklungsleiter im Detail analysiert. In diesem Gespräch wurde viel über die Deploymentstrategie und über die Entwicklung der Arbeitsweise sowie welche Anpassungen passieren mussten, um ein hochqualitatives Release Management zu erschaffen, gesprochen. Im Mittelpunkt dieser Interviews lag das Tool. Der Entwicklungsleiter ist in diesen Beispielen dafür zuständig, dass der Release Prozess und vor allem der Entwicklungsprozess eine Software garantiert, die am Markt bestehen kann. Aus

diesem Grund beschäftigt dieser sich vor allem mit dem Prozess, der Automatisierung und das Team. Und um diese Komponenten effizient zu vereinigen, benötigt es ein Tooling, das dafür sorgt, dass diese Teile im Zusammenspiel den definierten Release Prozess durchführen können. Vor allem in Interview 6 wurde das selbst entwickelte Add-On des Öfteren erwähnt. Mithilfe dieser Ergänzungssoftware hat das Unternehmen das Ziel erreicht hat, die Probleme, die aufgrund der Komplexität nicht mit dem Tool selbst gelöst werden konnten, zu lösen.

Kürzel	**Kategorie**	**Häufigkeit**
K6	Tool	8
K5	Prozess	7
K10	Automatisierung	7
K4	Team	7
K12	Deployment	6
K2	Rolle	5
K8	Arbeitsmethode	4
K1	Erfahrung	2
K15	Probleme Team	2
K19	Add-On	2
K13	Problembehandlung	2
K3	Verwendung	2
K16	Probleme Prozess	1
K14	Planung	1
K20	Benutzerfreundlichkeit	1
K11	Metriken	1
K7	Tracking	1

Tabelle 11: Kodierung Entwicklungsleiter

5.5 Kurzzusammenfassung der Ergebnisse

Mithilfe der qualitativen Inhaltsanalyse von Mayring (Mayring 2021) wurden die Ergebnisse der Experteninterviews erarbeitet. Nach der Paraphrasierung der Transkripte aller Interviews konnten unter der Verwendung der induktiven Kategorienbildung jeder Aussage eine Kodierung zugewiesen werden. Um passende Kategorien zu erstellen, wurde der Kodierungsprozess nach zwei Interviews unterbrochen und validiert, ob es eine ähnlich verteilte Häufigkeit zwischen den

Kategorienzuweisungen gibt sowie ob diese Kategorien das Forschungsziel unterstützen. Nachdem diese Validierung als passend eingestuft wurde, konnten die weiteren Interviews kodiert werden und ausgewertet werden.

Im nächsten Schritt konnten die Häufigkeiten der Kategorien im gesamten Datensatz berechnet werden. Mit dieser einfachen statistischen Kennzahl konnte eine Relevanz der Kategorie im Bereich des Release Managements erfasst werden. Weiteres gab das Ergebnis Informationen über weniger relevante Themen im Release Management, weswegen in diesem Schritt schon Hinweise auf eine Richtung für die Bestätigung und auch Widerlegung mancher Hypothesen erkennbar wurde.

Klar erkennbar aus der Auswertung der qualitativen Inhaltsanalyse war die Relevanz in Bezug auf die Verwendung. Einerseits bestätigt diese Kennzahl, dass es sich bei den Befragten tatsächlich um Experten in dem Bereich der Tools handelte. Andererseits wurde dadurch auch ersichtlich, dass im Release Management die Verwendung der Tools essenziell ist. Weitere wichtige Themen, die aus den Kennzahlen ersichtlich wurden, sind die Arbeitsmethode, Automatisierung und auch der Prozess. Diese Kategorien stehen eng in Zusammenhang zueinander und bilden, wie im Theorieteil bereits vermerkt, den Kern des Release Managements.

Zu den am wenigsten erwähnten Kategorien zählten unter anderem die externen Probleme, die Benutzerfreundlichkeit sowie die Notwendigkeit beziehungsweise die Verwendung eines Add-Ons und der konkrete Umgang mit Problemen. Vor allem der Punkt mit den Add-Ons ist aussagekräftigt, da die Tools bereits so konzipiert sind, dass die dringende Notwendigkeit einer selbst entwickelten Software zur Ergänzung nicht vorhanden ist. Weiteres sei zu erwähnen, dass die Benutzerfreundlichkeit der Tools keinen großen Einfluss im Release Management hat und mit der bereits erwähnten großen Verwendung im Zusammenhang steht.

6 DISKUSSION

Trotz bereits vorhandener Tools zur Unterstützung des Release Managements kommt es nach wie vor zu Problemen in den IT-Unternehmen, den Prozess, der definiert wurde, ständig zu verfolgen. Immer wieder treten Stolpersteine auf und auch mithilfe dieser Tools werden Meilensteine und Liefertermine nicht eingehalten. Im Zuge der Interviews und auch der theoretischen Ausarbeitung dieses Themengebietes kamen Informationen auf, die in manchen Fällen für Lösungen der Probleme sorgen können. Diese Informationen werden nun in diesem Abschnitt der Arbeit zusammengefasst, um einen Überblick der neuen Erkenntnisse zu gewährleisten. Nach der Darstellung der Ergebnisse werden nun im Diskussionsteil konkret Stellung zu den Ergebnissen bezogen. Das Ziel dieses Abschnittes soll eine Liste von Handlungsempfehlungen sein, die mittel- und auch teilweise großständische IT-Unternehmen anwenden können, um ihr Release Management zu verbessern.

6.1 Einleitung Diskussion

Die zu Beginn definierte Forschungsfrage *„Wie setzen mittelständische IT-Unternehmen Azure DevOps und Jira ein, um die Qualität im Release Management zu steigern?“* soll in diesem Abschnitt nun beantwortet werden und als Ergebnis eine Liste an Handlungsempfehlungen anbieten. Um jedoch zu der Antwort zu kommen, müssen die Ergebnisse der Interviews analysiert werden und auf die Hypothesen und die Forschungsfrage gerichtet werden, sodass mithilfe der erforschten Informationen, neue Erkenntnisse in diesem Bereich darstellen zu können.

Als Grundlage dient in dieser Arbeit die Analyse einer definierten Gruppe an IT-Unternehmen, die Tools wie Jira oder Azure DevOps im Unternehmen für das Release Management nutzen. In diesem Fall wurden die Grenzen bei Unternehmen von 50 bis 1000 Mitarbeiter gesetzt, da sich vor allem die Arbeitsweisen stark unterscheiden können. Aus einem Interview kamen in diesem Sinne auch Informationen aus Erfahrungswerten der beruflichen Vergangenheit bei Atlassian, das Jira und weitere wichtige Tools wie Confluence oder Bitbucket für das Release Management anbietet.

Aus der Literaturrecherche konnten zusätzlich bereits Probleme im Release Management identifiziert werden. Des Weiteren wurden in der theoretischen Ausarbeitung die unterschiedlichen Möglichkeiten zur Softwareentwicklung nähergebracht, die in dem Teil der Diskussion eine sehr wichtige Rolle spielt. Aus all den Informationen und Erkenntnissen des Theorieteils, der Hypothesen sowie der Forschungsfrage und den Experteninterviews werden nun also die Ergebnisse dokumentiert.

6.2 Neue Erkenntnisse dokumentieren

In erster Linie konnten durch die Experteninterviews Informationen zum Release Management aus verschiedenen Teams und auch Unternehmen erfragt werden. Informationen über die Verwendung der Tools sowie Prozesseigenschaften und auch Entwicklungsmethoden, die in diesem Bereich auch eine entscheidende Rolle spielen, konnten Einblick in verschiedene Unternehmen geben. Vor allem Interview 5 gab tiefere Einblicke, wie ein Großunternehmen ein Release Management handhabt.

6.2.1 **Analyse der Hypothesen**

In Hypothese A wurde der Nutzungszeitraum des Tools genauer betrachtet. Mit der Hypothese *„Je höher der Nutzungszeitraum des Tools ist, desto kürzer ist die Dauer eines Deployment“* soll überprüft werden, ob sich der Release Prozess über die Jahre der Nutzung beschleunigt hat oder nicht. In diesem Fall konnte die Hypothese nicht bestätigt werden. Es konnten von allen Befragten Informationen eingeholt werden, wie sich der Release Prozess über die Jahre mit der Nutzung entwickelt hat. Während im Interview mit Person 1 mehr die Anpassung des Entwicklungsprozesses dazu führte, dass das Release Management über die Jahre an Qualität zugenommen hat, wird Azure DevOps von Person 2 erst seit knapp zwei Jahren verwendet. Jedoch ist auch dort trotz des bisher noch eher geringeren Knowhows über das Tool ein Release Prozess zu erkennen, der trotz der internen Schwierigkeiten im Projektteam einen ausgereiften Grad hat, gemessen an der Automatisierung.

Person 3 gab in diesem Zusammenhang Informationen zu der Ausrollung des Tools und wie der erste Kontakt zu der neuen Software war. Aus den Antworten hierbei konnte man erkennen, dass vor allem in den IT-affinen Unternehmen die Einführung eines neuen Tools zu keinen großen Problemen geführt hat, weswegen dort auch der Ansatz verfolgt werden konnte, das Tool an den vorhandenen Prozess anzupassen und mit der Nutzungsdauer stieg schlussendlich nur das Ausmaß der Funktionalität des Tools. Eine Kernaussage aus diesem Interview, die auch in weiteren Antworten sich herausstellte war, dass sich der Release Prozess selbst ständig weiterentwickelt.

Aus Interview 4 kam die Information hervor, dass sogar zu einem späteren Zeitpunkt die Qualität des Release Management nachgelassen hat, da die grundsätzliche Nutzung zurückging. Jedoch kann man im Vergleich zu Interview 3 auch erkennen, dass eine hohe Erfahrung an der Nutzung des Tools nicht ein gutes Release Management mit sich bringt, weswegen Hypothese A als nicht bestätigt angesehen werden kann. Jedoch kann durchaus bestätigt werden, dass bei der Prozessentwicklung des Release Managements das Tool selbst eine sehr große Unterstützung darstellt.

Auch Hypothese B befasst sich mit der Nutzung des Tools und auch mit der Dauer eines Releases. Mit der Aussage *„Je mehr Stunden pro Woche das Tool genutzt wird, desto kürzer ist die Dauer eines Releases“* wird überprüft, ob eine regelmäßige Verwendung des Tools auch einen positiven Einfluss auf die Dauer eines Releases hat. Wie sich aus den Interviews herausstellte, kann pauschal die Durchlaufzeit eines Codeteiles nicht gemessen werden, da

Faktoren wie die Kundenakzeptanz, Dauer des Testens oder auch die große Menge an Operational Tasks eine schnelle Durchführung verhindern. Nichtsdestotrotz kam durch das Interview von Person 4 zum Vorschein, dass eine Vernachlässigung der Nutzung des Tools zu einem Rückgang in der Qualität des Release Managements führte. Zusätzlich konnte der Befragte aus Interview 3 durchaus mit einer Wochenstundenzahl von 35 Stunden im Tool für die Höchstzahl sorgen, was sich laut der Qualität im Release Management auch widerspiegelt und in Form von Erfahrung erkennbar wird. Aus diesen Gründen kann Hypothese B als bestätigt angegeben werden.

In Hypothese C stand der Prozess selbst und das Tool im Vordergrund. Während in dieser Arbeit meist von der Differenzierung und dem Vergleich zwischen Azure DevOps und Jira geschrieben wurde, soll mithilfe dieser Hypothese bestätigt werden, dass die Wahl des Tools keine Auswirkungen auf die Qualität im Release Management hat. In diesem Zusammenhang darf nicht unerwähnt bleiben, dass beide Tools nur ein Teil einer gesamten Suite ist. Während Atlassian neben Jira mit Confluence, Bitbucket und viele weiteren im Verhältnis kleineren und unbekannteren Produkten im Bereich der Kollaboration am Markt vertreten ist, kann Azure DevOps von Microsoft vor allem mit der gesamten Microsoft Toollandschaft Argumente sammeln.

Ein Punkt, der jedoch aus allen Interviews und vor allem aus Interview 5 zum Vorschein kam, war, dass das Tool selbst nur als Unterstützung dient. Weitaus wichtiger in diesem Fall ist die Definierung eines Prozesses, der funktioniert. Wie in Interview 3 und 5 zu erkennen war, spielt in diesen Fällen sogar die Anpassung des Tools eine große Rolle, da komplexe Themen und auch individuelle Prozesse nicht in der Grundfunktionalität der Tools enthalten sind. Auch erwähnt wurden selbst entwickelte Erweiterungen oder wie in Interview 5 sogar eigene Tools zur Automatisierung. Diese Faktoren benötigen jedoch auch ein gewisses Maß an Ressourcen, die für diese Weiterentwicklungen erforderlich sin, was in Interview 3 hervorging, als die Meinung kundgetan wurde, dass ab einer gewissen Unternehmensgröße eine Anpassung des Tools durchaus notwendig ist.

Auch im Interview mit Person 1 steigerte sich die Qualität des Release Management nicht nur durch die Verwendung des Tools, sondern über die Entwicklung eines funktionierenden Prozesses. Noch ersichtlicher wurde es durch das Interview 5, wo der Fokus der Antworten klar auf den Entwicklungsprozess selbst lag und dieser ausschlaggebend für die Qualität des Releases ist. Aus diesem Grund kann auch hier die Hypothese bestätigt werden, dass ein funktionierender Release Prozess toolunabhängig ist, da sich auch vor allem die Grundfunktionalität dieser zwei Komponenten stark ähnelt.

Hypothese D fokussiert sich auf die Rollen im Release Management. Während in fünf von sechs Interviews der agile Ansatz in der Entwicklung der Software verfolgt wird und auch in diesen Fällen konkret nach SCRUM oder an Kanban orientiert gearbeitet wird, kann dort auch grundsätzlich schon eine Teamstruktur erkennbar sein. Neben den Softwareentwicklern spielen in SCRUM der SCRUM Master und der Product Owner eine große Rolle. Das schlussendlich aus zwischen fünf bis zehn Personen große Team verfolgt das Ziel in regelmäßigen Abständen eine funktionierende Software den Kunden übergeben zu können. In dieser Arbeitsmethodik ist

jedoch keine Rolle für die Analyse eines Supporttools für das Release Management vorgesehen. Das Interview 3 wurde in diesem Fall mit einem Jira und Confluence Experten geführt, was jedoch zeigt, dass es auch durch die Bestätigung von Interview 5 auch von Unternehmen gehandhabt wird, eigene Rollen oder sogar Teams im Entwicklungsprozess zu integrieren, um Funktionalitäten des Tools in den Release Prozess miteinzubringen.

Als Gegenstück dazu, gibt es in Team von Person 1 keine zuständige Rolle dafür und Funktionalitäten kommen über Teammitglieder in den Prozess. Auch in Interview 2 und 4 kamen keine Informationen über eigene Rollen für die Toolverwendung, wobei durch erweiterte Rollenzuweisung der Person Ressource frei sind, um die Funktionalität des Tools in den Prozess zu integrieren. In diesem Punkt spielt dann die Größe des Unternehmens selbst eine Rolle, da sich dann auch entscheidet, ob eine eigene verantwortliche Person für das Tool finanzierbar ist. Aus diesem Grund ist es hier auch nicht möglich eine endgültige Antwort zu geben, ob sich die Hypothese „Eine erweiterte Rollenverteilung für die Tools verringert die Anzahl an Problemen im Release Prozess“ bestätigen lässt. Jedoch kann hier auch nicht ausgeschlossenen werden, dass die Rolle eine Verringerung der Anzahl an Problemen mit sich bringt.

Zu guter Letzt wird Hypothese E *„Die online verfügbaren Dokumentationen über die Programme Azure DevOps und Jira helfen die Probleme im Release Management zu minimieren“* überprüft. Aus Interview 4 kam aus dem Aussagen heraus, dass die Grundexpertise über ein Selbststudium angeeignet wurde, woraus geschlossen werden kann, dass eine qualitativ hochwertige Dokumentation vorhanden ist und auch für die Praxis online Ressourcen vorfügbar sind. Weiteres weist auch Interview 1 vor, dass Features des Tools während eines Sprints beziehungsweise in der täglichen Arbeit erlernt werden. Auch dieser Punkt gibt Informationen über online Dokumentationen, denn ohne dieser dokumentierten Use Cases werden Entwickler nicht fündig und die Suche nach Lösungen weist auf Probleme hin. Aus diesem Grund kann Hypothese E bestätigt werden, dass die online verfügbaren Ressourcen für Problembehebungen ausreichen und das Release Management verbessern kann.

Zusammengefasst konnten durch die Interviews die Hypothesen B, C und E bestätigt werden, da einerseits die permanente Verwendung des Tools essenziell für die Qualität im Release Management ist. Außerdem bestätigten die Aussagen, dass das Tool selbst nicht über die Qualität des Release Managements bestimmt und der Prozess selbst, der sich über die Zeit weiterentwickelt und ändert einen weitaus größeren Impact auf die Qualität hat. Zu guter Letzt konnte auch bestätigt werden, dass die online verfügbaren Informationen, Handbücher sowie Nutzungsleitfäden durchaus als eine Problemlösung fungieren können.

6.2.2 **Analyse der Forschungsfrage**

In diesem Abschnitt werden die Ergebnisse aus den Interviews in Bezug zu der Forschungsfrage *„Wie setzen mittelständische IT-Unternehmen Jira und Azure DevOps ein, um die Qualität im Release Management zu steigern?“* gestellt. Das Ziel in diesem Absatzes ist eine Sammlung von Erkenntnissen, die in den Handlungsempfehlungen verwendet werden

können. Wie eingangs beschrieben werden bei der Analyse konkret Unternehmen der Größenordnung 50 bis 1000 Mitarbeiter analysiert. Es wurden Interviews geführt mit Personen verschiedener Rollen im Unternehmen, die teilweise aktiv bei der Produktiventwicklung teilnehmen und andererseits eine Unterstützung im Release darstellen. Durch die Informationen aus den Interviews konnten die verschiedenen Nutzungsweisen der Tools erforscht werden, um daraus einen Leitfaden zu erstellen, der Aufschlüsse über die Problemlösung im Release gibt.

Vor allem die Frage nach dem „Wie“ die Unternehmen die Tools verwenden ist in diesem Umfeld interessant. In zwei von sechs Interviews wurde das Tool nicht geändert und an den eigenen Prozess angepasst, während vier von sechs Experten das Tool um Funktionalitäten erweitert hat, damit der bereits bestehende Prozess weitergeführt und durch das Tool ergänzt werden kann. Auch in Handbüchern von Azure DevOps (Rossberg 2019) wird geraten den Prozess nicht auf das Tool anzupassen. Die Stärken dieser sind vor allem, dass sie flexibel sind und an eigene Anforderungen angepasst werden können. Aus diesem Grund ist es nicht empfehlenswert, den Prozess an das Tool anzupassen. Jedoch muss die Teamstruktur berücksichtigt werden und in den zwei Fällen, in dem Azure DevOps nicht angepasst wird, gibt es auch nicht die personellen Ressourcen das Tool anzupassen. Aus Interview 2 ist hervorgekommen, dass es kein kontinuierliches Team gibt. Weiteres arbeitet die Person auch nicht nur an einem Projekt, sondern an mehreren, was dazu führt, dass auch die Kunden nicht immer dieselben sind. So gesehen muss ein Release Prozess von der Organisation selbst kommen, da in der Entwicklung Unterschiede in den Projekten aufkommen.

Weiteres war ein großer Punkt, der aus den Interviews hervorkam, die Dokumentation aller Projekte. Aus allen Aussagen konnte die Information entnommen werden, dass die Tools vor allem in Meetings und in Zeiten, in denen das Thema Home-Office und Digitalisierung auch gezwungenermaßen stark forciert wurde, eine großartige Unterstützung darstellen. Außerdem profitiert das Projekt Controlling von der Trackingfunktionalität, die in beiden Tools ein Keyfeature ist. Dadurch ermöglicht das Tool während eines Releases ständig den aktuellen Status des Fortschritts verfolgen zu können.

Außerdem spielte auch das Thema rund um die Automatisierung und den Entwicklungsmethoden eine große Rolle. Im Interview 1, 3, 5 und 6 wurde bereits von einem sehr hohen Automatisierungsgrad gesprochen. In der Softwareentwicklung ist vor allem das Testen und die Strategie wichtig, da in der mittlerweile sehr ausgereiften Softwareentwicklung, die im Gesamtkonzept eine sehr hohe Komplexität mit sich bringt, das Testen immer zeitaufwändiger und schwieriger wird. Aus diesem Grund ist eine Automatisierung vor allem in diesem Bereich essenziell geworden, da wie in Interview mit Person 1 zu lesen ist, der zeitliche Aufwand allein für das Testen zu hoch werden würde, um einen zufriedenstellenden Prozess durchführen zu können. Wie in diesem Fall eine Teststrategie aussehen kann, ist in Interview 5 zu lesen. Hier spielt das Tool selbst ebenso eine sehr wichtige Rolle, da sowohl Azure DevOps als auch Jira für eine Testdokumentation und geeignete Features beinhaltet und ad hoc eine große Auswahl zur Nutzung bereitstellen können.

Außerdem wurde in den Interviews über Deploymentstrategien gesprochen, was im Release Prozess von Wichtigkeit ist. Das Ziel im Release Management beinhaltet eine neue Version

einer Software in der Produktivumgebung und bevor es zu dieser Ausrollung kommt, werden in diesem Prozess sogenannte Stages durchlaufen. Wie diese Stages in der Praxis aussehen und mit welchen Schritten und Zwischenergebnissen diese verbunden sind, zeigt Abbildung 7 aus einem Artikel über den Deploymentprozess im Release Management (Schwartz 2020). Für die Abbildung eines solchen Prozesses gibt es außerdem in den Azure DevOps und Bitbucket, das in Jira vor allem für Softwareentwicklung integrierbar ist, die Pipelines, mit denen Unternehmen, diese Prozesse abbilden können und zusätzlich neben Tests die Deployments automatisierbar macht.

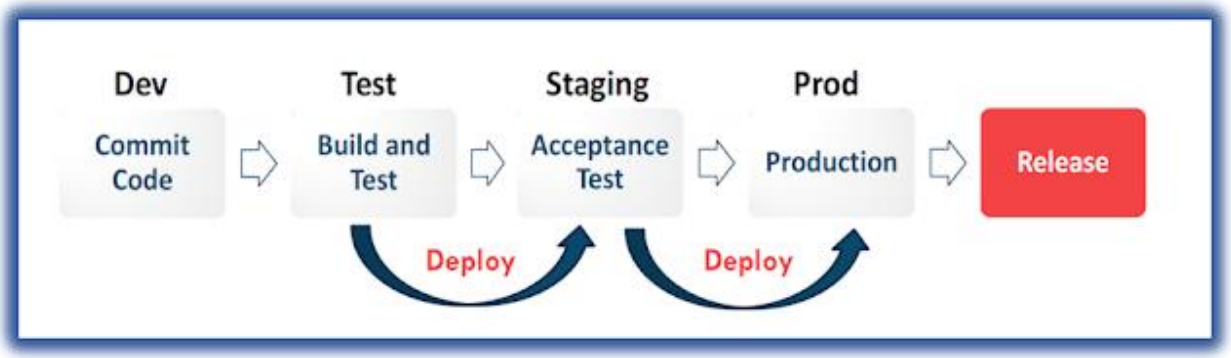

Abbildung 11: Stages Release Management (Lisa Schwartz 2020)

Zusammengefasst werden die Tools in den Unternehmen für ähnliche Teile im Prozess des Release Managements verwendet. Während vor allem Features wie das Issue-Tracking oder die SCRUM-Features für die Projektplanung und das Controlling große Vorteile bringt, spielen in der Entwicklung selbst Automatisierungsjobs, die über die Tools abgebildet werden können, eine große Rolle. Nichtsdestotrotz sieht die Nutzung im Detail in jedem Unternehmen unterschiedlich aus, jedoch werden die Grundfunktionalitäten der Tools in allen Unternehmen verwendet.

6.3 Handlungsempfehlungen definieren

Wie bereits erwähnt, stellt das Ziel dieser Arbeit einen Leitfaden, der für IT-Unternehmen, die in die Kategorie der Forschungsfrage fallen und zwischen 50 und 1000 Mitarbeiter haben, angewendet werden kann, um Probleme im Release Management zu lösen.

In erster Linie spielt in der Qualität des Release Managements der Prozess selbst eine große Rolle. Wenn Probleme am Tool oder aber auch externe Themen als Ursachen für ein qualitativ nicht hochwertiges Release Management auftreten, ist das ein klares Signal an das Team oder auch für die Prozessverantwortlichen, dass es Schritte in dem Prozess gibt, die überarbeitet werden müssen. In manchen Fällen ist hier auch eine Änderung der Entwicklungsstrategie eine Lösung, was zu einer organisatorischen Umstrukturierung des Teams führen kann. In vielen Unternehmen kam es zu solchen Umstrukturierungen, da aufgrund der Veränderung in der

Softwareentwicklung, die in der theoretischen Ausarbeitung erläutert wurden, eine agile Entwicklungsmethodik eingeführt wurde.

Im Zuge der Prozesserstellung spielen auch Teile der Entwicklungsstrategie eine große Rolle. Beispiele dafür sind die Deployment- und Teststrategie. In allen Interviews kam der Punkt rund um das Testen der Software auf. Vor allem in diesem Bereich rückte in den letzten Jahren die Automatisierung in den Vordergrund und entwickelte sich zu einer Grundlage, um in der Softwareentwicklung mit der Konkurrenz mithalten zu können. In diesem Sinne ist die Entwicklung einer Teststrategie notwendig, um in weiterer Folge das bestmögliche Ergebnis aus dem Prozess erhalten zu können.

Basierend auf das Produkt und auch der Unternehmensstrategie sowie der Vision muss ein Prozess entwickelt werden, der jedem Teammitglied und auch den Kunden bekannt ist. Die Deployment Pipeline stellt den Prozess und die Durchführung eines Releases dar. In diesem Prozess ist definiert, welche Bedingungen erfüllt sein müssen, um in die nächste Phase des Prozesses zu gelangen. Bereits bei dieser Definierung wird der Grad an Automatisierung berechnet. Aus der Visualisierung daraus wird erkennbar, welche Tasks automatisiert ausgeführt werden können, welche manuell passieren sowie welche Teile in Zukunft automatisiert werden müssen (Humble and Farley 2010).

Dass die Definierung einer Deployment Pipeline positive Auswirkungen auf die Qualität des Release Management hat, wurde in den Interviews bestätigt, indem in der Beschreibung des Prozesses eine detaillierte Auskunft über den Ablauf gegeben wurde. In diesem Bereich müssen Fragen wie *„Welche Systeme stehen zur Verfügung?“* und *„Wann wird von einem System auf das nächste deployed?“* beantwortet werden. Außerdem darf die Rolle des Kunden hier nicht vergessen werden. In vielen Fällen wird die Entscheidung von ihm getroffen, ob und vor allem was in die Produktionsumgebung deployed werden darf. Jedoch zeigt das Beispiel aus Interview 5 auch, dass bei einem kontinuierlichen Deployment, das zum Beispiel alle zehn Minuten eine neue Version ausrollt, zu einer Kundenzufriedenheit führen kann. Jedoch darf bezweifelt werden, dass diese Strategie in allen Unternehmen umsetzbar ist.

Nachdem die Deployment Pipeline definiert ist, kann eine Analyse der Automatisierung stattfinden. In diesem Sinne ist ein funktionierender Prozess die Voraussetzung für einen Einsatz der Automatisierung in den Teilbereichen. Die detaillierte Definierung und die Erfahrungen im Prozess helfen jedoch einen effizienten Einsatz von automatisierten Tasks einzuführen. Die Entscheidung, in welchen Teilen Automatisierung notwendig oder möglich ist, kann nur durch vorhandene Erfahrung getroffen werden. Automatisierung in der IT bringt jedoch enorme Vorteile mit sich, die langfristig die Qualität des Softwareproduktes erhöht (Sikender 2019b).

Ein weiterer Ansatzpunkt im Unternehmen ist die Adaptierung des Tools. Anstatt den Prozess in ein von Microsoft oder Atlassian vorgefertigtes Framework zu drängen, sollte das Tool für den definierten Ablaut verwendet werden. Einen nicht auf die Organisation ausgerichteten Prozess zu verfolgen, bringt in den meisten Fällen Stolpersteine im Release Management und diese können oft durch kleinere Anpassungen in den Tools oder selbst erstellte Add-Ons verhindert werden. Um Änderungen durchführen zu können, müssen einerseits Ressourcen in Form von

Personen und Zeit und andererseits das notwendige Know-how im Unternehmen sein. Diese Punkte müssen abgeklärt sein und basierend auf das Ergebnis dann entschieden werden, ob und in welchem Ausmaß das Tool angepasst oder erweitert werden muss.

Zu guter Letzt darf eine Analyse des Teams nicht fehlen. Wie in Interview 2 und 4 zu entnehmen ist, können auch interne Probleme im Team Auslöser für Probleme im Release Management sein. Aus diesem Grund muss eine Definition aller Rollen und vor allem Verantwortungsbereichen erstellt werden. In weiterer Folge können aus dokumentierten Problemen Zuständigkeiten evaluiert werden, sodass Lücken im Team identifiziert werden. Daraufhin kann in weiterer Folge analysiert und entschieden werden, ob eine Rolle wie im Beispiel aus Interview 4 gefehlt hat oder ob ein Zuständigkeitsbereich einer bestehenden Rolle erweitert werden muss.

In weiterer Folge soll das Ziel verfolgt werden, die Lücke zwischen der organisatorischen Einheit und dem Entwicklerteam so klein wie möglich zu halten. Die richtige Verwendung der Tools wie Azure DevOps oder Jira halten den Kommunikationsweg zwischen den beiden Komponenten so kurz wie möglich. Die Einführung des DevOps-Ansatzes, der diese Strategie verfolgt, erhöht die Effektivität und die Qualität in der Softwareentwicklung. Dies wurde bereits nachgewiesen und baut auch auf viele weitere Arbeiten auf (Sikender 2018).

Aus den oben genannten Punkten lässt sich folgender Leitfaden entwickeln, der für eine Überprüfung im Unternehmen dienen kann, um Probleme im Release Management identifizieren und in weiterer Folge beheben werden können.

6.3.1 **Prozess**

Wie bereits erwähnt spielt der Prozess und dessen Definierung eine große Rolle in der Qualität des Release Managements. Dieser muss visualisiert dargestellt sein und allen Beteiligten in der Entwicklung bekannt sein. Der Ablauf muss kontinuierlich überprüft werden, ob sich organisatorische oder technische Komponenten geändert haben. Demzufolge muss sich der Prozess über die Jahre verändern können.

6.3.2 **Rollendefinierung**

Aus der Definierung des Prozesses werden benötigte Rollen, die beteiligt sind, erkennbar. Basierend auf dem Ergebnis des vorigen Schrittes muss das Team analysiert werden. Alle Tasks im Prozess müssen einer Rolle zugeteilt werden. Dies kann zu einer personellen Änderung führen. In vielen Fällen ist eine Umschulung eine naheliegende Option, da bereits Erfahrung in dem Team vorhanden ist und aus finanzieller Perspektive eine vernünftige Option darstellt. Als Alternative dient eine Neubesetzung der Rolle.

6.3.3 **Entwicklungsstrategie**

Neben der Definierung der Rollen im Team muss auch die Entwicklungsstrategie festgelegt werden. Basierend auf den Prozess und dem Team muss eine Entscheidung getroffen werden, welche Methodik für die Entwicklung verwendet werden soll. Die agile Arbeitsweise hat sich in diesem Bereich am IT-Sektor etabliert. SCRUM und Kanban sind Methoden, um diesen Ansatz zu verfolgen. Ob diese Methodik für das Team die passende Lösung ist, hängt von mehreren Faktoren ab. Einerseits muss die Kundenperspektive berücksichtigt werden, da eine Änderung der Entwicklungsstrategie Auswirkungen auf die Kunden hat. Andererseits muss das Produkt kompatibel mit dem Prozess sein und iterative Weiterentwicklungen zulassen.

Für die Strategie spielt auch die Wahl der Entwicklungsumgebungen und der Infrastruktur eine Rolle. Wie in ITIL definiert können Änderungen über die Systeme Dev, Test und Staging auf die Produktivumgebung deployed werden. Wenn in der aktuellen Infrastruktur diese Umgebungen nicht vorhanden sind, muss diese erweitert werden, sodass zumindest eine Entwicklungs- und eine Staging-Umgebung vorhanden ist. Wie diese Systeme im Prozess verwendet werden, wird in der Deployment-Pipeline definiert.

6.3.4 **Deployment-Pipeline**

Eng im Zusammenhang mit der Entwicklungsstrategie steht die Deployment-Pipeline. Nachdem in der Entwicklungsstrategie die notwendigen Änderungen in der Arbeitsweise sowie in der Infrastruktur erfolgt sind, muss nun der Deploymentprozess definiert werden. In Abbildung 12 ist eine mögliche Deployment-Pipeline visualisiert, die zeigt, wann von welcher Entwicklungsphase auf die nächste Umgebung deployed wird und was die Voraussetzungen für die nächsten Schritte sind. In der Praxis spricht man in diesem Kontext von einer Definition of Done (DoD) und Definition of Ready (DoR), die im Deploymentprozess manifestiert sein müssen.

In der Deployment-Pipeline sind in weiterer Folge auch die Teststrategien verankert. Diese sollen beschreiben, auf welchem System wie getestet wird. Einerseits spielen hier Kategorien wie Unit- und Integration-Tests eine große Rolle. Andererseits muss definiert sein, welche Tests manuell und welche automatisiert ausgeführt werden.

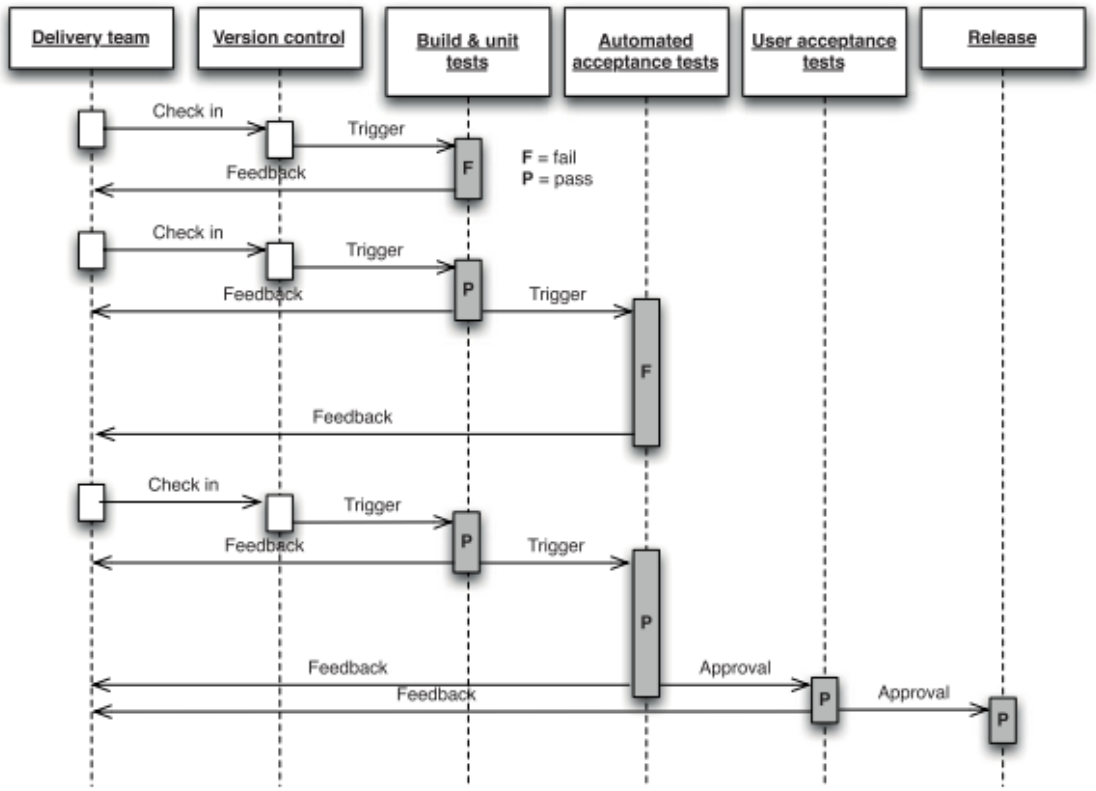

Abbildung 12: Deployment Pipeline (Humble and Farley 2010)

6.3.5 **Tool**

Zu guter Letzt muss der Prozess mit dem Tool in Zusammenhang gestellt werden. In vielen Unternehmen ist bereits ein Tool für das Release Management intgeriert und in Verwendung. Durch die bereits definierten Strategien und Prozesse soll erkennbar sein, in welcher Form die Tools unterstützen sollen. Wie bereits in den Ergebnissen dieser Arbeit zu entnehmen ist, spielt die Wahl des Tools keine große Rolle. Sofern in weiterer Folge Features fehlen, muss definiert werden, wie dieses Problem gelöst werden kann und beschlossen werden, ob eine Anpassung des Prozesses notwendig ist. Sofern jedoch Ressourcen vorhanden sind, sollte jedoch der Ansatz gewählt werden, dass das Tool eher angepasst werden soll als der Prozess.

7 NACHWORT

In dieser Arbeit wurde das Thema Release Management in mittelständischen IT-Unternehmen analysiert. Basierend auf einer Literaturrecherche, die den Prozess von der Planung bis zur Ausrollung einer Software unter die Lupe nimmt, konnten darauffolgend Tools für die Unterstützung im Release Management untersucht werden. Microsoft bietet mit Azure DevOps eine Software an, die in diesem Bereich unterstützend wirkt. Azure DevOps, das bereits ein Schlagwort aus der IT beinhaltet, verknüpft mit seiner Funktionspalette die Entwicklung und organisatorische Tätigkeiten (englisch: Development and Operational) und bietet daher in dieser Verbindung dem Unternehmen eine Plattform, auf der ein Release geplant und durchgeführt werden kann.

Auf der anderen Seite spielt auf dem IT-Markt die Atlassian Suite rund um Jira, Confluence und Bitbucket eine ähnlich große Rolle. Das australische Unternehmen Atlassian bietet mit ihren Produkten ähnlich wie Azure DevOps den IT-Unternehmen Plattformen, um den gesamten Softwarelebenszyklus zu managen. Mittlerweile haben sich beide Anbieter mit ihren Tools am Markt etabliert und finden sich im Release Management der meisten IT-Unternehmen wieder. Nichtsdestotrotz stoßen diese Unternehmen auf Probleme, die ein effizientes und erfolgreiches Release Management verhindern.

Arbeiten, die bereits diese Probleme untersuchen, wurden analysiert und anschließend in die Erstellung einer Interviewleitfadens integriert, womit in der qualitativen Forschung Erfahrungen im Bereich des Release Management gesammelt wurden. In weiterer Folge konnten die Ergebnisse der Forschung dokumentiert werden und in Handlungsempfehlungen umgeformt werden. Diese geben mittelständische IT-Unternehmen Themenbereiche vor, die im Zuge einer Analyse des Release Management berücksichtigt werden sollen. Zusätzlich konnten anhand der Ergebnisse aus der Forschung Hinweise und mögliche Lösungsvorschläge definiert werden, die die Probleme mindern oder vermeiden können.

8 WEITERFÜHRENDE FORSCHUNG

Wie bereits im Nachwort zu erkennen war, spielt in der Qualität des Release Managements der Prozess eine große Rolle. Durch die unterschiedlichen Produkte und Möglichkeiten in den Unternehmen lassen sich die Entwicklungsprozesse sehr schwer vergleichen. Während jedoch Unternehmen bereits ein hochqualitatives Release Management besitzen, der basierend auf den Prozess entwickelt wurde, müssen andere die Probleme im Prozess bereinigen. Die Probleme in der Prozessdefinierung konnten in dieser Arbeit nicht näher verfolgt werden, da es durch die unterschiedlichen Strategien und Produkten der Unternehmen nicht möglich ist, einen einheitlichen Prozess zu definieren.

Für zukünftige Forschungen kann das jedoch eine mögliche Untersuchung darstellen. Anhand einer Kategorisierung der Entwicklungsmethoden und der Kundenanforderungen können Prozessvorlagen oder ein Leitfaden für die Erstellung eines Prozesses erstellt werden. Mithilfe dieses Leitfadens ist es möglich, diese Arbeit einzubinden und einen Prozess inklusive der Nutzung eines entsprechenden Tools zur Unterstützung zu erstellen.

Ein weiterer Punkt, der aus dieser Forschungsarbeit nicht untersucht werden konnte, ist der Vergleich der Tools in der Atlassian Suite und Azure DevOps. Neben der Funktionalität, die in dieser Arbeit analysiert wurde, spielt auch die Einbindung in einem Unternehmen eine große Rolle. Vor allem aus wirtschaftlicher Perspektive muss eine Entscheidung auch im Hinblick auf die Kosten getroffen werden, was in dieser Arbeit nicht behandelt wurde. In weiterer Folge gibt es im Detail kleine Unterschiede in der Funktionalität dieser Tools und für die Unternehmen und vor allem für eine langfristige Strategie ist es notwendig, welches System als Unterstützung im Release Management am besten geeignet ist.

In Anbetracht dieser Punkte und im Hinblick auf die ständige Weiterentwicklung in der Methodik der Softwareentwicklung, die neue Ansätze hervorbringen, wird es in Zukunft wie im Release Prozess eine ständige Forschung geben müssen, die den aktuellen Stand der Softwareentwicklungsprozesse im Zusammenhang in das Release Management bringt.

ANHANG A - Transkript Interview 1

A: Als Einleitung würde ich dich Fragen eine kurze Beschreibung über dich, wie lange du in der Firma schon bist, welche Rolle du derzeit hast und Position. Also Position, Unternehmen und wie lange du schon bei der Firma bist.

B: Ich bin seit fast neun Jahren im Unternehmen und seit diesen neun Jahren arbeite ich am gleichen Großprojekt. Die ersten etwa sieben Jahre als Entwickler und seit etwas über einem Jahr als Scrum-Master und parallel in einer Devops-Rolle. Devops im Sinne von in den aller kleinsten Schritten. Wir haben diese Rolle eingeführt, weil wir das Projekt von Entwicklung über Test bis zu Betrieb fast allein machen, also als ziemlich atomares Team und meine Rolle als Devops-Engineer ist in diesem Fall über Logfiles, Bericht von Fehler und Aufarbeiten von wiederkehrenden Problemen, die über Tickets oder Kundenfeedback genannt werden.
Ansonsten als Scrum-Master den aktuellen Prozess begleiten, ein wenig Requierement Engineering dazu und versuchen, den Testprozess, den wir seit zwei Jahren auf neue Beine gestellt haben, kontinuierlich noch weiter voranzutreiben.Derzeit schon mit Devops-Unterstützung, das heißt Buildpipelines und automatischen Tests, denn es ist ein sehr großes Projekt und die manuellen Tests würden einfach viel zu lange dauern und über sehr viele Rechte und Rollen gehen, was einfach für den Testaufwand untragbar wäre und es an sich notwendig ist, dass eine Qualitätssicherung passieren muss.

A: Das Projekt an und für sich. Wie kann man sich das vorstellen? Wie groß ist jetzt das Entwicklungsteam? Und was in dem Fall relevant wäre, vor allem, wie das Team strukturiert ist

B: Wir sind 3 Backend-Entwickler und ein Frontend Entwickler. Ein sehr stabiles Team und der Prozess sieht vor, dass wir so bald unsere Features funktionieren, Teile davon über automatische Tests abbilden beziehungsweise ist es verpflichtend, dass manuelle Test, der sowohl von einem anderen Entwickler als jemand, der nicht sämtliche mitentwickelt hat, testet beziehungsweise die manuellen Tests werden auch von den Kunden ausgeführt.

A: Wir haben es schon rausgehört, ihr verwendet Azure Devops als Entwicklungstool als Task Tool. Mit Jira selbst habt jetzt noch nie was zu tun gehabt?

B: Jira ist für uns nur eine Datenquelle, weil unsere Endkunden mit Jira arbeiten und ihre Tickets tauchen bei uns eben in Open Issue Lists auf, sind also als Artefakte im Projekt vorhanden. Aber wir haben selbst mit Jira nichts zu tun.

A: Sprich also gibt es von deiner Seite keine Erfahrungen mit Jira, sodass man einen Vergleich ziehen könnte?

B: In dieser Firma nicht. Ich habe Jira schon vor fast einem Jahrzehnt gesehen, als ich in der vorigen Firma war und da habe ich es als Bug Tracking Software erlebt. Da ich in diesem Projekt aber nicht viel mitentwickelt habe, sind meine Erfahrungen eher beschränkt.

A: Azure Devops ist also vor deiner Zeit schon in der e1 eingeführt worden oder bist du dabei gewesen bei der Integrierung von dem neuen Tool?

B: Devops in unserem Projekt gibt es seit 2017. Ich habe vor kurzem einige Alte Sprints gesehen und da steht die Jahreszahlen dabei und wird seitdem immer weiter genutzt. Es sind zusätzliche Features dazugekommen und wir haben in diesen drei Jahren drei Scrum-Master gehabt, das heißt in unterschiedlichen Ausprägungen. Man kann DevOps als Taskliste benutzen, was funktioniert aber mit einem agilen Ansatz oder einer Projektleitung oder Koordination in dieser Größe ist es damit nicht getan. Wir haben gesehen, dass es nicht sehr zielführend ist, wenn man mehrere Tasklisten hat. Es gab einen Moment, wo es vier Listen gab mit Todos und kurz gesagt, diese Aufteilung hat nicht sehr gut funktioniert. Seit es Sprints gibt, in denen die Dinge, die passieren müssen, drinnen stehen, und Seitenthemen nicht nebenbei passieren, ist die Übersicht über die Dinge, die passieren sollen, deutlich einfacher. Wir haben zusätzlich noch ein Ticketing System. Das ist die zweite Liste von Dingen, um die wir uns kümmern müssen. Aber mit zwei Listen von Tasks kann man haben, mehr halte ich für sinnfrei.

A: Ja, das ist wahrscheinlich nicht von Anfang an so gewesen. Bei der Einführung bist du jetzt nicht direkt dabei gewesen, aber es hat sicher einen ersten Kontakt mit Azure Devops gegeben. Hast du schon im Vorhinein schon ungefähr gewusst, was es machen kann oder wie du das einsetzen könntest oder wie es benutzt wird? Oder war das einfach so ein Trial-and-Error Prinzip, klicken und schauen wie was funktioniert?

B: Der Erstkontakt war bei einer Abart von agile, die damals gemacht wurde und es war der Übergang zwischen Tasklisten-basierten Arbeiten und einer agileren Herangehensweise. Wir haben zu Beginn Schwierigkeiten gehabt, User Stories, Epics und Tasks so zu formulieren, dass sie transparent genug sind. Zu Beginn war DevOps auch rein intern bei uns, das heißt, unsere Kunden haben es nicht gesehen. Da es damals noch keine Dailys gab, war der Kommunikationsfluss für den Kunden, was aktuell passiert, wo es Probleme gibt, wo man vielleicht Unterstützung braucht, schwieriger. Das hat sich vor zwei Jahren geändert. Seit zwei Jahren gibt es Daily Standups. Der Project Owner ist dabei, hat den Überblick über die laufenden Tasks und weiß zumindest zweimal in der Woche ganz genau, was wir gerade tun, wo die Leute arbeiten und auch wo Bugs auftauchen. Also sowohl interne wie auch externe Bugs, die von Kundenseite aufgedeckt werden. Das geflügelte Wort bei uns ist: Es gibt keine Geheimnisse. Also es muss transparent sein, denn anders kann die Zusammenarbeit in diesen komplexen Themen eigentlich nicht mehr funktionieren.

A: Ja, absolut. Das heißt, im Laufe der Jahre hat sich einerseits der Entwicklungsprozess angepasst an den Prozess, wie man es verwenden kann. Andererseits habt ihr auch profitiert von der Funktionalität, was es anbietet. Ist es richtig zusammengefasst?

B: Das ist richtig zusammenfasst. Durch ein paar glückliche Fügungen haben wir auch Leute ins Team bekommen, die Automatisierung Erfahrung hatten. Und dadurch haben wir es auch geschafft, automatisierte Tests einzuführen. Wir sind auch immer dabei und werden vermutlich nie fertig sein.

Aber es macht mich stolz sagen zu können, dass wir inzwischen fast 300 automatische Tests haben, die bei einem Check-in der Branche laufen, aktuell nur 90 Prozent davon positiv. Aber es sind gerade sehr große Entwicklungen.

A: Heißt aber, wenn ich es richtig verstehe, dass es nicht eine spezielle Person für Azure Devops gibt, die Azure Devops analysiert und herausfindet, wie man es nutzen könnte. Eben vor allem die Pipelines, oder ist es eher so, dass die Erfahrung in das Team kommt und jeder der etwas dazu beitragen kann, dann den Beitrag hinzufügen kann und das dann anpassen kann an den Prozess. Also gibt es da auch keine spezielle Person für das Tool analysieren.

B: DevOps bietet unterschiedliche Herangehensweisen. Also in meiner Rolle als Scrum Master freue ich mich über die Taskbar und habe die auch geringfügig customized dass wir über Tags und Farben sehr einfach sehen, wo Dinge liegen. Wir haben einige Dashboards gebaut, um zu sagen, wie viele Leute wie viele Stories gleichzeitig offen haben. Wer sehr viele Stories in Verantwortung hat. Das macht uns die Planung und auch ein erstes Gefühl für Überlastung gibt es sehr schnell. Die Testcases, die ein weiterer Teil von DevOps Portfolio sind, werden vom gesamten Team genutzt und im zweiwöchigen StandUps auch präsentiert und neue Features, die wir dort finden und welche die uns nützlich sind werden weitergegeben. Die automatisierten Tests werden von den Kollegen, der früher als Testengineer gearbeitet hat, getrieben. Also wir sind ein diverses Team mit dem gemeinsamen Ziel und DevOps wird von jedem so gut genutzt für Scrum und gibt uns Wissen so weiter, weil es uns das Leben extrem einfacher macht.

Wir haben zum Beispiel auch in der Release Planung ein Custom Field eingeführt. Bei den Userstories sehen wir, ob sie deployed sind oder nicht. Somit tun wir uns im Vergleich zu früher, wo nicht ganz klar war, ist etwas schon fertig oder ist es schon getestet oder ist es schon deployed oder ist es überhaupt deploybar. Das ist heute relativ einfach. Es gibt ein Dashboard, da ist eine Liste von Userstories drinnen, die getestet und abgenommen sind. Die Kunden können in einem kurzen Meeting die Features festlegen, die mit dem nächsten Release deployed werden, sollen beziehungsweise können wir auch zwischen den StandUps, ob es ein kleines wöchentliches Release ist oder als Hotfix rausgehen soll. Also nachdem diese Daten sehr transparent sind, ist es sehr einfach solche Themen zu behandeln. Wir haben es jetzt aus Entwicklerteam viel einfacher als Excelliste sukzessiv zu pflegen oder die Arbeit einen Release Plan zu erstellen. Also einen Release Plan wurde jetzt letztes Jahr einmal gefordert, aber wir sind sehr schnell reingekommen, dass dadurch dass sich die Prioritäten auf der Kundenseite wieder verschieben eine Release Planung für uns also nur eingeschränkt nützlich ist, denn es gibt gewisse große Features, die in bestimmten Zeitpunkten angekündigt werden und diese halten auch meistens es sei denn irgendetwas anderes wird dringender, dann kommt dieses in dem Release oder ein Zwischenrelease passiert. Andererseits haben wir Features fertig seit einem dreiviertel Jahr, die wir nicht live nehmen können, weil die Prozesse im Business noch nicht so weit sind. Solche Dinge lassen sich relativ einfach abbilden und sind transparent und macht Scrum einfacher.

A: Heißt also auch, dass ihr keinen grundsätzlichen Release Prozess habt, sondern einen der eher

flexibel gestaltet ist, dass eben, wenn von dem Kunden die Priorität geändert, dass der Prozess zwar als Framework dargestellt wird aber nicht für jeden Release angewendet werden kann.

B: Ich behaupte wir halten uns an Scrum. Mit jeder Iteration gibt es ein Inkrement of potential shippable Software. Es gibt Iteration, wo es danach nichts gibt, dass wir deployen sollen. Es gibt Iteration wonach wir einen Release machen. Das ist abhängig davon was die Kunden brauchen. Wir wissen typischerweise einem Monat vorher, dass ein Release ansteht. Kleinere Themen können wir unter der Woche releasen.

A: Du hast es auch schon öfters angesprochen, durch Azure DevOps ist das Entwickeln und vor allem das ganze Projektmanagement und Release Management einfacher geworden. Kannst du das auch in Kennzahlen zeigen, sodass man sagt, dass die Dauer der Releases gesunken ist oder dass die Fehler, die durch Releases verursacht wurden, gesunken ist? Oder gibt es da Kennzahlen, die ihr mitbeachtet oder ist das Gefühl, dass es einfacher ist?

B: Es gibt keine klaren Kennzahlen. Es gibt Beobachtungen. Also wir haben viel in unserer Art des Arbeitens geändert. Es gab eine Zeit, in der wir sehr viele Schwierigkeiten hatten, in der Entwicklung, beim Testen und beim Go-Live. Es gab ein berüchtigtes Release mit 16 Subpräventionen, die alles Hotfixes waren. Wir sind nicht stolz drauf. Derzeit haben wir 3 bis 5 Releases unter dem Jahr und können theoretisch einmal die Woche ein Maintanance-Release rausgeben und ich muss kurz nachschauen. Typischerweise haben wir jetzt zu einem Major-Release, zum Beispiel 1.21, gab es 7 Subreleases, wo Tickets und kleinere User Stories oder Bugs deployed wurden. Und wenn ich mir jetzt das letzte Jahr anschaue. Wir haben jetzt weniger große Releases und sehr wenig Releases, wo wir viele Subpräventionen haben. Nachdem wir nicht zwischen Hotfix und Feature-Release unterscheiden kann ich es, ohne in die Release-Nodes und in das Deploymentpaket hineinzuschauen nicht aus dem Stehgreif beantworten. Aber ich habe eine Präsentation erstellt Ende letzten Jahres, wie sich die Ticketzahlen, Supportstunden und Releases in den letzten Jahren verändert haben, das kann ich dir zukommen lassen.

A: Danke, ist in dem Fall nicht notwendig, aber es ist in dem Fall wichtig, dass ihr KPIs verwendet, um zu sehen, dass sich der Prozess verbessert hat, mit der Hilfe von Azure DevOps.

B: Ich habe da sehr gerne die Anzahl der Tickets hinzugezogen. Erfahrungsgemäß haben wir die Kapazität etwa 50 Tickets pro Monat zu bearbeiten zusätzlich zu der Entwicklung und seit etwa einem Jahr haben wir pro Monat durchschnittlich 30 Tickets, die aufgehen und auch wieder zugehen. Das heißt also eine Anzahl von offenen Tickets jenseits 20 war vor einigen Jahren normal. Derzeit sind 10 offene Tickets ein „Irgendetwas läuft nicht optimal oder irgendjemand arbeitet nicht genug an Tickets.“

A: Noch einmal kurz zu der Verwendung von Azure DevOps aus deiner Perspektive. Kannst du ungefähr in Prozentzahlen oder in Stunden sagen, wie der Anteil der im Tool verbrachten Stunden im Verhältnis zu dem Entwicklungsstunden aussieht auf die Gesamtwochenstunden betrachtet? Also, dass du zum Beispiel 25% in Azure DevOps verbringst und der Rest für andere Sachen oder ist der Prozentsatz sogar höher?

B: Ich würde sagen Meetings ohne DevOps wäre derzeit nicht vorstellbar. Also nicht nur ein Daily

StandUp ist ohne ein DevOps, vor allem, da wir ja alle im Home-Office sind unvorstellbar. Auch ein Sprint Planning 1 oder 2 ohne vordefinierte Userstories wäre eigentlich absurd. Und ein einfacher Status über die Entwicklung mit „so stehen wir in diesem Sprint“ ist derzeit eigentlich mit einem Blick zu sehen, im Gegensatz zu man müsste mit 5 Leuten reden. Also die Zeiten, die wir in DevOps verbringen ist eigentlich Meetingzeit, weil es immer die Unterstützung des Meetings ist plus unsere Entwicklungszeit ist auch zum Teil in DevOps, weil die Userstories und die Akzeptanzkriterien dort drinnen sind plus die Testcases. Also offen ist es bei jedem Entwickler mindestens 50% der Zeit allerdings meistens nur zum Nachschauen. Bei Meetings natürlich, je nachdem in welcher Rolle die Teilnehmer sind. Wenn die Project OwnerIn ihre Userstories präsentiert und ihr Fragen stellen passiert sehr viel in DevOps. Wenn wir eine Userstory reviewen oder testen, dann in DevOps. Es würde also ohne gehen aber derzeit verteilt nicht.

A: Würdest du auch bestätigen können, dass auch abgesehen von Corona und der Home-Office Situation, dass das Tool über die Jahre auch immer mehr genutzt wird oder ist das eher gleichbleibend gewesen?

B: Es ist bei uns deutlich mehr genutzt worden. Wir sind deutlich agiler geworden. Also wir hatten früher Spezifikationsdokumente, die sehr wichtig waren und zeitweise als einzige Wahrheit betrachtet wurden. Während wir letzte Woche in eine kurze Diskussion gegangen sind, wie genau etwas spezifiziert wurde, bis ich es abgedreht habe mit „Es ist nicht wichtig, was in der Spezifikation steht, was ist das was du brauchst.“ Insofern, es macht das Leben einfacher. Aber das hat auch viel miteinander zu tun, wie wir jetzt arbeiten und wie wir jetzt mit den Kunden umgehen.

A: Zum Prozess noch einmal eine kurze Frage. Hat sich dieser über die Jahre der Nutzung von Azure DevOps weiterentwickelt und wurde stetig angepasst? Also vor allem als Azure DevOps eingeführt wurde, hattet ihr da bereits einen Prozess, wie ihr deployen wollt, wie ihr Features abarbeiten wollt und wie der Weg von einem Feature bis zu einem Produktivdeployment aussieht. Hat sich der mit Azure DevOps verändert oder hat sich Azure DevOps so anpassen lassen, dass der Prozess, so wie er vor 5 Jahren war zum Beispiel herzunehmen war?

B: Der Prozess ein Deploymentpacket zu schnüren, haben wir vor 8 Jahren gelernt. Also nachdem wir sehr datenbank-lastig entwickeln, gibt es da eine lange Liste von Objekten, die zu deployen sind. Zusätzlich die Tests für die Middleware plus ein Angular Frontend, die sind sehr einfach zu deployen. Unsere Deploymentstrategie sieht vor, dass vom Entwicklungssystem das Ganze auf unser Staging-System als erstes geht. Auf diesem System wird getestet, sowohl von einem anderen Entwickler als auch von den Kunden. Danach gibt es ein System, das heißt bei uns Train, ist allerdings ein Near-Live-System. Das heißt es entspricht exakt den letzten Deploymentstand des Livesystems und von dort geht es erst auf das Livesystem. Diesen Prozess gibt es schon länger als wir DevOps verwenden. Also den Deploymentprozess haben wir noch nicht automatisiert. Teilweise automatisiert auf den Dev- und Stagesystem für die Entwickler. Also über die Pipelines machen wir schon die Deployments. Auf den Livesystem nicht. Geändert hat sich eher das Organisatorische. Also die Transparenz, welche Pakete gerade in Entwicklung sind, wie die Features mit dem Aufwand

bewertet sind, um jetzt den klassischen Wasserfallmodell, das wir damals benutzt haben, die Abwandlung von agile, die wir seit zwei Jahren benutzt haben und jetzt Scrum kann man sagen. Wir haben viele andere Listen und Dokumente eliminiert. Wir haben viel Information in ein zentrales System gegeben, das uns das Arbeiten erleichtert und auch transparenter macht, womit uns viel Kommunikation und andere anfällige Prozesse abgelöst haben.

A: Zum Thema Automatisierung noch. Hast du da eine Kennzahl, wieviel Prozent des Prozesses jetzt schon automatisiert ist oder ist das pauschal so auch nicht zu sagen?

B: An Entwicklungsarbeit kann ich es nicht sagen. Ein Frontenddeployment auf die Stage ist ein Commit in den Integrationsbranch. Ein Middlewaredeployment ist ein Commit auf der Stage in den Integrationsbranch. Wir haben derzeit noch keine Strategie unsere Datenbankdeployments zu automatisieren. Das ist etwas, das wir schon lange auf unserer Wunschliste haben, aber uns immer andere Prioritäten dazwischengekommen sind. Natürlich wäre es praktisch ein Deployment, das zu schlechtesten Zeiten vier Leute für drei Stunden beschäftigt hat, was heute eigentlich von einem Entwickler und einem Kollegen, der als Copilot dabei ist und die Arbeitsschritte verifiziert und im Notfall auch deployen kann. Also der Aufwand ist sehr gesunken. Über die Erfahrungen in den Jahren ist ein Deployment jetzt normalerweise kein großer Aufwand mehr. Natürlich ist die Deploymentvorbereitung noch immer sehr viel Arbeit. Tests auf den Near-Live System sind noch immer Arbeit, aber im Vergleich zu früher ist ein Deployment jetzt mit sehr viel weniger Aufwand in der Vorbereitung und vor allem mit sehr viel weniger Problemen nach dem Go-Live verbunden.

A: Zum Schluss noch einmal eher etwas konkreter zu den Problemen. Gibt es bei euch bestimmte Themen, die man eigentlich auch in vielen Releaseprozessen und in vielen Unternehmen kennt. Gibt es da bei euch konkrete Fälle? Also ich habe schon herausgehört, dass mit Kunden die Abstimmung oder auch generell die Priorisierung der Kunden auch ein Problem darstellt, dass der eigene Prozess nicht immer so durchgezogen werden kann? Gibt es noch konkrete Probleme, die du noch ergänzen könntest? Auch jene, die vielleicht im ersten Moment nicht mit dem Tool lösbar wären aber vielleicht in weiterer Folge schon?

B: Also die Frage was das wichtigste Feature ist, ist ein Problem, dass wir es richtig an unsere Project Ownerin ausgesourced haben. Also normalerweise zu Sprintbeginn ist es sehr klar was gemacht werden muss und die Probleme mit, diese Probleme kommen zusätzlich dazu, haben wir inzwischen fast ausgeräumt. Inzwischen ist sehr klar, wenn ein zusätzliches Thema aufkommt, das wir anderes nicht machen können. Die Datenquellen sind manchmal noch ein Problem, wenn wir uns darauf verlassen, dass wir bestimmte Daten schon haben und das dann zu Beginn oder im schlimmsten Fall mitten in der Entwicklung aufkommt, dass dann die Daten nicht ganz das sind, was wir geglaubt haben, das kann manchmal passieren, aber in diesem Fall glaube ich nicht, dass man das einfach automatisieren kann. Was immer ein sehr großes Thema war, waren die Tests. Wir haben früher Testcases in unterschiedlicher Qualität gehabt. Testcases, die sporadisch dokumentiert waren. Dadurch, dass unser Entwicklungsprozess vorsieht, dass ein Entwickler, der dieses Feature nicht

entwickelt hat, den Testcase durchführt, bevor ihn die Kunden durchführen haben wir gelernt, unsere Testcases in einer sehr harmonisierten Art zu schreiben. Das heißt, niemand möchte von seinem Kollegen angerufen werden, weil er nicht weiß, wie er dieses Feature testen soll. Das ist zugegebenermaßen ein Lernprozess, aber sobald das einmal passiert ist und klar ist, dass das Schreiben eines Testcases durchaus einen oder drei Tage dauern kann, ist es auch in der Planung dementsprechend berücksichtigt. Das heißt wenn ich dann als Scrum-Master frage: Ist in dieser Schätzung das Testen schon drinnen, kann es sein, dass sich die Story-Points noch ändern. Das passiert uns immer wieder, dass wir zu solchen Mitteln greifen. Wenn wir etwa 60 Rollen im System haben, mit unterschiedlichen Berechtigungen können Berechtigungstest komplizierter werden. Wir haben in der letzten Woche etwa 80 automatisierte Tests für Berechtigungen geschrieben. Dadurch brauchen wir das Subset, das wir händisch hätten testen können nicht mehr testen, denn wir wissen, dass es so funktionieren wird, wie wir es automatisiert testen. Das ist sehr beruhigend.

A: Die Probleme die du jetzt aufgelistet hast. Werden die auch niedergeschrieben und gibt es dafür eine zuständige Person, die nach Lösungen sucht oder sind die Probleme jedem bewusst und es werden im Laufe der Zeit die Probleme angesprochen und versucht gelöst zu werden. Wie geht ihr da mit Problemen um, wenn ihr wisst, dass es diese gibt?

B: Typischerweise im Sprint-Retrospektivenmeeting sind wir da sehr ehrlich und typischerweise endet es damit, dass der Scrum-Master sich einen Task fasst, sich um etwas zu kümmern.

A: Also werden die Probleme konkret angegangen und direkt nach Lösungen gesucht?

B: Das reicht von „Gut dafür brauchen wir eine Userstory“ oder wir wissen, dass es irgendwo ein Problem gibt, dass wir eigentlich beheben sollten, aber machen es nicht, dann wird auch hier eine Userstory erstellt. Ja Entwickler sollten keine Userstories definieren, aber wenn sie eben wissen, dass etwas behoben werden kann, wenn es etwas gibt, das die Performance verbessert. Wenn es Prozessfragen gibt, zum Beispiel einen großen Begriff wie jetzt die Kunden anders arbeiten und wir Anpassungen machen und den Prozess nicht als Ganzes verstehen, kann es passieren, dass ich als Task ausfasse „Ein Meeting mit den Kunden organisieren, wo man uns diesen Prozess im Quellsystem erklärt.“ Das haben wir bis jetzt zweimal gemacht und die Kunden sind sehr freigiebig mit Information. Je besser wir gemeinsam das Produkt kennen, das wir entwickeln sollen, desto einfacher tun wir, das beste Produkt mit den Informationen, die wir zur Verfügung haben, zu erstellen. Dass es nicht immer das was bestellt ist, denn unsere zweite Aufgabe lautet: Die Features, die wir entwickeln sollen, möglichst gut und möglichst billig herzustellen. Das heißt gelegentlich schlagen wir Prozessänderungen vor. Es kann passieren, dass man via Userstories konzipieren und abschätzen und fragen, was passiert, wenn wir es anders machen? Es kann dann passieren, dass wir es anders machen. Wir sind alle über Jahre in diesem Projekt. Manche von uns über ein Jahrzehnt. Wir kennen den Prozess mittlerweile recht gut. Wir kennen die Kunden mittlerweile gut. Mehrere Augen sehen mehrere Möglichkeiten. Diese Änderungen der letzten Jahre waren sehr positiv. Nicht alles in DevOps ist technisch. All das was in Prozessen passieren kann ist eine Unterstützung.

A: Vielen Dank für das Interview.

ANHANG B - Transkript Interview 2

A: Am Anfang gleich mal eine kurze Vorstellung von deiner Person. Also Position in der Firma und wie lange du bereits in dem derzeitigen Unternehmen arbeitest.

B: Ich arbeite seit 2008 professionell als Softwareentwickler im Webbereich, seit 2019 bei der e1 Business Solutions tätig, eine Schwesterfirma der AVL List und habe hier die Rolle des Frontenddevelopers übernommen und mehr oder weniger den Frontend Testbereich, also automatisiertes Testen und auch jetzt in der Automatisierung der Releases.

A: Automatisierung ist hier eh schon ein gutes Stichwort. Welches Tool verwendet ihr für die automatisierten Tests hauptsächlich?

B: Also wir verwenden Azure DevOps. DevOps ist für uns eigentlich nur das Tool der Pipeline und der Userstories. Für das Release Management direkt müssen wir das noch manuell machen, aber die Pipeline wird darüber verwaltet, sowie die Tests und das Bauen der Apps.

A: Also wenn ihr etwas deployt, der Release Fortschritt wird also hauptsächlich im Azure DevOps Tool beziehungsweise in einem Board getrackt und verfolgt?

B: Nur bis zum fertigen Artefakt. Danach nicht mehr. Das ist dann über ein weiteres Portal, über das wir ein Package Request erstellen und das Deployment übernimmt dann ein anderes Team. Aber bis zu dem Punkt, wo das Artefakt fertig hochgeladen wird, wird über Azure DevOps gemanaged.

A: Das Entwicklungsteam, das an einem Artefakt arbeitet, wieviel Personen arbeiten dann an diesem Artefakt oder bist du da allein im Team?

B: Nein also allein bin ich nie. Es kommt immer mindestens ein zweiter Frontendentwickler dazu, der mich hier unterstützt, oftmals auch ein Backendentwickler und wenn es große Projekte sind, haben wir noch einen dritten Frontendentwickler dabei.

A: In Bezug auf das Tool, weißt du seit wann ihr Azure Devops für die Pipelines, die im Zuge der Entwicklung zum Testen verwendet, nutzt?

B: Wir verwenden diese noch gar nicht so lange. Seit 2020 hätte ich gesagt. Das hat das angefangen, dass wir das Tool für die Userstories verwendet haben und fürs Bug-Tracking und seit letzten Jahr August erst die Pipeline-Funktionen.

A: So gesehen der erste Kontakt mit dem Tool war für das Bug-Tracking und zum Tasklisten erstellen für die Entwicklung.

B: Der Erste Kontakt war selbst nur als GIT-Server. Erst später sind Bug-Tracking und Userstories hinzugekommen. Was ich noch vergessen habe zu sagen, es wird auch für manuelle Testpläne verwendet. Das haben wir schon vor der Pipeline genutzt.

A: Heißt also zusammengefasst, dass es 2020 den ersten Kontakt zum Tool gab und von vorherigen Unternehmen du auch noch keine Erfahrungen mit Azure Devops mitnehmen können.

B: Genau. Vorher habe ich alles was Automatisierung betrifft immer selbst gemacht. Da war kein Budget für solche Tools.

A: Nur zur Vollständigkeit die Frage bezüglich Jira. Kennst du Jira und hast du schon Kontakt zu Jira gehabt?

B: Ich kenne Jira, aber habe noch nicht wirklich damit gearbeitet. Ich bin mir nicht sicher, ob nicht Bitbucket ziemlich ähnlich ist, weil es ja glaub ich sogar vom gleichen Unternehmen ist.

A: Ja also Atlassian bietet Jira an, was mit Confluence und Bitbucket erweitert werden kann, sodass du grundsätzlich auf eine ähnliche Funktionalität wie Azure DevOps zurückgegriffen werden kann. Wenn ich das richtig verstanden habe, dadurch dass 2020 der erste Kontakt mit Azure DevOps entstanden ist, kannst du auch nicht etwas zur Entscheidungsfindung sagen, welches Tool für das Release Management genutzt wird.

B: Doch, also es ist eigentlich mehr oder weniger Vorgabe gewesen. Es hätte grundsätzlich die Möglichkeit gegeben, Jira zu verwenden aber nur in anderen Teams beziehungsweise in anderen Abteilungen. Das heißt für die IT war eigentlich DevOps vorgegeben. Das war der erste Punkt. Zusätzlich haben wir uns mit den anderen Teams abstimmen wollten und Vorträge gehalten haben, wodurch zu dem anderen Team, das DevOps verwaltet der Kontakt gestärkt worden und da ist dann ein gemeinsames Ziel daraus entstanden und das haben wir jetzt weitergeführt, indem wir ein Pipeline Framework erstellt haben. Das heißt also es war schon meine Entscheidung mit, aber mehr oder weniger auch Vorgabe.

A: Zum Thema Nutzung generell. Ein großes Stichwort war jetzt schon öfters die Pipeline. Weitere wichtige Punkte wie Issue Tracking was ich noch notiert habe, wird auch in Azure genutzt. Kannst du einen Prozentsatz sagen, wieviel ihr glaubt von der Funktionalität von Azure DevOps verwendet?

B: Das ist wirklich schwer zu sagen. Das Tool ist gefühlt wie Excel, ich glaube ich kenne nicht alle Funktionen davon, aber wenn ich eine Zahl sagen müsste, von den Sachen, von denen ich einen Überblick habe, dann wären das 30 bis 40% hätte ich gesagt.

A: Also Issue Tracking, Testing oder habe ich etwas vergessen?

B: Source Code Verwaltung, Testpläne, Build Pipeline, die Stories und Issue Tracking.

A: Die Artefakte liegen nicht am DevOps Server selbst?

B: Die liegen dort nur für die Build-Zeit plus 30 Tage am Server und werden direkt nach dem Build auf das Artifactory hochgeladen.

A: Nun zu den Kennzahlen. Wenn ihr eben ein Projekt entwickelt, habt ihr da Kennzahlen, die ihr verfolgt, wie zum Beispiel, wie lange das Release braucht, die Entwicklung selbst, wie lange der Deploymentprozess selbst braucht und wieviel Rollbacks es gegeben hat? Gibt es Zahlen, die ihr hierfür verwendet?

B: Kennzahlen haben wir nie aufgeschrieben. Es war bisher immer ein durcheinander, bis wir das eingeführt haben. Das Release Management ist jetzt das erste Mal in der ersten Iteration durchgeführt worden, ist also schwer da jetzt Zahlen zu erheben. Es ist auf jeden Fall das Release Management ein Prozess der anpassbar ist, weswegen wir versuchen es jedes Jahr etwas zu optimieren. Wir stehen jetzt kurz vor der zweiten Iteration. Der Plan ist mindestens ein Bugfix beziehungsweise ein Security-Patch rauszubringen pro Jahr. Das wird vorbereitet in Q2. In Q3 werden alle unsere Applikationen mit diesem Update migriert und sollten dann direkt ausgerollt werden. Wir haben dann noch in Q1 und Q4 Platz für Features für die Releases. So sieht der Plan aus.

A: Klingt jetzt so als habt ihr den Prozess sicher visualisiert dargestellt und dokumentiert? Also den

Prozess adaptiert ihr im Laufe der Zeit sodass dann irgendwann ein perfekter Release Prozess vorhanden ist oder wie kann ich eure Entwicklung verstehen?

B: Genau. Wir haben mit Flipcharts aufgeschrieben, wie wir uns das vorstellen für die erste Iteration. Das hat teilweise gut funktioniert. Mit der nächsten Iteration wird es dann eine Punktetabelle geben, die zu befüllen ist, die wir dann erweitern können. Aber grundsätzlich der Prozess ist jetzt nirgends visualisiert, sodass es ein Visio gibt. Es ist jetzt rein nur noch ein Scratch.

A: Ok aber das Ziel ist es dann auch den Prozess und ein Ergebnis zu erreichen, das dann auch auf mehreren Teams ausgerollt werden kann und dann auch als Vorlage für das ganze Unternehmen dienen kann?

B: Könnte für das gesamte Unternehmen funktionieren. Grundsätzlich ist es jetzt aber hauptsächlich für Frontendentwicklungen gedacht.

A: Welche Prozessteile im Release Management, abgesehen von Tests, schon automatisiert? Gibt es noch weitere Teile, die automatisiert sind? Auch wenn es nur Handover sind oder auch Pull Requests oder 4-Augen-Prinzip Code Review?

B: Ja also was ist jetzt alles automatisiert. Der Prozess sieht grundsätzlich so aus. Wir haben eine Planungsphase mit Features. Dann eben die Umsetzungszeit. Dann haben wir eine Testwoche. Da sind nur manuelle Tests für uns Entwickler. Darauf haben wir eine Bugfix Woche. Dann kommt das erste Deployment. Das sind hauptsächlich noch manuelle Steps. Dann gibt es eine Testwoche mit Testusern. Dann gibt es noch einmal eine Bugfix-Phase und schließlich dann das Live-Deployment. Das Live-Deployment ist mehr oder weniger semiautomatisch. Semiautomatisch, denn der Build wird automatisch getriggert und das Artefakt landet auf dem Artifactory und wir stellen dann manuell dann den Package Request für das andere Team und die übernehmen dann das Deployment. Was da jetzt dazwischen automatisch passiert ist eben der Build, die Unittests, die Integrationstests und die n-to-n Tests. Das Handover könnte irgendwann automatisiert werden aber ist jetzt aber noch nicht angedacht, denn es kann noch durchaus vorkommen, dass man nicht auf dem kompletten Userkreis ausrollen will. Das ist auf jeden Fall ein schwieriges Thema. Da haben wir als Entwickler leider nicht die Kontrolle darüber, das muss der Projektleiter dann bestimmen.

A: In Bezug auf die Verbindung auf den Prozess und das Tool. Wie geht ihr das an? Versuchts ihre einen Prozess zu finden, mit im Hinterkopf zu behalten was das Tool kann oder schaut ihr was das Tool kann und versucht einen Prozess zu finden, der mit dem Tool abgebildet kann?

B: Grundsätzlich hat die Entwicklung des Prozesses so stattgefunden, dass wir die repetitiven Arbeiten automatisieren wollen, die einmal wie es geschafft haben, umgesetzt zu haben. Wir haben uns einfach den Prozess auf das Tool, würde ich jetzt nicht sagen angepasst aber im Endeffekt wird schon der Weg sein, wie Azure DevOps das auch gedacht. Aber es halt einfach. Also war jetzt nicht viel großartig an unserer Arbeitsweise zu ändern, sodass es funktioniert.

A: Und generell von DevOps Funktionalitäten, die sie anbieten? Wie werden diese erforscht oder bringt hier jeder seinen eigenen Input mit oder wenn neue DevOps Funktionen veröffentlicht werden. Wie kommt ihr zu diesen Funktionen oder gibt es wirklich einen Beauftragten, der die Funktionalitäten analysiert?

B: Es gibt keinen der das macht. Bei uns ist es immer so, wenn wir glauben gewissen Arbeiten

benötigen kein manuelles Tun und Schauen, ob das irgendwie automatisiert werden kann und gehen dann auf die Suche ob und wie das in DevOps umgesetzt werden kann. Also in Grunde genommen ist es auch so die Herangehensweise, wie wir den gesamten Build-Prozess entwickelt haben. Wir haben eben genau den Fall gehabt, dass wir etwas umsetzen wollten und dann geschaut haben, wie wir das in Azure Devops umsetzten können und dann haben wir das so umgesetzt. Das haben wir dann also gemacht. Also da gibt es niemanden der da ständig schaut und versucht das auf Biegen und Brechen zu automatisieren, sondern immer dann, wenn der Bedarf da ist.

A: Und zu dem Prozess noch einmal, den ihr so entwickelts. Kannst du da einen groben Überblick geben, welche Rollen in den Prozess involviert sind oder ist das wirklich nur rein aus der Entwicklerperspektive angeschaut worden? Also gibt es einen Release Manager, gibt es eine Kundenperspektive in dem Prozess?

B: Es ist auf jeden Fall so, dass das eigentlich der Projektleiter steuere, wann was raus geht. Von den Entwicklern selbst kann jeder jede Rolle übernehmen. Meistens ist es dann so, dass das Release von mir dann freigegeben wird, also die letzten Merges und die letzten manuellen Tests und übergib das den Package Request, aber es ist dann noch der Teamleiter, der dann noch dabei ist, der als eine Art Moderation zwischen Projektleiter und Entwicklern ist. Der Kunde hat eigentlich nichts zu sagen, der hat eigentlich nichts zu sagen. Der hat eigentlich nur mit dem Projektleiter Kontakt und da geht es eigentlich nur darum, was gemacht wird und wann es fertig ist.

A: Aber die Akzeptanz vom Kunden muss dann eingeholt werden, bis dann das Artefakt deployed werden kann und auf das System ausgerollt werden kann.

B: Genau.

A: Kannst du da auch ungefähr sagen, wo ihr gerade stehts und vor allem welche Probleme immer neu auftreten bei dem Prozess? Oder ist es hier die Meinung, dass es schon gut ist, so wie es ist?

B: Grundsätzlich ist es schon ziemlich gut. Vor allem wenn man es mit vorher vergleicht, wie wir entwickelt haben und auch deployed haben. Wir sparen uns jetzt sicher nur an den manuellen Arbeiten drei Wochen, hätte ich gesagt und viel Frust vor allem.

A: Auf welche Dauer gesehen, also drei Woche in einem Release? Ein Release dauert wie lange dann?

B: Zwei Monate. Das ist jetzt eigentlich auch der Großteil. Jetzt haben wir eben Zeit für die wichtigen Sachen. Was im Prozess auf jeden Fall noch schiefläuft ist, dass wir zwar definiert haben, dass im Q3 ein Deployment zu erfolgen hat, beziehungsweise sollte durch den Prozess ein Release manuelle durchgeführt werden können, aber es zieht sich dann doch länger. Das hat aber mehr mit dem Projektleiter zu tun. Dass das Projekt oft zurückgestellt wird, jetzt vom Release her, dass das mit den manuellen Tests nicht gefunden wird. Das ist noch so eine Geschichte. Wir haben so eine Art agilen Ansatz, also teil agil, hätte ich fast gesagt, der aber nicht konsequent durchgezogen wird. Also es ist zu Teilen dann immer wieder mehr Wasserfall, dann wieder agil.

A: Findest du dann, dass eine Umstellung auf einen reinen agilen Ansatz der Lösungsweg wäre oder ist das überhaupt ein Ansatz, der verbessert werden kann?

B: Das ist auf jeden Fall eine gute Sache. Agil passt perfekt für das Release Management, wie wir gedacht haben, weil wir einfach viel öfters ein Artefakt rausbringen kann, was ja auch bei Scrum gefordert ist. Also es würde eigentlich Hand in Hand gehen.

A: Also die Probleme, die du jetzt aufgezählt hast, sind aber eben größtenteils Ressourcenprobleme kann man sagen also Mangel an Testusern und generell geringe Anzahl an Stunden?

B: Ja also es ist auch das Engagement des Projektleiters, das er das nicht forciert, dass die Sachen rausgehen. Da wäre ich mehr dahinter, dass wir schneller releasen und auch das Risiko eingehen. Das Thema Risiko hätte ich dazu genommen als Punkt im Prozess. Das Risiko ein fehlerhaftes Release rauszubringen, das traut sich keiner eingehen.

A: Also würdet ihr sagen, dass ihr zu genau testen wollt, bis endlich deployed wird?

B: Richtig. Man könnte direkt und schnell Bugfix Releases nachschießen. Die Leute sind da noch in dem alten Muster, dass ein Release, wenn es rausgebracht wird, drei Wochen manuelle Arbeit bedeutet.

A: Das Thema Continuous Integration und Continuous Deployment hat ja das Ziel, das es möglich ist, alle 10 Minuten oder jede Stunde ein Release zu deployen sozusagen.

B: Ja das wäre sogar schaffbar. Wir haben aber noch andere Probleme, die nicht nur den Prozess betreffen, sondern dann auch technische Probleme. Dass noch nicht vollautomatisier gebuildet werden kann für alle Plattformen. Wir haben da zum Beispiel mit MAC Probleme. Das ist mit Build-Skripte umgesetzt worden. Das lauft nicht über die Azure Pipeline und das ist auch Problem, dass gewisse Tools auf MAC Probleme verursachen, dass nach Updated nichts mehr geht oder ein Update gemacht werden muss, dass es wieder geht. Also das sind solche unvorhersehbaren technischen Schwierigkeiten, die dort entstehen.

A: Also beim Verlassen von Windows Betriebssystem Umgebung gibt es Probleme, wie das Tool und auch die Pipeline funktionieren? Also muss man das auf eine andere Art und Weise lösen?

B: Es hat auch vor allem mit den firmeninternen Policies zu tun. Zum Beispiel, dass der MAC noch nicht in das Netzwerk gehängt werden kann, in der die Azure Pipeline eingehängt ist. Das ist eben ein Punkt. Der andere Punkt ist mit diesen Tools, die wir verwenden selbst. Noch eine Sache, die ich dazu erwähnen möchte, ist folgendes. Wir haben zwar automatische Tests aber die Testabdeckung ist noch sehr gering. Durch die vielen manuellen Tests, die in jedem Release zwingend notwendig sind.

A: Werden diese Probleme, die du aufgehzählt hast, werden diese dann irgendwo niedergeschrieben und gibt es eine zuständige Person, die diese Probleme immer wieder aufgreift und im Team versucht zu lösen oder wie geht ihr mit diesen Problemen um?

B: Also die Prozessprobleme wie eben das Risiko und das lange Zurückhalten der Apps wird regelmäßig angesprochen, aber das scheint sich allerdings nichts akut zu bessern. Was die technischen Möglichkeiten betrifft, das versuch ich ständig etwas zu verbessern. Das ist so gesehen mittlerweile mein Steckenpferd, hätte schon fast gesagt mein Baby. Also jedes Mal, wenn ich Zeit habe etwas daran zu verbessern, dann nutz ich die natürlich.

A: Aber da gibt es keine offizielle Liste mit den Möglichkeiten, die ihr auftreten oder ausgebessert werden können?

B: Ja also offiziell ist keiner dafür zuständig, fühlt sich auch nicht so richtig jemand, die Sachen liegen auch nur in meinen internen Notizen, was verbessert werden kann. Aber es ist angedacht, bevor die

nächste Iteration startet, wie gesagt im Q2, stehen also kurz davor, dass wir danach ein Lessons Learned machen. Dann wird aus dem Lessons Learned auch offiziell eine Liste rauskommen, die wir dann abarbeiten müssen.

A: Findest du auch, dass Azure DevOps in Bezug auf die konkreten Probleme, die du aufgezählt hast, dass das Tool überhaupt als Lösung oder als Unterstützung dienen kann, die Probleme zu mindern oder sind das Probleme, die immer auftreten würden?

B: Das sind Probleme, die immer auftreten würden, unabhängig vom Tool. Wie gesagt Firmenpolicies, also es kann sein, dass es bei manchen Firmen gar nicht auftritt, weil die garkeinen Mac haben. Die holen sich in der Azure Cloud eine MAC Build-Machine.

A: Aber in Grunde genommen, unter den Punkt, dass ihr mit den Tests bereits einen guten Fortschritt habt, und mit der Automatisierung kann man schlussendlich sagen, dass das Tool als Unterstützung im Release Management dient?

B: Ja genau. Also es hat einiges erleichtert. Es ist der größte Vorteil alles in einer zentralen Stelle zu haben. Die Testpläne, die Pipeline, die Artefakte direkt nach dem Build und so weiter. Also es bringt schon sehr viele Vorteile mit sich. Ein Tool, das wieder auf mehrere einzelnen Tools aufgesplittet ist, wäre wieder ein Nachteil.

A: Wenn ich das richtig verstanden habe, versucht ihr auch nicht groß Azure DevOps so anzupassen, dass es immer für euern Prozess erweiterbar ist, sondern versucht mit so wenig Adaptierungen des Tools den Prozess zu gestalten.

B: Genau. Und wie gesagt, bis jetzt hat es immer sehr gut gepasst. Wir haben uns damit gespielt, bis es funktioniert hat, aber das mit dem Spiel bezieht sich nicht auf den Prozess, sondern mehr auf die Umsetzung, denn man hat schon eine steile Lernkurve dabei. Das sind da einige Notationen für die Pipelines und auch die Rechteverwaltung war nicht sehr einfach.

A: Letzte Frage noch, weil du gesagt hast ein zentraler Speicherort für alle Sachen, die im Release Management auftreten? Habt ihr auch Dokumentationen in Azure DevOps hinterlegt?

B: Ja also das sind mittlerweile aus. Wir haben unser Q2 Update und ein Migrationsheft, das in Mark down geschrieben ist und wird dort abgelegt. Und auch andere Sachen werden mittlerweile als PlantUML heißt das, so eine Markdown-Schrift, die dann einen Prozess visualisieren kann, dafür haben wir dann einen eigenen Container, der uns da einen Build rendern kann. Also das liegt dann jetzt alles im Repository drinnen und ich würde mir wünschen, dass die gesamte Dokumentation zum Repository reinbringen und nicht separat im OneNote niederschreiben.

A: Vielen Dank für das Interview.

ANHANG C - Transkript Interview 3

A: Danke für die Zeit und für das Interview zuerst. Ich würde dich kurz bitten deinen Lebenslauf zu erläutern. Wieviel Jahre arbeitest du schon in deinem Unternehmen, indem du jetzt bist? Deine generelle Berufserfahrung in den IT-Unternehmen und was derzeit deine Rolle ist.

B: Also Berufserfahrung habe ich seit 2011 also mittlerweile fast 11 Jahre, in denen ich eigentlich durchgängig in der IT-Branche arbeite. Begonnen habe ich als Systems Engineer, bin dann in das Consulting von Datenbank Management Systemen gewechselt, im speziellen Microsoft SQL-Server und habe dann in die Bankenbranche gewechselt und war dann in der IT-Organisation tätig und habe dann intern in weitere Folge gewechselt in die Konzernorganisation und seit ca. einem Jahr bin ich in einem sehr großen Softwareunternehmen tätig mit über 3000 Mitarbeitern und ca. 1000 Entwicklern und bin dort in der Rolle des Jira- und Confluence-Experten.

A: Also in dem Sinne verwendet ihr in euern Unternehmen Confluence und Jira oder Azure DevOps?

B: Jira und Confluence.

A: Gibt es dann eben, da du Microsoft Vergangenheit hast Erfahrungen mit Azure DevOps, sodass Vergleiche möglich wären grundsätzlich?

B: Mit Azure DevOps habe ich persönlich keine Erfahrungen.

A: Du hast angesprochen, dass ihr 1000 Entwickler im ganzen Konzern seid. Das wird wahrscheinlich in Teams aufgeteilt sein. Wie sieht dort ein Entwicklungsteam aus und wieviel Personen sind im Durchschnitt in einem Team?

B: Im Schnitt sind sieben Personen in einem Team. Diese Zahl ändert sich von Team zu Team, wobei ich glaube, sind es meistens zwischen fünf und sieben Personen.

A: Kannst du oder darfst du zum Entwicklungsprozess sagen, ob es ein agiler Ansatz ist oder ob ihr Scrum verwendet?

B: Ja es sind auf jeden Fall agile Ansätze, die verfolgt werden. Primär wird nach Scrum gearbeitet, wobei vereinzelt Teams auch nach Kanban arbeiten.

A: So gesehen verwendet ihr eben Jira und Confluence auch aktiv im täglichen Arbeiten.

B: Absolut. Das ist eines der zentralen Dinge in unserem Unternehmen mit expliziter Ownership für das Tool eben.

A: Weißt du auch wann Jira im Unternehmen eingeführt worden ist beziehungsweise warst du da schon im Unternehmen?

B: Ja. Ich persönlich habe Jira in den letzten beiden Unternehmen, in denen ich tätig war, mehr oder weniger eingeführt und ausgerollt.

A: Kannst du dich auch noch erinnern, wie dein erster Kontakt zu Jira war und wie du auf das Tool gestoßen bist?

B: Es ist unterschiedlich. Also im ersten Unternehmen, in dem ich tätig war, das war im IT-Systemhouse, da ist Jira aufgrund wahrscheinlich der IT-Affinität würde ich behaupten sehr gut angenommen worden, da war die Akzeptanz schnell da. In der Bankenbranche, wo ich auch gearbeitet habe, war Jira eigentlich gehostet, um unsere Rechenzentren zu betreiben und da haben

unsere Mitarbeiter mehr oder weniger geschult werden müssen. Da war die Akzeptanz mit Sicherheit niedriger und auch der Funktionalitätsumfang, den wir genutzt haben, wesentlich geringer. Das heißt der Einstieg war auch auf einem niedrigeren Level und je mehr man mit dem Tool vertraut war umso mehr wurde auch die Funktionalität am Tool dann auch genutzt. Es ist also im ersten Schritt wirklich nur Jira und mit Kanban die Arbeitsweise etwas nähergebracht worden und in weiterer Folge das Scrum Framework stark abgeändert und Confluence in einer nächsten Iteration oder Phase eingeführt worden.

A: Bei der Einführung hat es dann wirklich eine eigene Rolle gegeben, sodass eine zuständige Person verantwortlich war, dass das Tool ausgerollt wird im Unternehmen?

B: Exakt, also in beiden Unternehmen, wo wir es ausgerollt haben. In dem jetzigen Unternehmen war es schon existent.

A: Die Schulungen sind dann intern geführt worden oder ist dann von der zuständigen Person extern bei Atlassian eine Schulung gemacht worden? Oder wie ist das Wissen über die Funktionalität von Jira hergekommen?

B: Das Wissen habe ich mir damals selbst über Selbststudium, Recherche, Videos und Unterlagen selbst beigebracht und hab im ersten Unternehmen die Möglichkeit und auch die Kapazität bekommen, dass ich das selbst von der grünen Wiese auch starte, selbst hoste, hochziehe und entsprechend mich da selbst einlese und das weitertreibe, sowie das Wissen entsprechend weitervermittle.

A: Jira ist ja auch noch aufgesplittet auf Jira Software, Jira Service Desk und auch die Atlassian Suite bietet noch viel mehr als nur Jira und Confluence. Verwendet ihr dann auch Bitbucket oder andere weitere Tools?

B: Früher hat man auch Jira Service Desk verwendet, von dem hat man sich dann aber getrennt. Bitbucket ist derzeit auch in Verwendung für diverse individuale Entwicklungen, auch self-hosted. Neben Confluence ist das eben unsere Atlassian Suite.

A: Wie würdest du euern Entwicklungsprozess beschreiben? Wieviel Prozent habt ihr dort automatisiert oder seid ihr da noch nicht so weit?

B: Also der Automatisierungsgrad ist bei uns bereits sehr hoch.

A: Also habt ihr dann also auch einen fixen Prozess, der für alle Projekte angewendet werden kann und dort ändert sich zwischen den verschiedenen Kunden nichts oder habt ihr nur einen Kunden und Großteils interne Projekte?

B: Genau. Bei uns ist es so, dass wir allesamt an einem einzigen Produkt arbeiten. Das heißt wir individualisieren keine Lösungen und bauen auch keine Individuallösungen für Kunden, sondern wir entwickeln zentral ein Produkt weiter. Das ist groß würde ich sagen mit zig Modulen und über alle Module hinweg herrscht der gleiche Prozess.

A: Du als Experte kannst das vielleicht sagen. Gibt es einen ungefähren Prozentsatz, wieviel der Funktionalität von Jira, ihr nutzt?

B: Von der Funktionalität von Jira? Ich bin mir in der Frage nicht klar, welcher Prozess in Jira speziell? Von der Bearbeitung der Tickets oder in weiterer Folge schon in Richtung der Releases?

A: Genau, in Richtung Releases eher.

B: Hier würde ich sagen ist er schon sehr hoch, also bei 80% ca.

A: Kannst du da auch sagen, welche Teile dort bereits automatisiert sind?

B: Ein Teil davon ist das Kreieren von Release Nodes, die Darstellung dementsprechend von Release Nodes und die Auflistung.

A: Testen ist glaub ich ein großer Punkt, der immer erwähnt wird.

B: Ja automatisierte Tests sind auch bei uns ein absolut großer Punkt.

A: Verwendet ihr im Release Management dann auch Kennzahlen, die analysiert werden, wo man auch sagen könnte, ob sich das über die Jahre verbessert hat durch die Anwendung von Jira?

B: Ja verwenden wir. Ich weiß aber im Detail nicht welche und kann da auch keine genauen Zahlen dazu nennen.

A: Ok, aber es gibt Kennzahlen, die auch bestätigen würden, dass der Prozess durch die Anwendung verbessert wurde. Kannst du auch sagen wie viele Stunden du in der Woche im Tool verbringst oder auch wieder in Prozent angegeben, wie hoch dieser Wert ist?

B: Also ich persönlich ausgehend von einer 40-Stunden-Woche behaupte ich, dass es 35 Stunden sind zuzüglich aller Meetings ist mein gesamter Handlungsspielraum in der Automatisierung und Weiterentwicklung von Jira.

A: Würdest du auch sagen, dass ihr, da ihr ja auch Bitbucket zusätzlich verwendet, dass ihr noch andere Tools, die in Verbindung mit Jira, Confluence und Bitbucket verwendet werden müssen, sodass der Prozess auch wirklich so stattfinden kann, wie ihr möchtet?

B: Ja also es gibt Individuallösungen, die umgebaut worden sind.

A: Ihr habt also den Prozess in der Vergangenheit selbst einmal entwickelt, wie ihr diesen haben möchtet. Wie hat sich dieser über die letzten Jahre verändert mit den gestiegenen Erfahrungen von Jira oder ist er am Anfang mit Jira entwickelt worden und ist dann immer gleichgeblieben? Kann man sagen, dass sich der Prozess an den Funktionalitäten von Jira angepasst hat, oder ist das Tool so angepasst worden, dass der Prozess funktioniert?

B: Ja es war sogar eine sehr starke Veränderung da, weil eben das Unternehmen wahnsinnig stark gewachsen ist, dementsprechend ist auch der Prozess weiterentwickelt worden. Es ist auch laufend eine Weiterentwicklung. Die jetzige Version hat sich erst nach mehreren Iterationen finden müssen. Es ist vor allem auf Detailebene sehr viele dazugekommen und das Tool rundherum ist auch Stück für Stück weitergebaut und wird auch stetig weiterentwickelt. Es ist also schon sehr stark der Gedanke da, den Prozess nicht an das Tool anzupassen, sondern dementsprechend Ergänzungen für den Prozess entwickelt und auch ständig weiterentwickelt, sodass dieser den Prozess unterstützt.

A: Ist dieser dann auch so detailliert, dass man den Prozess auch für andere Unternehmen anwenden könnte?

B: Ich glaube er ist sehr individuell in dem Fall.

A: Jira und Confluence beziehungsweise Atlassian arbeitet auch ständig an den Tools selbst. Wie werden diese Änderungen im Prozess integriert und wie kommen solche funktionalen Änderungen in den Prozess rein? Gibt es hier noch eine zuständige Person, die Informationen aus erster Hand erhält und versucht, die neuen Funktionen in den Prozess zu integrieren?

B: Das heißt wirklich, wenn Jira als Tool eine neue Version ausrollt, inwieweit man diese neuen

Funktionen in den derzeitigen Prozess integriert? Dafür gibt es auch eine eigene Rolle beziehungsweise ein Core-Team oder auch Administrationsteam, dass diese Release Nodes in dem Fall von Atlassian bespricht und analysiert.

A: Wird das Tool dann wirklich von allen Rollen benutzt oder gibt es noch Entwickler, die mit Jira gar nicht arbeiten?

B: Also die komplette Research and Development ist in Jira abgebildet.

A: Zum Schluss noch eine grundsätzliche Frage? Hättest du noch Ideen oder Probleme, die in euern Release Management mit dem Tool oder auch gar nicht mit dem Tool zusammenhängen? Oder seid ihr der Meinung, dass ihr schon einen sehr gut ausgereiften Prozess habt?

B: Ich glaube im Moment haben wir einen sehr guten Prozess im Unternehmen. Wo ich ein Problem mit Jira im Release Management sehe, ist, dass Versionen eigentlich nur auf Projektebene verwaltet werden können. Das heißt es gibt keine projektübergreifenden Versionen in Jira. Das müssen wir beispielsweise mit einem zusätzlichen Plug-In, das wir dazu gebaut haben, syncen. Das gescheduled dafür sorgt, dass es überall dieselben Versionen gibt, wenn wir dann nur in einem einzigen Entwicklungsprojekt arbeiten, sondern dann auch verteilt wird auf zig Projekte und dann wird dort sichergestellt, dass zumindest namentlich die gleichen Versionen existieren, die dann in weiterer Folge als restricted Version verwendet wird. Das sehe ich als kleines Hindernis, dass diese nicht zentral an einer Stelle verwaltet werden können. Ansonsten dient es als Basis grundsätzlich sehr gut.

A: Also würdest du auch sagen, dass eine eigene Weiterentwicklung oder auch Supporttools sind absolut empfehlenswert, auch wenn der Prozess nicht 1:1 abbildbar ist.

B: Ab einer gewissen Unternehmensgröße, also ab mittelständigen Unternehmen definitiv ja.

A: Vielen Dank für das Interview.

ANHANG D - Transkript Interview 4

A: Zuerst würde ich dich bitten, mir eine kurze Vorstellung deiner Person, welche Position du im Unternehmen hast und auch generell, wie deine bisherige Berufserfahrung ausschaut.

B: Also ich habe in der AVL vor fast acht Jahren begonnen. Ich habe begonnen bei Sontorin/TFMS als Produktentwickler, wie es damals noch geheißen hat Integration Engineer. Heute sagt man eher DevOps Engineer dazu. Dort war ich für drei Jahre insgesamt tätig. Danach habe ich in das Toolservice gewechselt. Ich habe dann bei ITS Toolservice Azure DevOps komplett administriert und war betriebsverantwortlich für Jira, Confluence, Artifactory, Sonarcube und alles was so herum gebaut worden. Das habe ich drei Jahre gemacht und dann bin ich zu der IT gekommen, wo ich seit zwei Jahre ungefähr bin, und bin dort weiterhin verantwortlich für Azure DevOps global und für das Taskservice, in dem Fall Kubernetes as a Service, was man für Softwareentwicklung braucht, für containerisierte Applikationen und dort sind wir mittlerweile mit einem Viererteam.

A: Das heißt, wenn ich das richtig verstanden habe, hast du auch Erfahrungen mit Jira und Confluence, wodurch du Vergleiche ziehen könntest zwischen diesen Tools?

B: Ja.

A: Hast du hier schon vorab Präferenzen oder kurz gesagt, was findest du besser?

B: Also alles was Issue-Tracking, Release Management und Bug-Tracking generell ist Jira vom Tool her, schon wesentlich besser als Azure DevOps. Sourcecode natürlich nur Azure DevOps und für Sourcecode und CICD würde ich jetzt nicht auf die Idee kommen, Bitbucket oder Bamboo verwenden, was von Atlassian kommt. Auch deswegen geschuldet, dass auf der Issue-Tracking- und Release Management Seite Jira schon immer gibt bzw. sehr lange gibt in der AVL, eine riesige Community hat und manifestiert ist. Auf der anderen Seite Azure DevOps für Sourcecode und CICD schon seit über zehn Jahre manifestiert ist und auch dort schon eine Community vorhanden ist. Ich glaube auch, dass es für Issue-Tracking eine Gruppe gibt, die sich drum kümmert und sich damit beschäftigt, die sich dort auch gut auskennen und Azure DevOps Workitems, dort kocht ein jeder sein eigenes Süppchen. Hat also beides seine Vor- und Nachteile.

A: Kannst du dann auch sagen ungefähr sagen, wann und wie dein erster Kontakt zu den jeweiligen Tools war?

B: Das war von Anfang an weg da. Also es bei Sontorin/TFMS habe ich Bug-Tracking in Jira gemacht und wie mein erster Zugang dazu war, weiß ich jetzt nicht. Man hat es gezeigt bekommen und für mich ist es sehr benutzerfreundlich. Wenn man es einmal überblickt hat, ist es schon sehr benutzerfreundlich. Vor allem was der Vorteil bei Jira ist, dass wir teamübergreifend und standortübergreifend an mehreren Produkten gearbeitet haben, teilweise nicht zwingend Softwareprodukte und auch andere Projekte. Da lasst sich in Jira einfacher abbilden als in Azure DevOps.

A: Warst du dann auch bei der Entscheidung involviert, wie und wann Azure DevOps in der IT implementiert wurde, oder bist du zu der Abteilung gestoßen und da ist es schon fix in Benutzung gewesen?

B: Also als ich zur IT gekommen bin, habe ich den DevOps-Server der e1 übernommen und dann haben wir begonnen sukzessive als dann fix wurde, dass das Toolservice zur IT kommt, dass dann der Devops-Server der e1 mit dem DevOps-Server der AVL zusammengeführt wird, was wir jetzt heuer Anfang Jänner gemacht haben. Und das war jetzt auch meine Hauptaufgabe bei im letzten Jahr.

A: Bezüglich Release Management, bist du da in manchen Projekten involviert überhaupt oder bist du wirklich rein für das Tool selbst zuständig als Administrator?

B: Ich war damals bei Sontorin/TFMS als Integration Engineer war das damals auch meine Aufgabe, dass wir Releases rausbringen vom Entwicklungsteam. Jetzt bei uns im DevOps-Team haben wir so gesehen auch unser Produkt, das IT-Portal, wo wir auch so eine Art Release Management machen, aber da haben wir den wunderbaren großen Vorteil, dass wir das als Software-as-a-Service betreiben können, sodass wir jetzt keine Software an den Kunden ausliefern müssen, die bei ihm lauffähig sein muss.

A: Also habt ihr so gesehen eine freie Entwicklung und auch keinen Kundendruck?

B: Ja wir haben schon Kundendruck aber unsere Kunden sind In-House also unsere IT-Arbeiter sind unsere Kunden und deswegen muss das IT-Portal doch eigentlich immer gehen. Das muss auch gepatcht werden und es muss auch funktionieren also ein vereinfachtes Release Management machen wir ja trotzdem.

A: Wenn ihr dann eben so ein vereinfachtes Release Management durchführt, trackt ihr das dann auch anhand von Kennzahlen mit, sodass ihr messt, wie der Fortschritt dort ist?

B: Jein. Also wir im Team aber auch generell in der IT verwenden wir das Issue-Tracking in DevOps selbst. Als ich zur IT gekommen bin, da haben wir die Entscheidung getroffen, ob wir Jira verwenden zur Arbeitsplanung oder ob wir im DevOps Server gehen. Da ist jedoch entschieden worden, dass wir als IT-Team schon in DevOps reingehen sollen, weil alle IT-Mitarbeiter es verwenden. Das war schon eine Entscheidung damals und wir haben dort dann also unsere Epics, Features und User Stories. Die planen wir dort und sind auch verknüpft mit den Releases.

A: Verfolgt ihr da auch einen konkreten Prozess? Ist dieser dann auch visualisiert dargestellt? Oder wie sieht eure Arbeitsmethodik aus? Arbeitet ihr als Scrum-Team oder habt ihr einfach einen agilen Ansatz?

B: Wir arbeiten als Scrum Team. Wir haben vier Wochen Sprints und wir versuchen so gut wie möglich zu planen und arbeiten so unsere Arbeitspakete ab. Da viele von uns zusätzlich auch noch operativ Service betreiben ist dann oft eine Weiterentwicklung zweitrangig im Vergleich zu welchen operativen Tasks.

A: Würdest du dann zum Beispiel auch sagen, dass euer Release Management zusätzlich zu Azure DevOps noch weitere Software oder angepasste Tools notwendig sind, dass ihr euern Prozess verfolgen könnt, oder ist das alles schon ad-hoc verfügbar?

B: Also, wenn ich sage, ich bin als einfaches Entwicklungsteam mit einer Handvoll Entwicklern, die an einem Produkt arbeiten, dann ist Azure DevOps super. Du hast alles in einem Bus. Du hast deine Workitems, deinen Sourcecode und du kannst alles wunderbar miteinander verlinken. Nur sobald du anfängst mit mehreren Teams an mehreren Produkten und du möchtest ein Portfolio Management

haben, dann funktioniert das so wie wir es gerade verwenden nicht. Da hätten wir es von Anfang an anders machen müssen, damit es funktionieren würde.

A: Über die Jahre hinweg mit der Nutzung von Azure DevOps hat sich der Prozess auch weiterentwickelt und auch angepasst? Wie sieht dort der Entwicklungsprozess, wie von welcher Umgebung deployed wird oder generell, wie die Kundenakzeptanz eingeholt wird? Gibt es dort auch einen konkreten Prozess, der visualisiert wurde, wie ihr von Stage auf Produktivumgebung deployed?

B: Also niedergeschrieben beziehungsweise ein Dokument haben wir nicht. Wir haben es in unsere Automatismen manifestiert, dass man es so machen muss und es gar nicht anders geht.

A: Habt ihr da auch schon ein Großteil oder manche Teile komplett automatisiert?

B: Bei uns ist alles automatisiert. Wenn man das IT-Portal ansieht, dort gibt es keinen manuellen Angriff außer eben Sourcecode schreiben und Pull Request absegnen.

A: Also würdest du auch sagen, dass 100% des Prozesses automatisiert sind.

B: Wir haben immer die Grundbedingung: Man muss es löschen können und der Automatismus muss darüber laufen können.

A: Das ist dann schon sehr gut. Da ihr so gesehen in Experten seid in dem Tool und auch sehr viel damit arbeitet, wie kommen bei euch neuen Versionen in den Prozess? Azure DevOps bringt ja auch neue Features in neue Releases.

B: Wir updaten den DevOps-Server regelmäßig. Es kommen auch sehr selten Updates, also wirklich neue Features kommen ein- oder zweimal im Jahr. Das ist sehr schleppend aber manchmal kommen auch sehr viele auf einmal.

A: Aber da bist eben du die zuständige Person, die dann die Release Nodes von Azure DevOps kontrolliert und analysiert, dass es da diese neuen Funktionen gibt.

B: Genau ja.

A: Dann zu eurem Team noch einmal. Du sagtest, ihr seid vier Personen im Team.

B: Ja also wir sind vier Engineers und dann haben wir noch unseren Teamleiter und Enterprise Architekt, der uns die Vision vorgibt. Zusätzlich haben wir noch einen Scrum Master. Es sind jedoch nicht nur wir vier, sondern mittlerweile 15 Leute bei unserem Teamleiter und Scrum war ihm zu zeitaufwändig, da er die anderen Teams auch noch hat, weswegen noch eine Scrum Masterin hinzugefügt wurde.

A: Azure Devops wird aber trotzdem von allen Rollen im Prozess genutzt.

B: Ja genau. Also wenn es um Controlling geht oder um Fortschritt geht, dann ist alles in DevOps die Wahrheit. Da ist es sogar so, dass wir „Prügel“ bekommen, wenn wir sagen, dass wir dies und jenes zusätzlich gemacht haben, aber noch nicht eingetragen haben.

A: Könntest du auch Probleme identifizieren in DevOps, die konkret am Tool liegen? Also hättet ihr da Verbesserungsvorschläge oder seid ihr da zufrieden, so wie es ist?

B: Wir gehen eher den Weg, dass wir uns an das Tool anpassen als wir passen das Tool an, weil wir vier Personen sind. Also in DevOps gibt es Epics, Features, User Stories und Tasks und eben die Bugs, das gibt es als Default immer und wir haben unsere Arbeit so festgelegt, dass wir damit auskommen. Vor allem auch weil wir glauben, dass es für uns weniger Aufwand ist, uns an das Tool

anzupassen, anstatt das Tool auf uns anzupassen, wobei wir damals nicht wirklich welche gehabt haben. Auch wenn man so viel anpasst, muss man bei jedem Update kontrollieren, ob es danach auch noch funktioniert. Und diese Arbeit haben wir gesagt, tun wir uns nicht und lassen das Tool so nahe am Standard wie möglich. Also bezüglich Features, die uns fehlen würden, habe ich keine Informationen dazu.

A: Und habt ihr dann überhaupt Probleme in euern Release Prozess oder habt ihr so gesehen Stolperfallen, über die ihr immer wieder fliegt? Oder seid ihr mit euern Release Prozess grundsätzlich schon so zufrieden, dass dieser so stehen bleiben kann und da gibt es eigentlich nie Probleme.

B: Es ist so, dass wir vier Personen sind und wir alle kennen uns gut aus. Wir haben jetzt nicht den Bedarf, dass wir jetzt sagen, bestimmte Aktionen dürfen nur von bestimmten Personen ausgeführt werden, weil die nicht wissen, was sie tun. Was man dann schon sagen muss, dass wenn ich an meine Produktentwicklungszeit zurückdenke, da 20 Personen in Graz, 40 Personen in Wien, da haben 60 – 70 Leute an einem Projekt gearbeitet und dort muss man schon auf vieles aufpassen. Ein Beispiel ist das Berechtigungskonzept. Nicht jeder Tester darf einen Bug anlegen, denn in 90% der Fälle ist der Bug sinnlos, weil er etwas vergessen hat. Da muss es jemanden geben, der noch einmal drüber schauen muss. Wir haben den Luxus, dass wir so klein sind und uns auch gut auskennen, sodass wir wissen was wir tun. Deswegen haben wir bewusst alles am Standard gelassen, dass wir uns keine Stolpersteine einfangen können und eben den Automatismus so weit entwickelt, dass das immer funktioniert. Wenn wir wollen, müssen wir selbst uns einen Stein legen. Zum Beispiel dass man in den Master-Branch, so nicht einfach commiten kann, weil das sonst später zu Problemen führt oder man muss eine User Story beim Pull Request angeben, damit man weiß, welchen Ursprung diese Codezeile hat und welches Feature oder User Story dahinter liegt beziehungsweise, was ist das überhaupt? Solche Sachen haben wir durch den Automatismus erzwungen.

A: Könntest du auch im Namen von deinem Scrum Master oder Projektcontrolling reden, dass Azure DevOps wirklich hilfreich ist, da sie mit einem Blick wissen, wie der derzeitige Stand in der Entwicklung ist?

B: Ja. Wenn du das so ausdrückst, dass sie weiß, was wir tun ist es für sie sehr hilfreich. Einerseits sollen wir das machen und andererseits müssen wir das auch machen. Eben um auch den Management Sachen rechtfertigen zu können. Warum haben wir länger gebraucht für eine User Story oder warum haben wir eine User Story nicht fertig geschafft, weil wir auf der anderen Seite so viel ungeplante Tasks gehabt haben.

A: Das Thema Transparenz ist hier eben ein großer Begriff. Zum Schluss noch. Könntest du auch ungefähr eine Zahl sagen, wie viele Stunden du pro Woche etwa in dem Tool verbringst oder ist das so jetzt ad-hoc schwer zu sagen und es variiert sehr stark?

B: Nur was Issue-Tracking und Planung so in etwa angeht?

A: Genau ja. Sourcecode ausgenommen.

B: Also wir haben jeden Tag 15 Minuten Stand-Up. Dann haben wir einmal in vier Wochen ein Tag Sprint Planung. Das wären dann nur in der Woche im Tool fünf Stunden. Also schon einen halben

Tag. Wenn wir die gesamte Planung mit einberechnen, findet doch sehr viel im Tool statt. Sprint Start, Sprint Ende, Sprint Retro, Sprint Review machen wir alles dort drinnen. Das ist auf den Monat gesehen schon viel Zeit. Das ist dann auf die Woche heruntergebrochen schon 5 Stunden. Also ich weiß in Jira habe ich sicher mehr Zeit drin verbracht.

A: Könntest du da auch Gründe sagen dafür?

B: Weil über Jira haben wir die ganzen Bugs abgehandelt. Wenn wir jetzt sagen, dass wir ein Release an das Testteam übergeben haben und das Testteam hat dann Bugs gefunden. Dann ist es immer wieder zu uns zurückgekommen, wir haben uns die Kommentare durchgelesen, was erweitern und im weiteren Schritt an das Testteam wieder übergeben, da kommt dann wieder etwas zurück oder auch wenn von einem anderen Produkt für Kundenservice Fehler auftreten, kommt das zu uns rein. Dann wird das alles in Jira abgehandelt. Da hätte ich gesagt, da sind dann am Tag mindestens 2 Stunden, die man als Integration Engineer in Jira verbringt.

A: Würdest du auch sagen, dass in den letzten Jahren der stündliche Aufwand in DevOps mehr geworden ist oder sogar weniger?

B: Also bei mir im konkreten Fall im Vergleich zum Toolservice Team und jetzt IT ist es sicher mehr geworden, weil wir im Toolservice Team praktisch keine Planung gemacht und im DevOps Team machen wir jetzt die Planung selbst in DevOps, also ist es hier schon definitiv mehr geworden. Aber es waren auch andere Arbeitsweisen. Auch kurzfristig gesehen ist es auch seitdem ich in der IT bin mehr geworden, weil wir hier immer mehr Fokus auf eine saubere und vernünftige Planung legen.

A: Würdest du auch sagen, dass es dann auch mit der Zeit immer besser geworden ist oder ist es schon seit einem Jahr auf dem jetzigen Stand?

B: Ja also wir haben schlecht angefangen, dann ist es super geworden und dann mit Corona haben wir es schleifen lassen. Dann ist die Verwendung nach unten gegangen. Weil wir uns nicht mehr getroffen haben und wir nicht mehr physisch darüber gesprochen haben, ist es nach unten gesackt und es war nicht mehr gut. Da haben wir gesagt wir wollen wieder den alten Prozess und wir hätten gerne jemanden, der sich drum kümmert und daraufhin ist ein Scrum Master gekommen. Es ist dann wieder besser geworden, zwar nicht von einem Tag auf den anderen und es hat auch eine Zeit gebraucht, bis wir dann das alte Planungschaos losgeworden sind und die Altlasten entfernt worden sind und seitdem ist es gleichbleibend gut.

A: Also sind die Probleme eher prozesstechnischer Natur und nicht vom Tool selbst?

B: Genau, einfach weil wir es für uns selbst nicht dringend gebraucht haben, aber ist die Verwendung wieder nach unten gegangen und haben das nie wo beschreiben können. Jetzt da wir auch jemanden haben, der täglich oder wöchentlich darauf schaut und den Status der Projekte analysiert und erkennt, wieso dort nichts weitergeht oder nichts gemacht wird oder auch Tasks sind, die eine Woche den Status „In Work“ haben, dann wird dort nachgefragt was dort los ist, woraufhin man erkennen kann, dass der Task vielleicht falsch geschrieben wurde.

A: Vielen Dank für das Interview.

ANHANG E - Transkript Interview 5

A: Zu Beginn kurze Vorstellung von deiner Seite; Position, welchen Beruf du hast, wie dein beruflicher Werdegang aussieht oder ausgehen hat bis jetzt.

B: Ja, alles klar. Ich habe Informatik studiert an der TU Wien und habe während des Studiums bald zu arbeiten angefangen für ein Start-Up in Wien. Start-up wurde aufgekauft von größerer Firma. Als ich mit Studium fertig war, habe ich mich gemeinsam mit einem Arbeitskollegen selbstständig gemacht und haben zu fünft insgesamt ein Add-On für Jira entwickelt. Das hieß Roadmaps für Jira. Das war damals noch ein reines Server Add-On. Jira hat Jira Cloud, Jira Server, Jira Data Center und Data Center hat es damals noch nicht gegeben, Jira Cloud war gerade im Kommen. Also es war ein Add-On für Server. Und das Ziel vom Add-On war es die Planung über einen längeren Zeitraum zu ermöglichen. Jira Software war immer sehr gut darin, diese kürzeren Iterationen zu planen, was ist im nächsten Sprint, was ist in den nächsten zwei bis drei Sprints.

Wenn man eine Roadmap für die nächsten Monate oder Jahre, gabs noch kein passendes Werkzeug und wir haben versucht sowas zu bauen und sind dann etwa nach einem Jahr von Atlassian aufgekauft worden. Sie kamen auf uns zu, wollten eine Lösung in dem Bereich und sind relativ rasch einig worden, alle zusammen nach Australien ausgewandert und haben für Atlassian gearbeitet. Meine Rolle war die eines Engineering Managers, also in der Engineering Organisation in einer Management Rolle und habe zunächst am Add-On weitergearbeitet. Zunächst wurde es regebrandet auf Portfolio von Jira und mittlerweile heißts tatsächlich wieder Advanced Roadmaps. Es gibt es also immer noch, schaut inzwischen, nach einigen Jahren, nur etwas anders aus. Und habe nach einiger Zeit vom Add on ins eigentliche Jira Team gewechselt.

Es gibt verschiedene Jira Produkte, die Jira Software, die Jira Service Management und Jira Business und die kann man alle getrennt verwenden und nutzen, jedoch basieren alle auf einen großen gemeinsamen Teil. Organisatorisch ist das so abgedeckt, dass es für jeden Teil eine Organisation gibt. Also es gibt eine Jira Software Org., eine Jira Service Desk Org. und eine Jira Business Org. und es gibt zusätzlich eine Organisation, die sich um den gemeinsamen Layer kümmert – die Jira Plattform. Ich bin dann zum Jira Plattform Team gewechselt und war größtenteils verantwortlich für – wie wir es nannten – das neue Jira Frontend, Jira UI.

Hintergrund ist der, dass vor vier Jahren ca. ein großes Re-do von der Jira Oberfläche begonnen hat. Sowohl designtechnisch als auch technologisch. Es ist immer noch am Laufen von Atlassian. Prinzipiell kannst du dir das so vorstellen, dass das Frontend, das Java Script, HTML, CSS, früher tief im Jira integriert war. Bisschen so wie man vor 10-15 Jahren Applikationen geschrieben hat. Der Request kommt zur App, die App generiert serverseitig HTML und retourniert das HTML. Heutige, moderne Apps funktionieren ganz anders. Hier gibt es eine viel klarere Trennung zwischen den teil, der auf deinem Handy rennt und dem Teil, der auf tatsächlich auf dem Server rennt. Mit einer viel definierten Schnittstellen. Es gibt Rough-QL-Abfragen oder wenn man keine Rough-QL-API hat, hat man REST abfragen. Man hat irgendeinen Endpunkt und das Frontend kommuniziert mit Endpunkt und es gehen eigentlich nur daten hin und her und das Frontend ist eine relativ eigenständige

Applikation. Das habe ich einige Jahre gemacht, war durchaus spannend und fordernd.
Vor ca. 2 Jahren sind ein paar Sachen zusammengefallen. Einerseits Corona, gleichzeitig hat es mir nicht mehr so Spaß gemacht, weil ich glaube, dass ich lange genug dabei gewesen bin und habe, beschlossen mich wieder selbstständig zu machen und wieder ein Add-On für Jira zu bauen.
Das ist das JXL. Und ja, damit sind wir beschäftigt.

A: Ok, sehr interessant schon mal. Sprich, du hast in dem Fall direkt Einblick in die Funktionalität von Jira gehabt. Habt ihr Jira selbst genutzt um, auch wenn es nur kurzfristige Planung ist, Releases wirklich selbst genutzt oder kann man sich das so vorstellen, dass auf andere Ressourcen zurückgegriffen wird?

B: Also eigentlich Jira definitiv. Wenn man Jira verwendet beziehungsweise vor allem Jira Cloud ist sehr viel von dem was man sieht, haben wir über die letzten 2-3 Jahre gebaut. Das war schon sehr umfangreich das ganze Projekt. Wir haben Jira definitiv verwendet, absolut. Es ist so, dass innerhalb von Atlassian verwendet jeder die Attlassian Tools. Confluence, Attlassian selbst betreibt die größte Confluence Instanz, die es überhaupt gibt. Die wird als Intranet Wiki genutzt. Es gibt zwei oder drei große Jira Instanzen, die gehören auch zu den größten überhaupt. Natürlich auch Bitbucket und Hipchat wenn es des noch geben hat, beziehungsweise der Nachfolger Stride. Trello wird natürlich auch verwendet. Zum Thema Release Management würde ich sagen, dass bei Atlassian und ich glaub beim Web Engineering grundsätzlich man versucht wegzugehen von den Heavyweight Release Prozess. Früher war das ja so, Jira Data Center ist es immer noch so – es gibt alle paar Wochen ein release und der Release hat eine Versionsnummer, beinhaltet Features, Major Release, Minor Release, Batch Release, dann kommt das ganze Thema mit semantic Versioning dazu. Wie welche Versions-Änderung, welche Release muss ich machen, dass ich meine APIs brechen darf. Wenn ich meine APIs brechen darf, muss ich ein Major Release machen und gleichzeitig will ich natürlich, nicht die Kunden überfordern damit, dass ich regelmäßig gern die APIs breche. Vor allem bei einem Tool wie Attlassian, das sehr viele externe Applikationen hat, ist es natürlich immer ein großes Drama, wenn man Apis bricht, im Web Engineering. Im Cloud Engineering versucht man weg davon zu gehen.

A: Also eher Richtung agiles arbeiten; flexiblere Prozessgestaltung wahrscheinlich?

B: Ich glaub es hat gar nicht so viel mit dem Entwicklungsprozess zu tun. Ich glaub der Entwicklungsprozess ist mit der ganzen Planung sehr schwer. Ich kann mir nicht mehr vorstellen, wie man nicht agile Software entwickeln kann. Ich weiß früher hat man das gemacht, ja. Für mich ist das heutzutage unvorstellbar. Mein gesamtes professionale Leben hat in Firmen stattgefunden wo Scrum impliziert wird oder von mir aus Kanban. Und dieses Iterative, Zyklische das gilt sowohl – das hat mit den Releases nicht so viel zu tun find ich. Du kannst sagen du hast einen sehr definierten Release Prozess und dann agil machen oder du hast zurück zu dem wie sie in Jira Cloud oder Attlassian Cloud funktioniert, dass man einfach versucht, dauernd zu releasen. Also wir waren sofort an einem Punkt, wo wir jeden Merge zum Master released haben. Und das ist ja der Idealzustand. Denn die Philosophie dahinter ist in Wirklichkeit ja die, wenn du sehr häufig sehr kleine Sachen released, sehr kleine Changes, dann ist das Risiko mit jedem einzelnen Change viel kleiner, weil ja nicht besonders viel Änderungen reinkommen. Du sagst jetzt ok, wenn du einen Fehler released,

kannst das sehr leicht einfach zurückholen auf die vorige Version und den Fehler fixen. Diese Idee mit dem roll back rather than roll forward. Roll forward heißt du musst einen Fix releasen und eine neue Version releasen. Da die ist die Philosophie eher, dass man zurück rollt auf die vorige Version. Problem gelöst, unmittelbar für den Kunden, und dann kann man sich in aller Ruhe um den Fix kümmern. Für den Kunden ist das Problem dann gelöst. Und es ist jetzt so, dass es für Jira Cloud oder andere Cloud Produkte gibt es das nicht wirklich. Die haben nicht eine Version. Intern haben die keine Version 11.15.3. Das hat irgendeinen Hash, einen Hash von git Commit, der gerade in Production ist. Und 10 Minuten später ist dann vielleicht ein neuer Hash in Production. Und damit fällt dieser Release Planning Aspekt zu einem gewissen Teil weg, wo man sagt „jetzt muss ich mir überlegen, nächste Woche mach ich ein Release mit 10 Features, wird des rechtzeitig fertig, und so weiter“. Das wird weniger relevant, weil Features laufend released werden. Whenever it is ready, sollte man es releasen.

So jetzt mal die Idealvorstellung. Und ich glaube Firmen wie Atlassian oder Facebook, diese high performant, sehr erfolgreichen Engineering Organisationen sind da schon relativ nah dabei. Aber es ist in Wirklichkeit so, dass es immer einiges komplexer ist als das. Zum Beispiel ein Aspekt den Atlassian ganz intensiv praktiziert ist, dass man durch verschiedene Environments den code durchschreibt. Du hast jetzt mal in Wirklichkeit drei Layer, eine Entwicklungsumgebung, da ist die Schiene gemeint, dass die Engineering Organisation selbst verwendet. Wenn du Codeänderung pushst, springt der Build Prozess an und die neue Version von Jira wird automatisch in Entwicklungsumgebung deployed. Und damit hat die gesamte Engineering Org. die gesamte Änderung als Allererstes. Ich weiß nicht was da die Cycle time ist, vielleicht 20 Minuten zwischen den Moment, wo du den code commitest, bis zum Punkt, wo die neue Version da ist.

Dann gibt es eine Staging-Umgebung, da hängt in Wirklichkeit ganz Atlassian drauf. Und am Ende des Tages gibt es eine Production, wo alle Kunden draufhängen. Production ist auch noch unterteilt, da gibt’s das Konzept oder die Idee von Kanaree-Kunden, wo man sagt, das kommt vom Kanarienvogel (diese Metapher von früher im Bergbau haben die Leute Kanarienvögel mitgehabt und man hat beobachtet, ob die Kanarienvögel bewusstlos werden. Wenn sie bewusstlos werden und runterfallen ist irgendein gasaustritt oder zu wenig Sauerstoff und wir sollten raus aus der Grube) die Idee ist, man hat eine Handvoll Kunden, die diesen Code vor allen anderen kriegt und die Idee ist, dass, falls irgendwas schief geht, falls ein Bug drinnen ist, dann ist der Blast-Radius mal kleiner. Wenn man mal nur 10.000 Benutzer, die von Bug betroffen sind, statt 10 Millionen, ist es für die 10.000 genauso schlecht wie, wenn es 99.000 andere auch hätten. Für den Rest der 10 Millionen, die haben den Fehler nicht. In Wirklichkeit ist das viel Komplexität und relativ viel Engineering dahinter, hinter den tatsächlichen release Prozess. Wie wandert Code jetzt zum Kunden, welche Stufen geht der durch, welche Checks passieren da. Da gibt es eigene Teams, die sich drum kümmern. Ich habe eine Organisation geleitet, die für den Prozess des Frontends verantwortlich ist und das eine hochkomplexe Angelegenheit, keine Frage.

A: In dem Fall müsste trotzdem ein spezieller Testprozess definiert sein oder wie schaut so a Testprozess aus?

B: Auch hier gibt es verschiedene Systeme. Eines, was dazu zu sagen ist, ist, dass mir erst in letzter

Zeit bewusst worden ist, wie viele Firmen eine eigene Testabteilung haben. Tatsächliche Tester, deren Jobtitel Tester oder Quality Assurance ist, aber in Wirklichkeit halt Tester, die dann, den Code, den die Entwickler produzieren, testen und wenn die Tester sagen, es ist alles ok, dann kommt es zum Kunden. Bei Attlassian und vielen anderen Softwarefirmen, gibt es diese Jobbezeichnung einfach nicht mehr. Das war bei Attlassian bisschen ein Kulturwechsel. Als ich begonnen habe, gabs sehr wohl noch Tester. Aber keine klassischen mehr, die irgendeine Version testen, sondern in Teams integriert und deren Aufgabe war eigentlich eine gewisse Awareness zu schaffen und Kultur für Softwarequalität zu erzeugen. Über die Jahre kam man davon ab, da man zu den Punkt kommt, dass man das schon so weit im Engineering integriert hat, dass man diese dezidierte Rolle eigentlich gar nicht mehr braucht. Dann haben wir die ganzen Quality Assurance Arbeiter zu normalen Engineers um- und ausgebildet. Und die sind ganz normale Entwickler. Die Idee ist die, dass in Wirklichkeit jeder Engineered für die Qualität verantwortlich ist. Das was man früher sah, dass der Engineer seinen Code macht und was falsch macht und jemand anderes sich darum kümmert. Von den wollen wir massiv weg. Das ist die ganze Idee von Dev auf Stage. Development kümmert sich auch, übernimmt Verantwortung für Operation. Develop und Operation kommen zusammen und ergeben DevOps.

Und das ist natürlich was, wo man bei Attlassian sehr viel Wert legte und ein großes Thema war. Sprich manuelle Tests gibt nicht mehr, es gibt keinen, der das manuell testet. Natürlich, während du den Code schreibst, wirst du schon deine innere Entwicklungsschleife vielleicht irgendwie ausprobieren. Aber das Ziel ist eher, dass du eine gesunde Testpyramide hast. Ich weiß nicht, ob du mit dem Konzept vertraut bist, aber die Idee ist, du hast verschiedene Kategorien von Tests und idealerweise ist das ganze pyramidenförmig aufgebaut. Je näher man am Code ist, desto mehr Tests hat man. Sprich man hat sehr viele Unittests. Warum? Weil billig auszuführen, laufen sehr schnell, sehr reliable auszuführen, weil es keinen Grund gibt, warum es am Monitor fleckig sein soll – manchmal grün, manchmal rot – der ist entweder grün oder rot und soll nicht fehlschlagen. Das heißt Unittests sind gut. Darüber macht man Component Integration Tests. Wo man sagt, ok kann eine Komponente mit der anderen Komponente funktionieren, die beiden Module miteinander. Davon hat man eher weniger. Ganz an der Spitze der Pyramide hat man End-to-End Integration Tests. Wo man sagt, ok funktioniert das ganze System von vorne bis hinten. Von denen will man wenig haben, weil sie extrem mühsam zum Schreiben sind, weil sie sehr lange brauchen, um sie auszuführen, um zu laufen und sehr anfällig sind dafür, dass sie sehr leicht fehlschlagen. Es können 1000 Sachen schief gehen und der Test ist rot. Es heißt aber nicht notwendigerweise, dass man einen Bug programmiert hat. Das kann ein Netzwerkproblem sein, oder anderes. Was sich durch die Testpyramide durchzieht ist, dass man das voll automatisiert haben will.

Bei Unittests sowieso, die sind natürlich automatisiert. Auch die Integration Tests, die End-to-End Tests durchgehend, da gibt es elegante Möglichkeiten das vollautomatisiert zu machen.

Da geht man heutzutage eher in die Richtung zu Post-Deployment-Verification. Man deployed etwas, das geht raus und dann führt man gewisse Tests gegen die Production-Instanz aus. Das macht man regelmäßig und führt alle fünf bis zehn Minuten automatisierte gegen Production aus. Die Idee ist, dass man sagt, dass es weniger darum geht zu verhindern, dass man einen Fehler im Production

shippt. Aber man sieht ein, dass es nicht absolut vermeidbar ist und das Ziel ist, dass man das möglichst schnell in Production erkennt und möglichst schnell Fehler rückgängig macht, indem man zurückrollt. Das ist einerseits der automatisierte Test, der alle paar Minuten läuft. Ein anderer Mechanismus ist Monitoring, ein Riesenaspekt. Die ganzen Silicon Valley Firmen investieren absurdes Vermögen in Monitoringfirmen. Da gibs eine in Österreich. Firma dynatrace, die sehr erfolgreich ist, die bietet solche Monitoring Lösungen an, eine wahnsinnig erfolgreiche Firma. Da ist das Ziel eher, dass man akzeptiert, dass ab und an ein Fehler raus geht und das Ziel ist, dass man den Fehler möglichst schnell erkennt und rückgängig macht.

A: Zum Thema automatisierten Testen, da habt ihr wahrscheinlich auch ein Bitbucket-Piplines verwendet oder wie kann man sich das vorstellen?

B: Für die automatisierten Tests meines Wissens nicht wirklich. Da haben wir etwas selbst geschnitzt, eine eigene Lösung. Aber Bitbucket Pipelines spielen in den gesamten Deploymentprozess eine große Rolle. Dieser Weg von „du als Entwickler commitest deine Änderung“ bis hin zu „deine Änderung ist bei dem Kunden“ ist ein komplexer Prozess. Dort werden wahnsinnig viele Stufen durchgespielt und der ist über Bitbucket Pipelines abgebildet ist. Wobei man fairerweise dazusagen muss, dass in Jira, wo noch viel basierend auf Bamboo, was das Vorgängertool von Bitbucket gewesen ist, aber die Idee ist die gleiche.

A: Fokus war jetzt Großteils auf Atlassian. Sagt dir Microsoft Azure DevOps etwas oder hast du Erfahrungen damit?

B: Ich habe persönlich nicht wirklich Erfahrungen mit den Microsoft Stack. Bei Atlassian verwenden wir unsere eigenen Tools. Github ist ja glaube ich im Microsoft Stack eingebunden und Github verwende ich nun in meinen Start-up, bin natürlich mit GitHub vertraut aber meiner Meinung nach ist es Geschmackssache, ob du das eine oder das andere verwendest. Github ist im Vergleich zu Bitbucket wahrscheinlich das etwas ausgereiftere Tool. Das war immer schmerzhaft bei Atlassian, weil es in dem Productivity Space, wo sich Atlassian aufhält, verschiedene Brennpunkt gibt. Einer davon ist Issue Tracking und dort ist Atlassian sehr gut positioniert, weil Jira absoluter Marktführer. Natürlich gibt es Konkurrenz. Es war ein guter Deal, dass wir Trello gekauft haben. Das war ein sehr schlauer Move, aber dort ist Atlassian sicher der Marktführer. Ein anderer Brennpunkt ist Kundenmanagement, sprich Git-related Stuff und dort ist Atlassian mit Bitbucket vertreten aber tut sich gegen Github irrsinnig schwer und seit Microsoft Github übernommen hat, ist dort ja eine wahnsinnige Engineering Power dahinter und ich glaube nicht, dass Atlassian sich aus diesem Teil zurückziehen wird aber der Fokus wird sicher sein mit Jira und Confluence eine Integration von Bitbucket zu erstellen und weiter voranzutreiben. Nichtsdestotrotz muss man einfach akzeptieren, dass in diesem Bereich Microsoft mit Github und mittlerweile auch Gitlab eine riesige Konkurrenz ist. Ich glaube die Idee, dass eine Firma den ganzen Stack abbildet, das haben die meisten schon eingesehen, das wird nie funktionieren, da bestimmte Firmen bestimmte Stärken hat und Engineering Firmen bestimmte Mischung aus verschiedenen Anbietern haben werden. Ein anderer Brennpunkt ist die ganze realtime Kommunikation, wo Slack sehr stark ist. Microsoft Teams versucht zum Beispiel dort hineinzudrängen. Aus diesem hat sich Atlassian zurückgezogen. Wir waren da drin mit Flipchat und dem Nachfolger davon, aber man hat eingesehen, dass gegen Slack werden wir

nicht bestehen können und haben uns dort zurückgezogen und in Slack investiert und versucht das über diese Schiene zu fahren. Um zu deiner Frage zurückzukommen: ich habe eigentlich nur mit GitHub Erfahrung. Ich habe mal vor vielen Jahren mit den Jetbrains Tools gearbeitet, mit den Deploymenttools wie TeamCity war damals gut aber wiederum machen diese Tools alle ziemlich das Gleiche. Also ich glaube, dass alles was man in Azure DevOps abbilden kann, kann man auch in Bitbucket abbilden. Ich glaube es ist dann eher auch Geschmackssache.

A: Siehst du dann eigentlich konkret im Entwicklungsprozess oder an dem Beispiel, das du genannt hast, dass dort mehr in Richtung CICD geht und täglich deployed wird und erst im Nachhinein geschaut wird, ob Fehler existieren? Wie sieht dort die Rolle des Projektleiters aus oder des Scrum Masters und vor allem wie läuft dort die Kommunikation ab, denn es muss ja immer eine zuständige Person geben, für den derzeitigen Stand?

B: Contentdeployment ist definitiv das Ziel und ich glaube in diese Richtung wird jede Firma früher oder später gehen, ist einfach der bessere Ansatz. Die Rolle des Projektleiters: ich glaube nicht, dass sich diese überhaupt ändert. Es gibt weiterhin Zuständigkeiten. Es gibt auch weiterhin Produktverantwortliche und Featureverantwortliche. Wenn ich zurückdenke an meine Arbeit bei Atlassian. Ein Team war da zum Beispiel verantwortlich für das Navigationskonzept. In Jira Cloud gibt es da eine Topbar, also eine Leiste, mit der du herumnavigierst und die Verantwortung dafür hat eines meiner Teams gehabt und dieses Team besteht aus einer Reihe von Engineers, einen Team-Lead, der die Management-Verantwortung für das Engineering hat, dann gibt es einen Produktmanager, der die Verantwortung des Produktes trägt und Fragen klärt wie „Verstehe ich den Kunden“ oder „Verstehe ich die Kundenanforderungen“ und macht auch Customer Research, schaut sich die Analyse und Statistiken an und analysiert, ob das Feature überhaupt Sinn ergibt oder ob man das weiter umbauen muss. Die Rolle ist also produktverantwortlich und da gibt es dann noch die Designer, die die Visuals erstellt und die Userinteractions erstellt. Ich glaube, ob du das in einen großen Big-Bang Releases, die jetzt alle paar Wochen oder Monate passieren oder ob du das laufend released macht es glaub ich keinen großen Unterschied in den Rollen, da du weiterhin deine Verantwortlichkeiten hast. Aber bei so großen Softwareprojekten wie Jira musst du das dann natürlich runterbrechen, weil es da keinen mehr gibt, der das Ganze im Blick hat. Natürlich schon aber von einer höheren Abstraktionsebene. Da brauchst du Leute, die Verantwortung übernehmen für einen Subteil, wo sie sich auch wirklich auskennen. Was weiterhin eine große Herausforderung ist, ist die Kommunikation, weil es dort weiterhin einen großen Abhängigkeitsgraphen gibt, denn mein Team ist verantwortlich für die Navigation. Aber dann gibt es ein anderes Team, das einen zusätzlichen Eintrag in der Navigation haben will. Das muss dann wiederum organisiert und orchestriert werden. Wie sind dort die Abhängigkeiten? Wie ist dort der Zeitplan? Dann muss besprochen werden, welche Vorbereitungen getroffen werden müssen und wann diese fertig sein müssen. Da gibt es im Jira Team laufend Planungsaktivitäten, wo dann verschiedene Teams zusammenkommen und man einfach versucht zumindest für die nähere Zukunft einen Plan zu erstellen. In Wirklichkeit gibt es verschiedene Zeitdimensionen, die parallel zu bedenken sind. Ich habe eine Vision, wo mein Produkt in den nächsten Jahren stehen soll. Diese soll ich im besten Fall artikuliert haben und ein gemeinsames Verständnis erstellt haben. Dann wird bei Atlassian von der

Executive Board bestimmt, wo man in den nächsten Jahren hinwill. Dann müssen sich die einzelnen Produkte überlegen, was das für mein Produkt, zum Beispiel Jira bedeutet. Dort wird dann ein genauerer Plan definiert, der jedoch dann nur für die nächsten zwei Jahre zum Beispiel artikuliert ist. In zwei Jahren möchte Jira das Ziel erreichen, dass das Unternehmen in den nächsten fünf Jahren das Ziel erreicht. Auf der nächsten Ebene, also in der Jira Plattform, wo muss ich dann in ein Jahr sein, damit das möglich ist. Und je weiter du runter kommst desto konkreter wird das aber auch desto kurzfristiger wird das. Also wenn ich das mit meinem Team mir überlege, was wir im nächsten Quartal machen, dann sollte natürlich die Vision im Blick behalten, aber gleichzeitig auch einen kürzeren Zeitraum ins Auge fassen und konkret sein und je konkreter der Plan desto kurzfristiger muss dieser sein. Und das ist natürlich ein sehr großer Planungsaufwand und das ist dann auch anders als in klassischen Release Management. Beim klassischen Release Management hast du eben diese Punkte, wo das gesamte Produkt zusammenkommen muss, damit wir da eine Version haben. Während du in einer Welt wo laufend released wird, dort gibt es diese großen Big-Bang-Releases nicht. Aber das heißt nicht, dass du dir nicht genau diese Gedanken machen musst. Du machst das dann vielleicht etwas kontinuierlicher.

A: Vielen Dank für das Interview.

ANHANG F - Transkript Interview 6

A: Hallo und danke für die Zeit vorneweg. Könntest du kurz erklären welche Position du in der Firma hast, welche Berufserfahrung du vorzuweisen hast und auch wie lange du in dem derzeitigen Unternehmen tätig bist?

B: Meine Position ist jetzt Tech-Lead, das ist gleich wie ein Entwicklungsmanager. Ich manage dort 5 Teams mit 27 Mitarbeiter, rein in der R&D und bin im Unternehmen seit Ende 2018. Berufserfahrung im IT-Bereich seit etwa 20 Jahren+ und war zuvor auch in Südafrika und habe sehr viel mit eCommerce gemacht, also in den letzten 10 bis 15 Jahren vor allem. Ursprünglich war ich selbst Softwareentwickler, Architekt und CTO für ein eCommerce Unternehmen in Südafrika und habe jetzt die Management-Rolle bei dem Unternehmen übernommen.

A: Für das Release Management verwendet ihr Unterstützungstools? Kannst du da sagen welche konkret ihr da verwendet?

B: Also fürs Release Management an sich machen wir unsere Deployment über Bamboo natürlich. Wir haben ein eigenes Tool geschrieben, um die Umgebung auch zu betreiben und GitLab sowie AWS-CLI für die Pipeline und wir verwenden derzeit PHP für den eCommerce-Teil und Java für den Point of Sales-Teil und haben da unsere Pipelines, die wir darüber laufen lassen. Tooling an sich gibt es den Atlassian Stack, da verwenden wir Jira, um das Issue Management zu machen. Bei Jira verwenden wir dann natürlich auch die Release Möglichkeiten aber reißen das nicht komplett aus und Confluence, das unser WIKI ist, wo wir unsere Dokumentationen machen und natürlich die Q&A-Abteilung verwendet XRAY-Plugin für Jira und dient dem Testmanagement.

A: Aber wenn ich das richtig verstanden habe, habt ihr auch selbst Weiterentwicklungen und auch Plugins erstellt für Jira oder Bamboo erstellt zur Entwicklung, damit ihr auf eure Umgebungen deployen könnt?

B: Ja also wir haben zwei eigene Deploymentskripte, die sind applikationsspezifisch und haben ein Tooling gebaut, um diese Deployment zu entwickeln beziehungsweise diese zu betreiben und wir nennen das unsere Automator-Umgebung, mit dem kann man ein Post-Deploy machen, Prozesse stoppen und starten.

A: Du bist 2018 in das Unternehmen eingestiegen. Hat es da Jira und Confluence schon im Unternehmen gegeben?

B: Ja.

A: Hast du es davor auch schon gekannt?

B: Ja.

A: Wie war, wenn du da zurückdenkst, der Erstkontakt zum Tool? War das für dich einfach zu verstehen?

B: Ja also es war sehr einfach Jira und Confluence zu verstehen. Speziell dann auch GitLab recht einfach. Ursprünglich bin ich vom SVN-Umfeld gekommen und mit GitLab ging es dann wesentlich einfacher.

A: Die Nutzung ist dann mit der Zeit auch immer mehr geworden mit den Tools oder war es von Anfang an, dadurch dass es so einfach zu verstehen war, gleich von Anfang an klar, dass ihr das Tool so

verwenden wollt, wie ihr das jetzt verwendet?

B: Wir haben definitiv während dem Doing Sachen gelernt und dementsprechend inkrementell ständig versucht das Release Management zu verbessern. Also ursprünglich war es wirklich so, dass wir nur auf Fixversions gearbeitet haben, das war einfach. Als ich angefangen habe, hat es 5 oder 6 Kunden gegeben und das Release Management mit momentan 25+ Kunden hat dann nicht mehr gereicht, wenn man nur Fixversions hat. Dann hat man mit Jira und den Release Labels, sowie Confluence-Komponenten verwendet, um eine bessere Sicht auf Releases zu kriegen.

A: Ihr habt also nicht nur ein Produkt, an dem gearbeitet wird, sondern auch mehrere Produkte, die für die 25 Kunden deployed werden?

B: Ja.

A: Verwendet ihr im Release Management auch Kennzahlen, an denen ihr seht, wie gut euer Release Management ist? Oder könnt ihr sagen, dass euer Release Management über die letzten Jahre besser geworden ist?

B. Es ist schwierig, weil es gibt keine reinen Support-Releases. Es finden sehr oft Feature-Entwicklungen statt. Also was wir schon haben ist im Release Prozess über die Q&A also das Testmanagement eine Aussage, wie schnell der erste Durchlauf ist und mit den Testerfolgen, die wir haben, ist für jeden Release in Jira einen Feedback Epic, wo dann der Kunde Issues einstellt, die bezogen auf den Release auftreten und aufgrund der Anzahl an Issues hat man schon relativ einfach eine Kennzahl, wie gut ein Release ist. Aber wie gesagt, wenn es große Feature-Entwicklungen gibt, dann wird es nach dem Deployment mehr Issues geben.

A: Da ihr auch einen ziemlich breiten Stack von Atlassian verwendet, kannst du da auch eine Prozentzahl wieviel du glaubst von der Funktionalität vom Tool zu verwenden oder ist das schwer zu sagen?

B: Ich denke schon, dass es wir 70% vom Tooling ausnutzen. Wir gehen schon in die Breite und nicht in die Tiefe. Wir verwenden also alles wie Kanban Boards und unterschiedlichen Dashboards. Wir verwenden was notwendig ist, um unseren Prozess zu unterstützen. Also wir haben schon so unser Tooling im Einsatz, wie wir denken, dass es effizient für uns läuft.

A: Bezüglich Automatisierung. Ihr habt dort die Pipelines, die du erwähnt hast. Kannst du da auch Prozessteile sagen, die ihr automatisiert habt?

B: Also natürlich der Build- und Deploymentprozess ist automatisiert. Wir sind jetzt schon seit knapp 2 Jahren dabei, die Testautomatisierung voranzutreiben also das ist unterschiedlich, je nach Team und Plattform-Produkt ist entweder Side-Pressing im Einsatz oder Karate für Api-Tests. Für das reine Testcase Management verwenden wir XRay. Das sind die unterschiedlichen Tools, die wir im Einsatz haben, um einen höheren Grad an Automatisierung zu haben.

A: Neben den Tests und der Automatisierung des Build- und Deployment Prozesses gibt es noch weitere Teile, die ihr automatisiert habt?

B: Ja das Monitoring und Alerting sowie Postdeployment-Aktivitäten können wir auch mit unseren Tools erweitern und automatisieren, wenn es notwendig ist.

A: Den Release-Prozess verwendet ihr dann auch für alle Produkte oder ist der doch an den verschiedenen Produkten und Kunden anzupassen?

B: Der Release Prozess ist generisch und passt auf alle Produkte. Wir sind jetzt auch gerade dran, den Prozess so weit anzupassen, dass dieser auch für andere Teams anwendbar ist.

A: Würdest du auch so weit gehen zu sagen, dass der Prozess auch für andere Unternehmen in dem Bereich verwendbar sein kann oder ist dieser dann doch noch unternehmensspezifisch?

B: Ich denke der Prozess selbst ist definitiv nichts Neues und kann auch in anderen Unternehmen verwendet werden. Es gibt natürlich gewisse Feinarten, die bei uns etwas spezieller sind wie Rollenbeschreibungen. Was sicher unterschiedlich ist, sind die Artefakten. Also diese sehen in den eCommerce Unternehmen sicher immer anders aus aber der Prozess selbst kann verwendet werden.

A: Im Team habt ihr da auch Spezialisten, die sich mit dem Tool beschäftigen? Also wenn Jira ein neues Release veröffentlicht und dass versucht wird, die neuen Funktionen in den aktuellen Prozess einzubinden oder wie kommen diese neuen Funktionen in das Team?

B: Wenn wir neue Funktionalitäten brauchen, suchen wir die auch aktiv. Also am Beispiel Testmanagement, wenn wir sagen wir brauchen eine Unterstützung für die Testmanagementverwaltung in Jira und haben uns selbst in der Entwicklung eine Analyse und einen Proof of Concept erstell und uns dort geeinigt. Also wenn dann passiert das innerhalb des Entwicklungsprozesses, wo wir das Tool dann aussuchen. Das gleiche war dann bei den Code Reviews. Also wir entwickeln ja mit PHP und PhpStorm, da haben wir uns dann für JetBrains entschieden, um Codereviews zu machen, weil das eines der wenigen Tools mit einer Jira-Integration ist. Also es ist Teil des Prozesses und kommt primär aus der Entwicklung.

A: Also gibt es keine spezielle Rolle, die für das Tooling in der Entwicklung zuständig ist?

B: Das wird aufgeteilt nach Bedarf und Notwendigkeit. Ansonst rein im IT-Ops-Bereich gibt es natürlich die ITIL-Leute, die sich für die Jira und Confluence Sparte kümmern. Das heißt jedoch nicht, dass die zuständig sind, neue Features in die Firma zu bringen.

A: Welche Entwicklungsmethode verwendet ihr in euren Team?

B: Das hängt vom Team ab. Also wir folgen jetzt den Kanban-Prozess, wo wir vordefinieren, wie groß der Umfang sein soll. Haben ein bisschen Scrum dabei. Über Kanban wird eine gut definierte Feature-Entwicklung abgebildet, weil es sehr abgrenzbar ist. Den regulären Sprint-Prozess verwenden wir für unsere Supportsprints. Man sagt dort je nachdem, wieviel Support Issues kann ich in der Woche einplanen und arbeiten diese in den Sprints ab. Wenn man Issues nicht schafft, nimmt man die dann einfach in den nächsten Support Sprint mit.

A: Welche Rollen habt ihr in euern Prozess abgebildet? Ist dort auch die Kundenperspektive berücksichtigt?

B: Ja. Die Rollen sind vielfältig. Wir haben dort die Entwickler drinnen. Wir haben den Endkunden, der Abnehmer, der Software ist. Wir haben den Support drin und den Release Manager, Projektmanager, Quality Assurance. Die Rollen sind auch klar definiert im Prozess.

A: Kannst du zu den Problemen sagen, auf welche ihr immer wieder gestoßen seid im Prozess?

B: Rein technisch ist die Releasedurchführung problemlos möglich. Es gab Phasen, in denen wir die Technologie erneuert haben, wie wir das mit PHP gerade machen. Es ist mehr ein inhaltliches Wachstumsproblem, weil unsere Software sehr spezifisch ist. Man braucht eine gewisse

Kundenunterstützung, um diese Software zu liefern. Wenn man sich anschaut in 2019 als ich angefangen hab, hatten wir 11 aktive Installationen und mit diesem Jahr werden wir 31 Installation haben. Wir haben ein enorm großes Wachstum gehabt und immer mit einer gleich großen Mannschaft. Da versucht man mehr auf in die Richtung Automatisierung und Tooling zu arbeiten, damit die gleichen Leute mehr schaffen können. Das sind so mehr die Hürden. Also das Wachstum ist definitiv eine Hürde, wo das Tooling die Skalierbarkeit vor allem unterstützt.

A: Du hast es gerade angesprochen. Die Probleme, die ihr gehabt hat, haben weniger mit dem Tool selbst zu tun gehabt. Habt ihr dann trotzdem die Probleme konkret aufgelistet oder ist notiert worden, dass diese Probleme aufgetreten sind, und gibt es dafür zuständige Personen?

B: Ja das war so eigentlich die Rolle von mir, um den Prozess dann so anzupassen, dass wir effizienter Releases parallel durchführen können. Wir hatten zuvor, wie ich in die Firma gekommen bin, für jeden Kunden einen eigenen Release und eine eigene Release Version. Da hat man im produktiven Betrieb mehrere unterschiedliche Versionen gehabt. Mit dem kommen dann Hotfixes und Batches und das wird dann unwartbar. Wir haben jetzt mittlerweile umgestellt auf eine Release Version. Die läuft dann über 4 Wochen und wird dann den Kunden ausgeliefert. Das sind so die Sachen, die wir aktiv in das Tagesgeschäft reinholen und die sind auch notwendig. Also die Rolle des Techleads und der Teamleads zusammen mit dem Q&A, die diese Informationen dann sammeln und entsprechend skalieren.

A: Wenn ich das richtig zusammenfasse, hat sich im Grunde genommen an der Organisation nicht viel geändert. Nur anhand des Tools hat sich der Entwicklungsprozess weiterentwickelt und ist dadurch dann auch besser geworden.

B: Nein also wir haben als ich gekommen bin sehr viel umstrukturiert. Über die letzten zwei Jahren ist dann nicht mehr viel passiert. Wir haben zum Beispiel eine zuvor ein ausgelagertes externes Q&A Team gehabt, dass wir in die Entwicklungsmannschaft reingezogen, wo das Entwicklungsteam jetzt zusammenarbeitet, was ein wesentlicher Umbruch gewesen ist. Wir haben auch Support in den Entwicklungsprozess mit einbezogen. Es gab einen kleinen Zeitraum im Entwicklungsprozess, wo wir jetzt das Supportteam direkt mitnehmen. Es gab schon diese organisatorischen Änderungen. Wir haben sicher jedes Jahr zwei bis drei Anpassungen auf eine organisatorische Struktur, um mit dem Wachstum und dem Produktwachstums zurechtzukommen.

A: Wie oft aktualisiert ihr den Release Prozess und wann wird der reviewed?

B: Das ist ein ständiges Reviewen und Anpassen. Es werden Tuningmaßnahmen gemacht, wo es notwendig ist, das wird aber auch regelmäßig gemacht. Also wenn wir ein Releasefenster haben, gibt es ein Post-Release-Meeting, wo wir darüber schauen, Feedback einsammeln, extern sowie intern und dann wird dieses Feedback nicht nur in den Release Prozess eingebunden, sondern auch in die Produktentwicklung eingetragen. Wenn man es eingrenzen würde, wäre das im 4-Wochen Zyklus, was unser Release Zyklus ist.

A: Danke für das Interview.

ANHANG G - Qualitative Inhaltsanalyse Tabelle

Person	Text	Kategorie
1	Seit neun Jahren arbeite ich am gleichen Großprojekt	Erfahrung
1	Die ersten etwa sieben Jahre als Entwickler und seit etwas über einem Jahr als Scrum-Master und parallel in einer Devops-Rolle. Wir haben diese Rolle eingeführt, weil wir das Projekt von Entwicklung über Test bis zu Betrieb fast allein machen.	Rolle
1	Ansonsten als Scrum-Master den aktuellen Prozess begleiten, ein wenig Requierement Engineering dazu und versuchen, den Testprozess, den wir seit zwei Jahren auf neue Beine gestellt haben, kontinuierlich noch weiter voranzutreiben.	Rolle
1	Derzeit schon mit Devops-Unterstützung, das heißt Buildpipelines und automatischen Tests.	Verwendung
1	Wir verwenden automatisierte Tests, denn die manuellen Tests würden viel zu lange dauern und der Aufwand wäre untragbar.	Automatisierung
1	Wir sind 3 Backend-Entwickler und ein Frontend Entwickler.	Team
1	Der Prozess sieht vor, dass wir so bald unsere Features funktionieren, Teile davon über automatische Tests abbilden beziehungsweise ist es verpflichtend, dass manuelle Test, der sowohl von einem anderen Entwickler als jemand, der nicht sämtliche mitentwickelt hat, testet beziehungsweise die manuellen Tests werden auch von den Kunden ausgeführt.	Prozess
1	Jira ist für uns nur eine Datenquelle, weil unsere Endkunden mit Jira arbeiten und ihre Tickets tauchen bei uns eben in Open Issue Lists auf, sind also als Artefakte im Projekt vorhanden.	Tool
1	Ich habe Jira schon vor fast einem Jahrzehnt gesehen, als ich in der vorigen Firma war und da habe ich es als Bug Tracking Software erlebt.	Tracking
1	Da ich in diesem Projekt aber nicht viel mitentwickelt habe, sind meine Erfahrungen eher beschränkt.	Tool
1	Es sind zusätzliche Features dazugekommen und wir haben in diesen drei Jahren drei Scrum-Master gehabt, das heißt in unterschiedlichen Ausprägungen.	Verwendung
1	Devops in unserem Projekt gibt es seit 2017.	Erfahrung

1	Man kann DevOps als Taskliste benutzen, was funktioniert aber mit einem agilen Ansatz oder einer Projektleitung oder Koordination in dieser Größe ist es damit nicht getan. Wir haben gesehen, dass es nicht sehr zielführend ist, wenn man mehrere Tasklisten hat. Es gab einen Moment, wo es vier Listen gab mit Todos und kurz gesagt, diese Aufteilung hat nicht sehr gut funktioniert. Seit es Sprints gibt, in denen die Dinge, die passieren müssen, drinnen stehen, und Seitenthemen nicht nebenbei passieren, ist die Übersicht über die Dinge, die passieren sollen, deutlich einfacher. Wir haben zusätzlich noch ein Ticketing System. Das ist die zweite Liste von Dingen, um die wir uns kümmern müssen. Aber mit zwei Listen von Tasks kann man haben, mehr halte ich für sinnfrei.	Planung
1	Der Erstkontakt war bei einer Abart von agile, die damals gemacht wurde und es war der Übergang zwischen Tasklisten-basierten Arbeiten und einer agileren Herangehensweise.	Arbeitsmethode
1	Wir haben zu Beginn Schwierigkeiten gehabt, User Stories, Epics und Tasks so zu formulieren, dass sie transparent genug sind. Zu Beginn war DevOps auch rein intern bei uns, das heißt, unsere Kunden haben es nicht gesehen. Da es damals noch keine Dailys gab, war der Kommunikationsfluss für den Kunden, was aktuell passiert, wo es Probleme gibt, wo man vielleicht Unterstützung braucht, schwieriger.	Probleme Prozess
1	Seit zwei Jahren gibt es Daily Standups. Der Project Owner ist dabei, hat den Überblick über die laufenden Tasks und weiß zumindest zweimal in der Woche ganz genau, was wir gerade tun, wo die Leute arbeiten und auch wo Bugs auftauchen. Also sowohl interne wie auch externe Bugs, die von Kundenseite aufgedeckt werden.	Kommunikation
1	Das geflügelte Wort bei uns ist: Es gibt keine Geheimnisse. Also es muss transparent sein, denn anders kann die Zusammenarbeit in diesen komplexen Themen eigentlich nicht mehr funktionieren.	Tracking
1	Durch ein paar glückliche Fügungen haben wir auch Leute ins Team bekommen, die Automatisierung Erfahrung hatten. Und dadurch haben wir es auch geschafft, automatisierte Tests einzuführen. Wir sind auch immer dabei und werden vermutlich nie fertig sein. Aber es macht mich stolz sagen zu können, dass wir inzwischen fast 300 automatische Tests haben, die bei einem Check-in der Branche laufen, aktuell nur 90 Prozent davon positiv. Aber es sind gerade sehr große Entwicklungen.	Automatisierung
1	DevOps bietet unterschiedliche Herangehensweisen. Also in meiner Rolle als Scrum Master freue ich mich über die Taskbar und habe die	Tracking

	auch geringfügig customized dass wir über Tags und Farben sehr einfach sehen, wo Dinge liegen. Wir haben einige Dashboards gebaut, um zu sagen, wie viele Leute wie viele Stories gleichzeitig offen haben. Wer sehr viele Stories in Verantwortung hat.	
1	Das macht uns die Planung einfacher und auch ein erstes Gefühl für Überlastung gibt es sehr schnell.	Planung
1	Die Testcases, die ein weiterer Teil von DevOps Portfolio sind, werden vom gesamten Team genutzt und im zweiwöchigen StandUps auch präsentiert und neue Features, die wir dort finden und welche die uns nützlich sind werden weitergegeben.	Verwendung
1	Die automatisierten Tests werden von den Kollegen, der früher als Testengineer gearbeitet hat, getrieben.	Automatisierung
1	Also wir sind ein diverses Team mit dem gemeinsamen Ziel und DevOps wird von jedem so gut genutzt für Scrum und gibt uns Wissen so weiter, weil es uns das Leben extrem einfacher macht.	Team
1	Wir haben zum Beispiel auch in der Release Planung ein Custom Field eingeführt.	Verwendung
1	Bei den Userstories sehen wir, ob sie deployed sind oder nicht. Somit tun wir uns im Vergleich zu früher, wo nicht ganz klar war, ist etwas schon fertig oder ist es schon getestet oder ist es schon deployed oder ist es überhaupt deploybar. Das ist heute relativ einfach. Es gibt ein Dashboard, da ist eine Liste von Userstories drinnen, die getestet und abgenommen sind. Die Kunden können in einem kurzen Meeting die Features festlegen, die mit dem nächsten Release deployed werden, sollen beziehungsweise können wir auch zwischen den StandUps, ob es ein kleines wöchentliches Release ist oder als Hotfix rausgehen soll. Also nachdem diese Daten sehr transparent sind, ist es sehr einfach solche Themen zu behandeln. Wir haben es jetzt aus Entwicklerteam viel einfacher als Excelliste sukzessiv zu pflegen oder die Arbeit einen Release Plan zu erstellen.	Tracking
1	Also einen Release Plan wurde jetzt letztes Jahr einmal gefordert, aber wir sind sehr schnell reingekommen, dass dadurch dass sich die Prioritäten auf der Kundenseite wieder verschieben eine Release Planung für uns also nur eingeschränkt nützlich ist, denn es gibt gewisse große Features, die in bestimmten Zeitpunkten angekündigt werden und diese halten auch meistens es sei denn irgendetwas anderes wird dringender, dann kommt dieses in dem Release oder ein Zwischenrelease passiert. Andererseits haben wir Features fertig seit einem dreiviertel Jahr, die wir nicht live nehmen können, weil die Prozesse im Business noch nicht so weit sind. Solche Dinge lassen	Probleme Extern

	sich relativ einfach abbilden und sind transparent und macht Scrum einfacher.	
1	Ich behaupte wir halten uns an Scrum. Mit jeder Iteration gibt es ein Inkrement of potential shippable Software. Es gibt Iteration, wo es danach nichts gibt, dass wir deployen sollen. Es gibt Iteration wonach wir einen Release machen. Das ist abhängig davon was die Kunden brauchen. Wir wissen typischerweise einem Monat vorher, dass ein Release ansteht. Kleinere Themen können wir unter der Woche releasen.	Arbeitsmethode
1	Es gibt keine klaren Kennzahlen. Es gibt Beobachtungen.	Metriken
1	Also wir haben viel in unserer Art des Arbeitens geändert. Wir haben jetzt weniger große Releases und sehr wenig Releases, wo wir viele Subpräventionen haben. Nachdem wir nicht zwischen Hotfix und Feature-Release unterscheiden kann ich es, ohne in die Release-Nodes und in das Deploymentpaket hineinzuschauen nicht aus dem Stehgreif beantworten.	Arbeitsmethode
1	Aber ich habe eine Präsentation erstellt Ende letzten Jahres, wie sich die Ticketzahlen, Supportstunden und Releases in den letzten Jahren verändert haben, das kann ich dir zukommen lassen.	Metriken
1	Ich habe da sehr gerne die Anzahl der Tickets hinzugezogen. Erfahrungsgemäß haben wir die Kapazität etwa 50 Tickets pro Monat zu bearbeiten zusätzlich zu der Entwicklung und seit etwa einem Jahr haben wir pro Monat durchschnittlich 30 Tickets, die aufgehen und auch wieder zugehen. Das heißt also eine Anzahl von offenen Tickets jenseits 20 war vor einigen Jahren normal. Derzeit sind 10 offene Tickets ein „Irgendetwas läuft nicht optimal oder irgendjemand arbeitet nicht genug an Tickets.“	Metriken
1	Ich würde sagen Meetings ohne DevOps wäre derzeit nicht vorstellbar. Also nicht nur ein Daily StandUp ist ohne ein DevOps, vor allem, da wir ja alle im Home-Office sind unvorstellbar. Auch ein Sprint Planning 1 oder 2 ohne vordefinierte Userstories wäre eigentlich absurd.	Kommunikation
1	Und ein einfacher Status über die Entwicklung mit „so stehen wir in diesem Sprint“ ist derzeit eigentlich mit einem Blick zu sehen, im Gegensatz zu man müsste mit 5 Leuten reden.	Tracking
1	Also die Zeiten, die wir in DevOps verbringen ist eigentlich Meetingzeit, weil es immer die Unterstützung des Meetings ist plus unsere Entwicklungszeit ist auch zum Teil in DevOps, weil die	Verwendung

	Userstories und die Akzeptanzkriterien dort drinnen sind plus die Testcases. Also offen ist es bei jedem Entwickler mindestens 50% der Zeit allerdings meistens nur zum Nachschauen.	
1	Bei Meetings natürlich, je nachdem in welcher Rolle die Teilnehmer sind. Wenn die Project OwnerIn ihre Userstories präsentiert und ihr Fragen stellen passiert sehr viel in DevOps. Wenn wir eine Userstory reviewen oder testen, dann in DevOps. Es würde also ohne gehen aber derzeit verteilt nicht.	Kommunikation
1	Es ist bei uns deutlich mehr genutzt worden. Wir sind deutlich agiler geworden. Also wir hatten früher Spezifikationsdokumente, die sehr wichtig waren und zeitweise als einzige Wahrheit betrachtet wurden. Während wir letzte Woche in eine kurze Diskussion gegangen sind, wie genau etwas spezifiziert wurde, bis ich es abgedreht habe mit „Es ist nicht wichtig, was in der Spezifikation steht, was ist das was du brauchst.“ Insofern, es macht das Leben einfacher. Aber das hat auch viel miteinander zu tun, wie wir jetzt arbeiten und wie wir jetzt mit den Kunden umgehen.	Tool
1	Der Prozess ein Deploymentpacket zu schnüren, haben wir vor 8 Jahren gelernt. Also nachdem wir sehr datenbank-lastig entwickeln, gibt es da eine lange Liste von Objekten, die zu deployen sind. Zusätzlich die Tests für die Middleware plus ein Angular Frontend, die sind sehr einfach zu deployen. Unsere Deploymentstrategie sieht vor, dass vom Entwicklungssystem das Ganze auf unser Staging-System als erstes geht. Auf diesem System wird getestet, sowohl von einem anderen Entwickler als auch von den Kunden. Danach gibt es ein System, das heißt bei uns Train, ist allerdings ein Near-Live-System. Das heißt es entspricht exakt den letzten Deploymentstand des Livesystems und von dort geht es erst auf das Livesystem.	Deployment
1	Also die Transparenz, welche Pakete gerade in Entwicklung sind, wie die Features mit dem Aufwand bewertet sind, um jetzt den klassischen Wasserfallmodell, das wir damals benutzt haben, die Abwandlung von agile, die wir seit zwei Jahren benutzt haben und jetzt Scrum kann man sagen.	Arbeitsmethode
1	Geändert hat sich eher das Organisatorische. Wir haben viele andere Listen und Dokumente eliminiert. Wir haben viel Information in ein zentrales System gegeben, das uns das Arbeiten erleichtert und auch transparenter macht, womit uns viel Kommunikation und andere anfällige Prozesse abgelöst haben.	Organisation
1	Diesen Prozess gibt es schon länger als wir DevOps verwenden.	Automatisierung

	Also den Deploymentprozess haben wir noch nicht automatisiert. Teilweise automatisiert auf den Dev- und Stagesystem für die Entwickler. Also über die Pipelines machen wir schon die Deployments. Auf den Livesystem nicht.	
1	Ein Frontenddeployment auf die Stage ist ein Commit in den Integrationsbranch. Ein Middlewaredeployment ist ein Commit auf der Stage in den Integrationsbranch.	Deployment
1	Wir haben derzeit noch keine Strategie unsere Datenbankdeployments zu automatisieren. Das ist etwas, das wir schon lange auf unserer Wunschliste haben, aber uns immer andere Prioritäten dazwischengekommen sind.	Probleme Prozess
1	Natürlich wäre es praktisch ein Deployment, das zu schlechtesten Zeiten vier Leute für drei Stunden beschäftigt hat, was heute eigentlich von einem Entwickler und einem Kollegen, der als Copilot dabei ist und die Arbeitsschritte verifiziert und im Notfall auch deployen kann. Also der Aufwand ist sehr gesunken. Über die Erfahrungen in den Jahren ist ein Deployment jetzt normalerweise kein großer Aufwand mehr.	Deployment
1	Natürlich ist die Deploymentvorbereitung noch immer sehr viel Arbeit. Tests auf den Near-Live System sind noch immer Arbeit, aber im Vergleich zu früher ist ein Deployment jetzt mit sehr viel weniger Aufwand in der Vorbereitung und vor allem mit sehr viel weniger Problemen nach dem Go-Live verbunden.	Probleme Prozess
1	Also die Frage was das wichtigste Feature ist, ist ein Problem, dass wir es richtig an unsere Project Ownerin ausgesourced haben. Also normalerweise zu Sprintbeginn ist es sehr klar was gemacht werden muss und die Probleme mit, diese Probleme kommen zusätzlich dazu, haben wir inzwischen fast ausgeräumt. Inzwischen ist sehr klar, wenn ein zusätzliches Thema aufkommt, das wir anderes nicht machen können.	Organisation
1	Die Datenquellen sind manchmal noch ein Problem, wenn wir uns darauf verlassen, dass wir bestimmte Daten schon haben und das dann zu Beginn oder im schlimmsten Fall mitten in der Entwicklung aufkommt, dass dann die Daten nicht ganz das sind, was wir geglaubt haben, das kann manchmal passieren, aber in diesem Fall glaube ich nicht, dass man das einfach automatisieren kann.	Probleme Extern
1	Was immer ein sehr großes Thema war, waren die Tests. Wir haben früher Testcases in unterschiedlicher Qualität gehabt. Testcases, die sporadisch dokumentiert waren. Dadurch, dass unser Entwicklungsprozess vorsieht, dass ein Entwickler, der dieses	Probleme Team

	Feature nicht entwickelt hat, den Testcase durchführt, bevor ihn die Kunden durchführen haben wir gelernt, unsere Testcases in einer sehr harmonisierten Art zu schreiben. Das heißt, niemand möchte von seinem Kollegen angerufen werden, weil er nicht weiß, wie er dieses Feature testen soll.	
1	Das ist zugegebenermaßen ein Lernprozess, aber sobald das einmal passiert ist und klar ist, dass das Schreiben eines Testcases durchaus einen oder drei Tage dauern kann, ist es auch in der Planung dementsprechend berücksichtigt. Das heißt wenn ich dann als Scrum-Master frage: Ist in dieser Schätzung das Testen schon drinnen, kann es sein, dass sich die Story-Points noch ändern. Das passiert uns immer wieder, dass wir zu solchen Mitteln greifen.	Prozess
1	Wenn wir etwa 60 Rollen im System haben, mit unterschiedlichen Berechtigungen können Berechtigungstest komplizierter werden. Wir haben in der letzten Woche etwa 80 automatisierte Tests für Berechtigungen geschrieben. Dadurch brauchen wir das Subset, das wir händisch hätten testen können nicht mehr testen, denn wir wissen, dass es so funktionieren wird, wie wir es automatisiert testen. Das ist sehr beruhigend.	Automatisierung
1	Typischerweise im Sprint-Retrospektivenmeeting sind wir da sehr ehrlich und typischerweise endet es damit, dass der Scrum-Master sich einen Task fasst, sich um etwas zu kümmern.	Problembehandlung
1	Das reicht von „Gut dafür brauchen wir eine Userstory“ oder wir wissen, dass es irgendwo ein Problem gibt, dass wir eigentlich beheben sollten, aber machen es nicht, dann wird auch hier eine Userstory erstellt. Ja Entwickler sollten keine Userstories definieren, aber wenn sie eben wissen, dass etwas behoben werden kann, wenn es etwas gibt, das die Performance verbessert. Wenn es Prozessfragen gibt, zum Beispiel einen großen Begriff wie jetzt die Kunden anders arbeiten und wir Anpassungen machen und den Prozess nicht als Ganzes verstehen, kann es passieren, dass ich als Task ausfasse „Ein Meeting mit den Kunden organisieren, wo man uns diesen Prozess im Quellsystem erklärt.“ Das haben wir bis jetzt zweimal gemacht und die Kunden sind sehr freigiebig mit Information.	Problembehandlung
1	Je besser wir gemeinsam das Produkt kennen, das wir entwickeln sollen, desto einfacher tun wir, das beste Produkt mit den Informationen, die wir zur Verfügung haben, zu erstellen. Dass es nicht immer das was bestellt ist, denn unsere zweite Aufgabe lautet: Die Features, die wir entwickeln sollen, möglichst gut und möglichst	Arbeitsmethode

	billig herzustellen. Das heißt gelegentlich schlagen wir Prozessänderungen vor. Es kann passieren, dass man via Userstories konzipieren und abschätzen und fragen, was passiert, wenn wir es anders machen? Es kann dann passieren, dass wir es anders machen. Wir sind alle über Jahre in diesem Projekt. Manche von uns über ein Jahrzehnt. Wir kennen den Prozess mittlerweile recht gut. Wir kennen die Kunden mittlerweile gut. Mehrere Augen sehen mehrere Möglichkeiten. Diese Änderungen der letzten Jahre waren sehr positiv. Nicht alles in DevOps ist technisch. All das was in Prozessen passieren kann ist eine Unterstützung.	
2	Ich arbeite seit 2008 professionell als Softwareentwickler im Webbereich, seit 2019 habe ich die Rolle des Frontenddevelopers übernommen und mehr oder weniger den Frontend Testbereich, also automatisiertes Testen und auch jetzt in der Automatisierung der Releases.	Erfahrung
2	Seit 2019 habe ich die Rolle des Frontenddevelopers übernommen und mehr oder weniger den Frontend Testbereich, also automatisiertes Testen und auch jetzt in der Automatisierung der Releases.	Rolle
2	Also wir verwenden Azure DevOps. DevOps ist für uns eigentlich nur das Tool der Pipeline und der Userstories. Für das Release Management direkt müssen wir das noch manuell machen, aber die Pipeline wird darüber verwaltet, sowie die Tests und das Bauen der Apps.	Verwendung
2	Nur bis zum fertigen Artefakt. Danach nicht mehr. Das ist dann über ein weiteres Portal, über das wir ein Package Request erstellen und das Deployment übernimmt dann ein anderes Team. Aber bis zu dem Punkt, wo das Artefakt fertig hochgeladen wird, wird über Azure DevOps gemanaged.	Prozess
2	Nein also allein bin ich nie. Es kommt immer mindestens ein zweiter Frontendentwickler dazu, der mich hier unterstützt, oftmals auch ein Backendentwickler und wenn es große Projekte sind, haben wir noch einen dritten Frontendentwickler dabei.	Team
2	Wir verwenden diese noch gar nicht so lange. Seit 2020 hätte ich gesagt.	Tool
2	Das hat das angefangen, dass wir das Tool für die Userstories verwendet haben und fürs Bug-Tracking und seit letzten Jahr August erst die Pipeline-Funktionen.	Tracking
2	Der Erste Kontakt war selbst nur als Gitserver. Erst später sind Bug-Tracking und Userstories hinzugekommen. Was ich noch vergessen	Verwendung

	habe zu sagen, es wird auch für manuelle Testpläne verwendet. Das haben wir schon vor der Pipeline genutzt.	
2	Genau. Vorher habe ich alles was Automatisierung betrifft immer selbst gemacht. Da war kein Budget für solche Tools.	Automatisierung
2	Ich kenne Jira, aber habe noch nicht wirklich damit gearbeitet. Ich bin mir nicht sicher, ob nicht Bitbucket ziemlich ähnlich ist, weil es ja glaub ich sogar vom gleichen Unternehmen ist.	Tool
2	Doch, also es ist eigentlich mehr oder weniger Vorgabe gewesen. Es hätte grundsätzlich die Möglichkeit gegeben, Jira zu verwenden aber nur in anderen Teams beziehungsweise in anderen Abteilungen. Das heißt für die IT war eigentlich DevOps vorgegeben. Das war der erste Punkt. Zusätzlich haben wir uns mit den anderen Teams abstimmen wollten und Vorträge gehalten haben, wodurch zu dem anderen Team, das DevOps verwaltet der Kontakt gestärkt worden und da ist dann ein gemeinsames Ziel daraus entstanden und das haben wir jetzt weitergeführt, indem wir ein Pipeline Framework erstellt haben. Das heißt also es war schon meine Entscheidung mit, aber mehr oder weniger auch Vorgabe.	Tool
2	Das ist wirklich schwer zu sagen. Das Tool ist gefühlt wie Excel, ich glaube ich kenne nicht alle Funktionen davon, aber wenn ich eine Zahl sagen müsste, von den Sachen, von denen ich einen Überblick habe, dann wären das 30 bis 40% hätte ich gesagt.	Verwendung
2	Source Code Verwaltung, Testpläne, Build Pipeline, die Stories und Issue Tracking.	Verwendung
2	Die liegen dort nur für die Buildzeit plus 30 Tage am Server und werden direkt nach dem Build auf das Artifactory hochgeladen.	Deployment
2	Kennzahlen haben wir nie aufgeschrieben. Es war bisher immer ein durcheinander, bis wir das eingeführt haben.	Metriken
2	Das Release Management ist jetzt das erste Mal in der ersten Iteration durchgeführt worden, ist also schwer da jetzt Zahlen zu erheben. Es ist auf jeden Fall das Release Management ein Prozess der anpassbar ist, weswegen wir versuchen es jedes Jahr etwas zu optimieren. Wir stehen jetzt kurz vor der zweiten Iteration. Der Plan ist mindestens ein Bugfix beziehungsweise ein Security-Patch rauszubringen pro Jahr. Das wird vorbereitet in Q2. In Q3 werden alle unsere Applikationen mit diesem Update migriert und sollten dann direkt ausgerollt werden. Wir haben dann noch in Q1 und Q4 Platz für Features für die Releases. So sieht der Plan aus.	Prozess
2	Genau. Wir haben mit Flipcharts aufgeschrieben, wie wir uns das vorstellen für die erste Iteration. Das hat teilweise gut funktioniert. Mit	Prozess

	der nächsten Iteration wird es dann eine Punktetabelle geben, die zu befüllen ist, die wir dann erweitern können. Aber grundsätzlich der Prozess ist jetzt nirgends visualisiert, sodass es ein Visio gibt. Es ist jetzt rein nur noch ein Scratch.	
2	Könnte für das gesamte Unternehmen funktionieren. Grundsätzlich ist es jetzt aber hauptsächlich für Frontendentwicklungen gedacht.	Prozess
2	Der Prozess sieht grundsätzlich so aus. Wir haben eine Planungsphase mit Features. Dann eben die Umsetzungszeit. Dann haben wir eine Testwoche. Da sind nur manuelle Tests für uns Entwickler. Darauf haben wir eine Bugfix Woche. Dann kommt das erste Deployment. Das sind hauptsächlich noch manuelle Steps. Dann gibt es eine Testwoche mit Testusern. Dann gibt es noch einmal eine Bugfix-Phase und schließlich dann das Live-Deployment. Das Live-Deployment ist mehr oder weniger semiautomatisch. Semiautomatisch, denn der Build wird automatisch getriggert und das Artefakt landet auf dem Artifactory und wir stellen dann manuell dann den Package Request für das andere Team und die übernehmen dann das Deployment. Was da jetzt dazwischen automatisch passiert ist eben der Build, die Unittests, die Integrationstests und die n-to-n Tests. Das Handover könnte irgendwann automatisiert werden aber ist jetzt aber noch nicht angedacht, denn es kann noch durchaus vorkommen, dass man nicht auf dem kompletten Userkreis ausrollen will. Das ist auf jeden Fall ein schwieriges Thema. Da haben wir als Entwickler leider nicht die Kontrolle darüber, das muss der Projektleiter dann bestimmen.	Prozess
2	Grundsätzlich hat die Entwicklung des Prozesses so stattgefunden, dass wir die repetitiven Arbeiten automatisieren wollen, die einmal wie es geschafft haben, umgesetzt zu haben. Wir haben uns einfach den Prozess auf das Tool, würde ich jetzt nicht sagen angepasst aber im Endeffekt wird schon der Weg sein, wie Azure DevOps das auch gedacht. Aber es halt einfach. Also war jetzt nicht viel großartig an unserer Arbeitsweise zu ändern, sodass es funktioniert.	Automatisierung
2	Es gibt keinen der das macht. Bei uns ist es immer so, wenn wir glauben gewissen Arbeiten benötigen kein manuelles tun und schauen, ob das irgendwie automatisiert werden kann und gehen dann auf die Suche ob und wie das in DevOps umgesetzt werden kann. Also in Grunde genommen ist es auch so die Herangehensweise, wie wir den gesamten Buildprozess entwickelt haben. Wir haben eben genau den Fall gehabt, dass wir etwas umsetzen wollten und dann geschaut haben, wie wir das in Azure	Team

	Devops umsetzten können und dann haben wir das so umgesetzt. Das haben wir dann also gemacht. Also da gibt es niemanden der da ständig schaut und versucht das auf Biegen und Brechen zu automatisieren, sondern immer dann, wenn der Bedarf da ist.	
2	Es ist auf jeden Fall so, dass das eigentlich der Projektleiter steuere, wann was raus geht. Von den Entwicklern selbst kann jeder jede Rolle übernehmen. Meistens ist es dann so, dass das Release von mir dann freigegeben wird, also die letzten Merges und die letzten manuellen Tests und übergib das den Package Request, aber es ist dann noch der Teamleiter, der dann noch dabei ist, der als eine Art Moderation zwischen Projektleiter und Entwicklern ist. Der Kunde hat eigentlich nichts zu sagen, der hat eigentlich nichts zu sagen. Der hat eigentlich nur mit dem Projektleiter Kontakt und da geht es eigentlich nur darum, was gemacht wird und wann es fertig ist.	Arbeitsmethode
2	Grundsätzlich ist es schon ziemlich gut. Vor allem wenn man es mit vorher vergleicht, wie wir entwickelt haben und auch deployed haben. Wir sparen uns jetzt sicher nur an den manuellen Arbeiten drei Wochen, hätte ich gesagt und viel Frust vor allem. Zwei Monate. Das ist jetzt eigentlich auch der Großteil. Jetzt haben wir eben Zeit für die wichtigen Sachen.	Prozess
2	Was im Prozess auf jeden Fall noch schiefläuft ist, dass wir zwar definiert haben, dass im Q3 ein Deployment zu erfolgen hat, beziehungsweise sollte durch den Prozess ein Release manuelle durchgeführt werden können, aber es zieht sich dann doch länger. Das hat aber mehr mit dem Projektleiter zu tun. Dass das Projekt oft zurückgestellt wird, jetzt vom Release her, dass das mit den manuellen Tests nicht gefunden wird. Das ist noch so eine Geschichte. Wir haben so eine Art agilen Ansatz, also teil agil, hätte ich fast gesagt, der aber nicht konsequent durchgezogen wird. Also es ist zu Teilen dann immer wieder mehr Wasserfall, dann wieder agil.	Probleme Team
2	Das ist auf jeden Fall eine gute Sache. Agil passt perfekt für das Release Management, wie wir gedacht haben, weil wir einfach viel öfters ein Artefakt rausbringen kann, was ja auch bei Scrum gefordert ist. Also es würde eigentlich Hand in Hand gehen.	Arbeitsmethode
2	Ja also es ist auch das Engagement des Projektleiters, das er das nicht forciert, dass die Sachen rausgehen. Da wäre ich mehr dahinter, dass wir schneller releasen und auch das Risiko eingehen. Das Thema Risiko hätte ich dazu genommen als Punkt im Prozess. Das Risiko ein fehlerhaftes Release rauszubringen, das traut sich	Probleme Team

	keiner eingehen.	
2	Richtig. Man könnte direkt und schnell Bugfix Releases nachschießen. Die Leute sind da noch in dem alten Muster, dass ein Release, wenn es rausgebracht wird, drei Wochen manuelle Arbeit bedeutet.	Prozess
2	Wir haben aber noch andere Probleme, die nicht nur den Prozess betreffen, sondern dann auch technische Probleme. Dass noch nicht vollautomatisier gebuildet werden kann für alle Plattformen. Wir haben da zum Beispiel mit MAC Probleme. Das ist mit Buildskripte umgesetzt worden. Das lauft nicht über die Azure Pipeline und das ist auch Problem, dass gewisse Tools auf MAC Probleme verursachen, dass nach Updated nichts mehr geht oder ein Update gemacht werden muss, dass es wieder geht. Also das sind solche unvorhersehbaren technischen Schwierigkeiten, die dort entstehen.	Probleme Tool
2	Es hat auch vor allem mit den firmeninternen Policies zu tun. Zum Beispiel, dass der MAC noch nicht in das Netzwerk gehängt werden kann, in der die Azure Pipeline eingehängt ist. Das ist eben ein Punkt.	Probleme Tool
2	Noch eine Sache, die ich dazu erwähnen möchte, ist folgendes. Wir haben zwar automatische Tests aber die Testabdeckung ist noch sehr gering. Durch die vielen manuellen Tests, die in jedem Release zwingend notwendig sind.	Automatisierung
2	Also die Prozessprobleme wie eben das Risiko und das lange Zurückhalten der Apps wird regelmäßig angesprochen, aber das scheint sich allerdings nichts akut zu bessern. Was die technischen Möglichkeiten betrifft, das versuch ich ständig etwas zu verbessern. Das ist so gesehen mittlerweile mein Steckenpferd, hätte schon fast gesagt mein Baby. Also jedes Mal, wenn ich Zeit habe etwas daran zu verbessern, dann nutz ich die natürlich.	Probleme Prozess
2	Ja also offiziell ist keiner dafür zuständig, fühlt sich auch nicht so richtig jemand, die Sachen liegen auch nur in meinen internen Notizen, was verbessert werden kann. Aber es ist angedacht, bevor die nächste Iteration startet, wie gesagt im Q2, stehen also kurz davor, dass wir danach ein Lessons Learned machen. Dann wird aus dem Lessons Learned auch offiziell eine Liste rauskommen, die wir dann abarbeiten müssen.	Problembehandlung
2	Das sind Probleme, die immer auftreten würden, unabhängig vom Tool. Wie gesagt Firmenpolicies, also es kann sein, dass es bei manchen Firmen gar nicht auftritt, weil die garkeinen Mac haben. Die	Probleme Tool

	holen sich in der Azure Cloud eine MAC Build-Machine.	
2	Es ist der größte Vorteil alles in einer zentralen Stelle zu haben. Die Testpläne, die Pipeline, die Artefakte direkt nach dem Build und so weiter. Also es bringt schon sehr viele Vorteile mit sich. Ein Tool, das wieder auf mehrere einzelnen Tools aufgesplittet ist, wäre wieder ein Nachteil.	Verwendung
2	Und wie gesagt, bis jetzt hat es immer sehr gut gepasst. Wir haben uns damit gespielt, bis es funktioniert hat, aber das mit dem Spiel bezieht sich nicht auf den Prozess, sondern mehr auf die Umsetzung, denn man hat schon eine steile Lernkurve dabei. Das sind da einige Notationen für die Pipelines und auch die Rechteverwaltung war nicht sehr einfach.	Arbeitsmethode
2	Ja also das sind mittlerweile aus. Wir haben unser Q2 Update und ein Migrationsheft, das in Mark down geschrieben ist und wird dort abgelegt. Und auch andere Sachen werden mittlerweile als PlantUML heißt das, so eine Markdown-Schrift, die dann einen Prozess visualisieren kann, dafür haben wir dann einen eigenen Container, der uns da einen Build rendern kann. Also das liegt dann jetzt alles im Repository drinnen und ich würde mir wünschen, dass die gesamte Dokumentation zum Repository reinbringen und nicht separat im OneNote niederschreiben.	Verwendung
3	Also Berufserfahrung habe ich seit 2011 also mittlerweile fast 11 Jahre, in denen ich eigentlich durchgängig in der IT-Branche arbeite.	Erfahrung
3	Begonnen habe ich als Systems Engineer, bin dann in das Consulting von Datenbank Management Systemen gewechselt, im speziellen Microsoft SQL-Server und habe dann in die Bankenbranche gewechselt und war dann in der IT-Organisation tätig und habe dann intern in weitere Folge gewechselt in die Konzernorganisation und seit ca. einem Jahr bin ich in einem sehr großen Softwareunternehmen tätig mit über 3000 Mitarbeitern und ca. 1000 Entwicklern und bin dort in der Rolle des Jira- und Confluence-Experten.	Rolle
3	Jira und Confluence.	Tool
3	Mit Azure DevOps habe ich persönlich keine Erfahrungen.	Tool
3	Im Schnitt sind sieben Personen in einem Team. Diese Zahl ändert sich von Team zu Team, wobei ich glaube, sind es meistens zwischen fünf und sieben Personen.	Team
3	Ja es sind auf jeden Fall agile Ansätze, die verfolgt werden. Primär wird nach Scrum gearbeitet, wobei vereinzelt Teams auch nach	Arbeitsmethode

	Kanban arbeiten.	
3	Absolut. Das ist eines der zentralen Dinge in unserem Unternehmen mit expliziter Ownership für das Tool eben.	Verwendung
3	Ich persönlich habe Jira in den letzten beiden Unternehmen, in denen ich tätig war, mehr oder weniger eingeführt und ausgerollt.	Erfahrung
3	Es ist unterschiedlich. Also im ersten Unternehmen, in dem ich tätig war, das war im IT-Systemhouse, da ist Jira aufgrund wahrscheinlich der IT-Affinität würde ich behaupten sehr gut angenommen worden, da war die Akzeptanz schnell da. Es ist also im ersten Schritt wirklich nur Jira und mit Kanban die Arbeitsweise etwas nähergebracht worden und in weiterer Folge das Scrum Framework stark abgeändert und Confluence in einer nächsten Iteration oder Phase eingeführt worden.	Benutzerfreundlichkeit
3	In der Bankenbranche, wo ich auch gearbeitet habe, war Jira eigentlich gehostet, um unsere Rechenzentren zu betreiben und da haben unsere Mitarbeiter mehr oder weniger geschult werden müssen. Da war die Akzeptanz mit Sicherheit niedriger und auch der Funktionalitätsumfang, den wir genutzt haben, wesentlich geringer.	Benutzerfreundlichkeit
3	Das heißt der Einstieg war auch auf einem niedrigeren Level und je mehr man mit dem Tool vertraut war umso mehr wurde auch die Funktionalität am Tool dann auch genutzt.	Erfahrung
3	Exakt, also in beiden Unternehmen, wo wir es ausgerollt haben. In dem jetzigen Unternehmen war es schon existent.	Rolle
3	Das Wissen habe ich mir damals selbst über Selbststudium, Recherche, Videos und Unterlagen selbst beigebracht und hab im ersten Unternehmen die Möglichkeit und auch die Kapazität bekommen, dass ich das selbst von der grünen Wiese auch starte, selbst hoste, hochziehe und entsprechend mich da selbst einlese und das weitertreibe, sowie das Wissen entsprechend weitervermittle.	Erfahrung
3	Früher hat man auch Jira Service Desk verwendet, von dem hat man sich dann aber getrennt. Bitbucket ist derzeit auch in Verwendung für diverse individuale Entwicklungen, auch self-hosted. Neben Confluence ist das eben unsere Atlassian Suite.	Tool
3	Also der Automatisierungsgrad ist bei uns bereits sehr hoch.	Automatisierung
3	Bei uns ist es so, dass wir allesamt an einem einzigen Produkt arbeiten. Das heißt wir individualisieren keine Lösungen und bauen auch keine Individuallösungen für Kunden, sondern wir entwickeln zentral ein Produkt weiter. Das ist groß würde ich sagen mit zig Modulen und über alle Module hinweg herrscht der gleiche Prozess.	Team

3	Von der Funktionalität von Jira? Ich bin mir in der Frage nicht klar, welcher Prozess in Jira speziell? Von der Bearbeitung der Tickets oder in weiterer Folge schon in Richtung der Releases? Hier würde ich sagen ist er schon sehr hoch, also bei 80% ca.	Verwendung
3	Ein Teil davon ist das Kreieren von Release Nodes, die Darstellung dementsprechend von Release Nodes und die Auflistung.	Automatisierung
3	Ja automatisierte Tests sind auch bei uns ein absolut großer Punkt.	Automatisierung
3	Ja verwenden wir. Ich weiß aber im Detail nicht welche und kann da auch keine genauen Zahlen dazu nennen.	Metriken
3	Also ich persönlich ausgehend von einer 40-Stunden-Woche behaupte ich, dass es 35 Stunden sind zuzüglich aller Meetings ist mein gesamter Handlungsspielraum in der Automatisierung und Weiterentwicklung von Jira.	Verwendung
3	Ja also es gibt Individuallösungen, die umgebaut worden sind.	Add-On
3	Ja es war sogar eine sehr starke Veränderung da, weil eben das Unternehmen wahnsinnig stark gewachsen ist, dementsprechend ist auch der Prozess weiterentwickelt worden. Es ist auch laufend eine Weiterentwicklung. Die jetzige Version hat sich erst nach mehreren Iterationen finden müssen. Es ist vor allem auf Detailebene sehr viele dazugekommen und das Tool rundherum ist auch Stück für Stück weitergebaut und wird auch stetig weiterentwickelt. Es ist also schon sehr stark der Gedanke da, den Prozess nicht an das Tool anzupassen, sondern dementsprechend Ergänzungen für den Prozess entwickelt und auch ständig weiterentwickelt, sodass dieser den Prozess unterstützt.	Prozess
3	Ich glaube er ist sehr individuell in dem Fall.	Add-On
3	Das heißt wirklich, wenn Jira als Tool eine neue Version ausrollt, inwieweit man diese neuen Funktionen in den derzeitigen Prozess integriert? Dafür gibt es auch eine eigene Rolle beziehungsweise ein Core-Team oder auch Administrationsteam, dass diese Release Nodes in dem Fall von Atlassian bespricht und analysiert.	Team
3	Also die komplette Research and Development ist in Jira abgebildet.	Rolle
3	Wo ich ein Problem mit Jira im Release Management sehe, ist, dass Versionen eigentlich nur auf Projektebene verwaltet werden können. Das heißt es gibt keine projektübergreifenden Versionen in Jira. Das müssen wir beispielsweise mit einem zusätzlichen Plug-In, das wir dazu gebaut haben, syncen. Das gescheduled dafür sorgt, dass es überall dieselben Versionen gibt, wenn wir dann nur in einem einzigen Entwicklungsprojekt arbeiten, sondern dann auch verteilt wird auf zig Projekte und dann wird dort sichergestellt, dass	Probleme Tool

	zumindest namentlich die gleichen Versionen existieren, die dann in weiterer Folge als restricted Version verwendet wird. Das sehe ich als kleines Hindernis, dass diese nicht zentral an einer Stelle verwaltet werden können. Ansonsten dient es als Basis grundsätzlich sehr gut.	
3	Ab einer gewissen Unternehmensgröße, also ab mittelständigen Unternehmen definitiv ja.	Add-On
4	Also ich habe vor fast acht Jahren begonnen.	Erfahrung
4	Ich habe begonnen bei Sontorin/TFMS als Produktentwickler, wie es damals noch geheißen hat Integration Engineer. Heute sagt man eher DevOps Engineer dazu. Dort war ich für drei Jahre insgesamt tätig. Danach habe ich in das Toolservice gewechselt. Ich habe dann bei ITS Toolservice Azure DevOps komplett administriert und war betriebsverantwortlich für Jira, Confluence, Artifactory, Sonarcube und alles was so herum gebaut worden. Das habe ich drei Jahre gemacht und dann bin ich zu der IT gekommen, wo ich seit zwei Jahre ungefähr bin, und bin dort weiterhin verantwortlich für Azure DevOps global und für das Taskservice, in dem Fall Kubernetes as a Service, was man für Softwareentwicklung braucht, für containerisierte Applikationen und dort sind wir mittlerweile mit einem Viererteam.	Rolle
4	Also alles was Issue-Tracking, Release Management und Bug-Tracking generell ist Jira vom Tool her, schon wesentlich besser als Azure DevOps. Sourcecode natürlich nur Azure DevOps und für Sourcecode und CICD würde ich jetzt nicht auf die Idee kommen, Bitbucket oder Bamboo verwenden, was von Atlassian kommt. Auch deswegen geschuldet, dass auf der Issue-Tracking- und Release Management Seite Jira schon immer gibt bzw. sehr lange gibt in der AVL, eine riesige Community hat und manifestiert ist.	Tracking
4	Auf der anderen Seite Azure DevOps für Sourcecode und CICD schon seit über zehn Jahre manifestiert ist und auch dort schon eine Community vorhanden ist. Ich glaube auch, dass es für Issue-Tracking eine Gruppe gibt, die sich drum kümmert und sich damit beschäftigt, die sich dort auch gut auskennen und Azure DevOps Workitems, dort kocht ein jeder sein eigenes Süppchen. Hat also beides seine Vor- und Nachteile.	Tool
4	Das war von Anfang an weg da. Also es bei Sontorin/TFMS habe ich Bug-Tracking in Jira gemacht und wie mein erster Zugang dazu war, weiß ich jetzt nicht.	Erfahrung

4	Man hat es gezeigt bekommen und für mich ist es sehr benutzerfreundlich. Wenn man es einmal überblickt hat, ist es schon sehr benutzerfreundlich. Vor allem was der Vorteil bei Jira ist, dass wir teamübergreifend und standortübergreifend an mehreren Produkten gearbeitet haben, teilweise nicht zwingend Softwareprodukte und auch andere Projekte. Da lasst sich in Jira einfacher abbilden als in Azure DevOps.	Benutzerfreundlichkeit
4	Also als ich zur IT gekommen bin, habe ich den DevOps-Server der e1 übernommen und dann haben wir begonnen sukzessive als dann fix wurde, dass das Toolservice zur IT kommt, dass dann der Devops-Server der e1 mit dem DevOps-Server der AVL zusammengeführt wird, was wir jetzt heuer Anfang Jänner gemacht haben. Und das war jetzt auch meine Hauptaufgabe bei im letzten Jahr.	Rolle
4	Ich war damals bei Sontorin/TFMS als Integration Engineer war das damals auch meine Aufgabe, dass wir Releases rausbringen vom Entwicklungsteam.	Rolle
4	Jetzt bei uns im DevOps-Team haben wir so gesehen auch unser Produkt, das IT-Portal, wo wir auch so eine Art Release Management machen, aber da haben wir den wunderbaren großen Vorteil, dass wir das als Software-as-a-Service betreiben können, sodass wir jetzt keine Software an den Kunden ausliefern müssen, die bei ihm lauffähig sein muss.	Arbeitsmethode
4	Ja wir haben schon Kundendruck aber unsere Kunden sind In-House also unsere IT-Arbeiter sind unsere Kunden und deswegen muss das IT-Portal doch eigentlich immer gehen. Das muss auch gepatcht werden und es muss auch funktionieren also ein vereinfachtes Release Management machen wir ja trotzdem.	Arbeitsmethode
4	Jein. Also wir im Team aber auch generell in der IT verwenden wir das Issue-Tracking in DevOps selbst. Als ich zur IT gekommen bin, da haben wir die Entscheidung getroffen, ob wir Jira verwenden zur Arbeitsplanung oder ob wir im DevOps Server gehen. Da ist jedoch entschieden worden, dass wir als IT-Team schon in DevOps reingehen sollen, weil alle IT-Mitarbeiter es verwenden.	Tracking
4	Das war schon eine Entscheidung damals und wir haben dort dann also unsere Epics, Features und User Stories. Die planen wir dort und sind auch verknüpft mit den Releases.	Planung
4	Wir arbeiten als Scrum Team. Wir haben vier Wochen Sprints und wir versuchen so gut wie möglich zu planen und arbeiten so unsere Arbeitspakete ab. Da viele von uns zusätzlich auch noch operativ	Arbeitsmethode

	Service betreiben ist dann oft eine Weiterentwicklung zweitrangig im Vergleich zu welchen operativen Tasks.	
4	Also, wenn ich sage, ich bin als einfaches Entwicklungsteam mit einer Handvoll Entwicklern, die an einem Produkt arbeiten, dann ist Azure DevOps super. Du hast alles in einem Bus. Du hast deine Workitems, deinen Sourcecode und du kannst alles wunderbar miteinander verlinken. Nur sobald du anfängst mit mehreren Teams an mehreren Produkten und du möchtest ein Portfolio Management haben, dann funktioniert das so wie wir es gerade verwenden nicht. Da hätten wir es von Anfang an anders machen müssen, damit es funktionieren würde.	Probleme Tool
4	Also niedergeschrieben beziehungsweise ein Dokument haben wir nicht. Wir haben es in unsere Automatismen manifestiert, dass man es so machen muss und es gar nicht anders geht.	Prozess
4	Bei uns ist alles automatisiert. Wenn man das IT-Portal ansieht, dort gibt es keinen manuellen Angriff außer eben Sourcecode schreiben und Pullrequest absegnen.	Automatisierung
4	Wir updaten den DevOps-Server regelmäßig. Es kommen auch sehr selten Updates, also wirklich neue Features kommen ein- oder zweimal im Jahr. Das ist sehr schleppend aber manchmal kommen auch sehr viele auf einmal.	Tool
4	Ja also wir sind vier Engineers und dann haben wir noch unseren Teamleiter und Enterprise Architekt, der uns die Vision vorgibt. Zusätzlich haben wir noch einen Scrum Master. Es sind jedoch nicht nur wir vier, sondern mittlerweile 15 Leute bei unserem Teamleiter und Scrum war ihm zu zeitaufwändig, da er die anderen Teams auch noch hat, weswegen noch eine Scrum Masterin hinzugefügt wurde.	Team
4	Ja genau. Also wenn es um Controlling geht oder um Fortschritt geht, dann ist alles in DevOps die Wahrheit. Da ist es sogar so, dass wir „Prügel“ bekommen, wenn wir sagen, dass wir dies und jenes zusätzlich gemacht haben, aber noch nicht eingetragen haben.	Tracking
4	Wir gehen eher den Weg, dass wir uns an das Tool anpassen als wir passen das Tool an, weil wir vier Personen sind. Also in DevOps gibt es Epics, Features, User Stories und Tasks und eben die Bugs, das gibt es als Default immer und wir haben unsere Arbeit so festgelegt, dass wir damit auskommen.	Arbeitsmethode
4	Vor allem auch weil wir glauben, dass es für uns weniger Aufwand ist, uns an das Tool anzupassen, anstatt das Tool auf uns anzupassen, wobei wir damals nicht wirklich welche gehabt haben. Auch wenn man so viel anpasst, muss man bei jedem Update	Prozess

	kontrollieren, ob es danach auch noch funktioniert. Und diese Arbeit haben wir gesagt, tun wir uns nicht und lassen das Tool so nahe am Standard wie möglich. Also bezüglich Features, die uns fehlen würden, habe ich keine Informationen dazu.	
4	Es ist so, dass wir vier Personen sind und wir alle kennen uns gut aus. Wir haben jetzt nicht den Bedarf, dass wir jetzt sagen, bestimmte Aktionen dürfen nur von bestimmten Personen ausgeführt werden, weil die nicht wissen, was sie tun.	Team
4	Was man dann schon sagen muss, dass wenn ich an meine Produktentwicklungszeit zurückdenke, da 20 Personen in Graz, 40 Personen in Wien, da haben 60 – 70 Leute an einem Projekt gearbeitet und dort muss man schon auf vieles aufpassen. Ein Beispiel ist das Berechtigungskonzept. Nicht jeder Tester darf einen Bug anlegen, denn in 90% der Fälle ist der Bug sinnlos, weil er etwas vergessen hat. Da muss es jemanden geben, der noch einmal drüber schauen muss.	Probleme Prozess
4	Wir haben den Luxus, dass wir so klein sind und uns auch gut auskennen, sodass wir wissen was wir tun. Deswegen haben wir bewusst alles am Standard gelassen, dass wir uns keine Stolpersteine einfangen können und eben den Automatismus so weit entwickelt, dass das immer funktioniert. Wenn wir wollen, müssen wir selbst uns einen Stein legen. Zum Beispiel dass man in den master Branch, so nicht einfach commiten kann, weil das sonst später zu Problemen führt oder man muss eine User Story beim Pull Request angeben, damit man weiß, welchen Ursprung diese Codezeile hat und welches Feature oder User Story dahinter liegt beziehungsweise, was ist das überhaupt? Solche Sachen haben wir durch den Automatismus erzwungen.	Prozess
4	Ja. Wenn du das so ausdrückst, dass sie weiß, was wir tun ist es für sie sehr hilfreich. Einerseits sollen wir das machen und andererseits müssen wir das auch machen. Eben um auch den Management Sachen rechtfertigen zu können. Warum haben wir länger gebraucht für eine User Story oder warum haben wir eine User Story nicht fertig geschafft, weil wir auf der anderen Seite so viel ungeplante Tasks gehabt haben.	Tracking
4	Also wir haben jeden Tag 15 Minuten Stand-Up. Dann haben wir einmal in vier Wochen ein Tag Sprint Planung. Das wären dann nur in der Woche im Tool fünf Stunden. Also schon einen halben Tag.	Kommunikation
4	Wenn wir die gesamte Planung mit einberechnen, findet doch sehr viel im Tool statt. Sprint Start, Sprint Ende, Sprint Retro, Sprint	Verwendung

	Review machen wir alles dort drinnen. Das ist auf den Monat gesehen schon viel Zeit. Das ist dann auf die Woche heruntergebrochen schon 5 Stunden. Also ich weiß in Jira habe ich sicher mehr Zeit drin verbracht.	
4	Weil über Jira haben wir die ganzen Bugs abgehandelt. Wenn wir jetzt sagen, dass wir ein Release an das Testteam übergeben haben und das Testteam hat dann Bugs gefunden. Dann ist es immer wieder zu uns zurückgekommen, wir haben uns die Kommentare durchgelesen, was erweitern und im weiteren Schritt an das Testteam wieder übergeben, da kommt dann wieder etwas zurück oder auch wenn von einem anderen Produkt für Kundenservice Fehler auftreten, kommt das zu uns rein. Dann wird das alles in Jira abgehandelt. Da hätte ich gesagt, da sind dann am Tag mindestens 2 Stunden, die man als Integration Engineer in Jira verbringt.	Verwendung
4	Also bei mir im konkreten Fall im Vergleich zum Toolservice Team und jetzt IT ist es sicher mehr geworden, weil wir im Toolservice Team praktisch keine Planung gemacht und im DevOps Team machen wir jetzt die Planung selbst in DevOps, also ist es hier schon definitiv mehr geworden. Aber es waren auch andere Arbeitsweisen. Auch kurzfristig gesehen ist es auch seitdem ich in der IT bin mehr geworden, weil wir hier immer mehr Fokus auf eine saubere und vernünftige Planung legen.	Verwendung
4	Ja also wir haben schlecht angefangen, dann ist es super geworden und dann mit Corona haben wir es schleifen lassen. Dann ist die Verwendung nach unten gegangen. Weil wir uns nicht mehr getroffen haben und wir nicht mehr physisch darüber gesprochen haben, ist es nach unten gesackt und es war nicht mehr gut. Da haben wir gesagt wir wollen wieder den alten Prozess und wir hätten gerne jemanden, der sich drum kümmert und daraufhin ist ein Scrum Master gekommen. Es ist dann wieder besser geworden, zwar nicht von einem Tag auf den anderen und es hat auch eine Zeit gebraucht, bis wir dann das alte Planungschaos losgeworden sind und die Altlasten entfernt worden sind und seitdem ist es gleichbleibend gut.	Probleme Team
4	Jetzt da wir auch jemanden haben, der täglich oder wöchentlich darauf schaut und den Status der Projekte analysiert und erkennt, wieso dort nichts weitergeht oder nichts gemacht wird oder auch Tasks sind, die eine Woche den Status „In Work“ haben, dann wird dort nachgefragt was dort los ist, woraufhin man erkennen kann, dass der Task vielleicht falsch geschrieben wurde.	Tracking

5	Ja, alles klar. Ich habe Informatik studiert an der TU Wien und habe während des Studiums bald zu arbeiten angefangen für ein Start-Up in Wien. Start-up wurde aufgekauft von größerer Firma. Als ich mit Studium fertig war, habe ich mich gemeinsam mit einem Arbeitskollegen selbstständig gemacht und haben zu fünft insgesamt ein Add-On für Jira entwickelt.	Erfahrung
5	Das hieß Roadmaps für Jira. Das war damals noch ein reines Server Add-On. Jira hat Jira Cloud, Jira Server, Jira Data Center und Data Center hat es damals noch nicht gegeben, Jira Cloud war gerade im Kommen. Also es war ein Add-On für Server.	Tool
5	Und das Ziel vom Add-On war es die Planung über einen längeren Zeitraum zu ermöglichen. Jira Software war immer sehr gut darin, diese kürzeren Iterationen zu planen, was ist im nächsten Sprint, was ist in den nächsten zwei bis drei Sprints. Wenn man eine Roadmap für die nächsten Monate oder Jahre, gabs noch kein passendes Werkzeug und wir haben versucht sowas zu bauen und sind dann etwa nach einem Jahr von Atlassian aufgekauft worden.	Planung
5	Meine Rolle war die eines Engineering Managers, also in der Engineering Organisation in einer Management Rolle und habe zunächst am Add-On weitergearbeitet. Zunächst wurde es regebrandet auf Portfolio von Jira und mittlerweile heißts tatsächlich wieder Advanced Roadmaps. Es gibt es also immer noch, schaut inzwischen, nach einigen Jahren, nur etwas anders aus. Und habe nach einiger Zeit vom Add on ins eigentliche Jira Team gewechselt.	Rolle
5	Es ist so, dass innerhalb von Atlassian verwendet jeder die Attlassian Tools. Confluence, Attlassian selbst betreibt die größte Confluence Instanz, die es überhaupt gibt. Die wird als Intranet Wiki genutzt. Es gibt zwei oder drei große Jira Instanzen, die gehören auch zu den größten überhaupt. Natürlich auch Bitbucket und Hipchat wenn es des noch geben hat beziehungsweise der Nachfolger Stride. Trello wird natürlich auch verwendet.	Tool
5	Zum Thema Release Management würde ich sagen, dass bei Atlassian und ich glaub beim Web Engineering grundsätzlich man versucht wegzugehen von den Heavyweight Release Prozess.	Prozess
5	Vor allem bei einem Tool wie Attlassian, das sehr viele externe Applikationen hat, ist es natürlich immer ein großes Drama, wenn man Apis bricht, im Web Engineering. Im Cloud Engineering versucht man weg davon zu gehen.	Probleme Prozess

5	Ich glaub es hat gar nicht so viel mit dem Entwicklungsprozess zu tun. Ich glaub der Entwicklungsprozess ist mit der ganzen Planung sehr schwer. Ich kann mir nicht mehr vorstellen, wie man nicht agile Software entwickeln kann. Ich weiß früher hat man das gemacht, ja. Für mich ist das heutzutage unvorstellbar. Mein gesamtes professionale Leben hat in Firmen stattgefunden wo Scrum impliziert wird oder von mir aus Kanban. Und das dieses Iterative, Zyklische das gilt sowohl – das hat mit den Releases nicht so viel zu tun find ich. Du kannst sagen du hast einen sehr definierten Release Prozess und dann agil machen oder du hast zurück zu dem wie sie in Jira Cloud oder Attlassian Cloud funktioniert, dass man einfach versucht, dauernd zu releasen.	Arbeitsmethode
5	Also wir waren sofort an einem Punkt, wo wir jeden Merge zum Master released haben. Und das ist ja der Idealzustand. Denn die Philosophie dahinter ist in Wirklichkeit ja die, wenn du sehr häufig sehr kleine Sachen released, sehr kleine Changes, dann ist das Risiko mit jedem einzelnen Change viel kleiner, weil ja nicht besonders viel Änderungen reinkommen.	Deployment
5	Zum Beispiel ein Aspekt den Atlassian ganz intensiv praktiziert ist, dass man durch verschiedene Environments den code durchschreibt. Du hast jetzt mal in Wirklichkeit drei Layer, eine Entwicklungsumgebung, da ist die Schiene gemeint, dass die Engineering Organisation selbst verwendet. Wenn du Codeänderung pushst, springt der Build Prozess an und die neue Version von Jira wird automatisch in Entwicklungsumgebung deployed. Und damit hat die gesamte Engineering Org. die gesamte Änderung als Allererstes. Ich weiß nicht was da die Cycle time ist, vielleicht 20 Minuten zwischen den Moment, wo du den code commitest, bis zum Punkt, wo die neue Version da ist.	Deployment
5	Production ist auch noch unterteilt, da gibt's das Konzept oder die Idee von Kanaree-Kunden, wo man sagt, das kommt vom Kanarienvogel die Idee ist, man hat eine Handvoll Kunden, die diesen Code vor allen anderen kriegt und die Idee ist, dass, falls irgendwas schief geht, falls ein Bug drinnen ist, dann ist der Blast-Radius mal kleiner.	Deployment
5	In Wirklichkeit ist das viel Komplexität und relativ viel Engineering dahinter, hinter den tatsächlichen release Prozess. Wie wandert Code jetzt zum Kunden, welche Stufen geht der durch, welche Checks passieren da. Da gibt es eigene Teams, die sich drum kümmern. Ich habe eine Organisation geleitet, die für den Prozess	Prozess

	des Frontends verantwortlich ist und das eine hochkomplexe Angelegenheit, keine Frage.	
5	Bei Attlassian und vielen anderen Softwarefirmen, gibt es diese Jobbezeichnung einfach nicht mehr. Das war bei Attlassian bisschen ein Kulturwechsel. Als ich begonnen habe, gabs sehr wohl noch Tester. Aber keine klassischen mehr, die irgendeine Version testen, sondern in Teams integriert und deren Aufgabe war eigentlich eine gewisse Awareness zu schaffen und Kultur für Softwarequalität zu erzeugen.	Team
5	Die Idee ist die, dass in Wirklichkeit jeder Engineered für die Qualität verantwortlich ist. Das was man früher sah, dass der Engineer seinen Code macht und was falsch macht und jemand anderes sich darum kümmert. Von den wollen wir massiv weg. Das ist die ganze Idee von Dev auf Stage. Development kümmert sich auch, übernimmt Verantwortung für Operation. Develop und Operation kommen zusammen und ergeben DevOps.	Arbeitsmethode
5	Und das ist natürlich was, wo man bei Attlassian sehr viel Wert legte und ein großes Thema war. Sprich manuelle Tests gibt nicht mehr, es gibt keinen, der das manuell testet. Natürlich, während du den Code schreibst, wirst du schon deine innere Entwicklungsschleife vielleicht irgendwie ausprobieren. Aber das Ziel ist eher, dass du eine gesunde Testpyramide hast	Automatisierung
5	du hast verschiedene Kategorien von Tests und idealerweise ist das ganze pyramidenförmig aufgebaut. Je näher man am Code ist, desto mehr Tests hat man. Sprich man hat sehr viele Unittests. Warum? Weil billig auszuführen, laufen sehr schnell, sehr reliable auszuführen, weil es keinen Grund gibt, warum es am Monitor fleckig sein soll – manchmal grün, manchmal rot – der ist entweder grün oder rot und soll nicht fehlschlagen.	Automatisierung
5	Da geht man heutzutage eher in die Richtung zu Post-Deployment-Verification. Man deployed etwas, das geht raus und dann führt man gewisse Tests gegen die Production-Instanz aus. Das macht man regelmäßig und führt alle fünf bis zehn Minuten automatisierte gegen Production aus. Die Idee ist, dass man sagt, dass es weniger darum geht zu verhindern, dass man einen Fehler im Production shippt. Aber man sieht ein, dass es nicht absolut vermeidbar ist und das Ziel ist, dass man das möglichst schnell in Production erkennt und möglichst schnell Fehler rückgängig macht, indem man zurückrollt.	Deployment

5	Für die automatisierten Tests meines Wissens nicht wirklich. Da haben wir etwas selbst geschnitzt, eine eigene Lösung. Aber Bitbucket Pipelines spielen in den gesamten Deploymentprozess eine große Rolle. Dieser Weg von „du als Entwickler commitest deine Änderung“ bis hin zu „deine Änderung ist bei dem Kunden“ ist ein komplexer Prozess. Dort werden wahnsinnig viele Stufen durchgespielt und der ist über Bitbucket Pipelines abgebildet ist. Wobei man fairerweise dazusagen muss, dass in Jira, wo noch viel basierend auf Bamboo, was das Vorgängertool von Bitbucket gewesen ist, aber die Idee ist die gleiche.	Automatisierung
5	Ich habe persönlich nicht wirklich Erfahrungen mit den Microsoft Stack. Bei Atlassian verwenden wir unsere eigenen Tools. Github ist ja glaube ich im Microsoft Stack eingebunden und Github verwende ich nun in meinen Start-up, bin natürlich mit GitHub vertraut aber meiner Meinung nach ist es Geschmackssache, ob du das eine oder das andere verwendest. Github ist im Vergleich zu Bitbucket wahrscheinlich das etwas ausgereiftere Tool.	Tool
5	Contentdeployment ist definitiv das Ziel und ich glaube in diese Richtung wird jede Firma früher oder später gehen, ist einfach der bessere Ansatz.	Deployment
5	Die Rolle des Projektleiters: ich glaube nicht, dass sich diese überhaupt ändert. Es gibt weiterhin Zuständigkeiten. Es gibt auch weiterhin Produktverantwortliche und Featureverantwortliche.	Rolle
5	Dieses Team besteht aus einer Reihe von Engineers, einen Team-Lead, der die Management-Verantwortung für das Engineering hat, dann gibt es einen Produktmanager, der die Verantwortung des Produktes trägt und Fragen klärt wie „Verstehe ich den Kunden“ oder „Verstehe ich die Kundenanforderungen“ und macht auch Customer Research, schaut sich die Analyse und Statistiken an und analysiert, ob das Feature überhaupt Sinn ergibt oder ob man das weiter umbauen muss. Die Rolle ist also produktverantwortlich und da gibt es dann noch die Designer, die die Visuals erstellt und die Userinteractions erstellt.	Team
5	Was weiterhin eine große Herausforderung ist, ist die Kommunikation, weil es dort weiterhin einen großen Abhängigkeitsgraphen gibt, denn mein Team ist verantwortlich für die Navigation. Aber dann gibt es ein anderes Team, das einen zusätzlichen Eintrag in der Navigation haben will. Das muss dann wiederum organisiert und orchestriert werden. Wie sind dort die Abhängigkeiten? Wie ist dort der Zeitplan? Dann muss besprochen	Probleme Team

	werden, welche Vorbereitungen getroffen werden müssen und wann diese fertig sein müssen. Da gibt es im Jira Team laufend Planungsaktivitäten, wo dann verschiedene Teams zusammenkommen und man einfach versucht zumindest für die nähere Zukunft einen Plan zu erstellen.	
6	Meine Position ist Tech-Lead.	Rolle
6	Ich manage seit 2018 im Unternehmen 27 Mitarbeiter in der R&D.	Team
6	Berufserfahrung im IT-Bereich habe ich 20 Jahre und hauptsächlich im eCommerce.	Erfahrung
6	Für das Release Management an sich machen wir unser Deployment über Bamboo.	Tool
6	Wir haben ein eigenes Tool geschrieben, um die Umgebung auch zu betreiben.	Add-On
6	In Gitlab laufen unsere Pipelines.	Automatisierung
6	Aus den Atlassian Stack verwenden wir Jira für das Issue Management	Tracking
6	Dort verwenden wir auch die Release Möglichkeiten.	Tool
6	Confluence ist unser Firmen WIKI, wo wir unsere Dokumentationen machen.	Tool
6	Q&A Abteilung verwendet XRAY, ein Jira Plugin.	Add-On
6	Wir haben zwei eigene Deploymentskripte und haben um diese ein Tooling gebaut.	Deployment
6	Es war sehr einfach Jira und Confluence zu verstehen. Auch Gitlab ist einfach zu intergrieren gewesen.	Benutzerfreundlichkeit
6	Wir haben definitiv während dem Doing Sachen gelernt und dementsprechend inkrementell ständig versucht das Release Management zu verbessern.	Prozess
6	Dann hat man mit Jira und den Release Labels, sowie Confluence-Komponenten verwendet, um eine bessere Sicht auf Releases zu kriegen.	Verwendung
6	Es gibt nicht nur ein Produkt, an dem gearbeitet wird.	Arbeitsmethode
6	Also was wir schon haben ist im Release Prozess über die Q&A also das Testmanagement eine Aussage, wie schnell der erste Durchlauf ist und mit den Testerfolgen, die wir haben, ist für jeden Release in Jira einen Feedback Epic, wo dann der Kunde Issues einstellt, die bezogen auf den Release auftreten und aufgrund der Anzahl an Issues hat man schon relativ einfach eine Kennzahl, wie gut ein Release ist.	Metriken

6	Ich denke schon, dass es wir 70% vom Tooling ausnutzen. Wir gehen schon in die Breite und nicht in die Tiefe. Wir verwenden also alles wie Kanban Boards und unterschiedlichen Dashboards. Wir verwenden was notwendig ist, um unseren Prozess zu unterstützen. Also wir haben schon so unser Tooling im Einsatz, wie wir denken, dass es effizient für uns läuft.	Verwendung
6	Also natürlich der Build- und Deploymentprozess ist automatisiert.	Automatisierung
6	Wir sind jetzt schon seit knapp 2 Jahren dabei, die Testautomatisierung voranzutreiben also das ist unterschiedlich, je nach Team und Plattform-Produkt ist entweder Side-Pressing im Einsatz oder Karate für Api-Tests. Für das reine Testcase Management verwenden wir XRay. Das sind die unterschiedlichen Tools, die wir im Einsatz haben, um einen höheren Grad an Automatisierung zu haben.	Automatisierung
6	Ja das Monitoring und Alerting sowie Postdeployment-Aktivitäten können wir auch mit unseren Tools erweitern und automatisieren, wenn es notwendig ist.	Automatisierung
6	Der Release Prozess ist generisch und passt auf alle Produkte. Wir sind jetzt auch gerade dran, den Prozess so weit anzupassen, dass dieser auch für andere Teams anwendbar ist.	Prozess
6	Ich denke der Prozess selbst ist definitiv nichts Neues und kann auch in anderen Unternehmen verwendet werden. Es gibt natürlich gewisse Feinarten, die bei uns etwas spezieller sind wie Rollenbeschreibungen. Was sicher unterschiedlich ist, sind die Artefakten. Also diese sehen in den eCommerce Unternehmen sicher immer anders aus aber der Prozess selbst kann verwendet werden.	Prozess
6	Wenn wir neue Funktionalitäten brauchen, suchen wir die auch aktiv. Also am Beispiel Testmanagement, wenn wir sagen wir brauchen eine Unterstützung für die Testmanagementverwaltung in Jira und haben uns selbst in der Entwicklung eine Analyse und einen Proof of Concept erstell und uns dort geeinigt. Also wenn dann passiert das innerhalb des Entwicklungsprozesses, wo wir das Tool dann aussuchen.	Tool
6	Das wird aufgeteilt nach Bedarf und Notwendigkeit. Ansonst rein im IT-Ops-Bereich gibt es natürlich die ITIL-Leute, die sich für die Jira und Confluence Sparte kümmern. Das heißt jedoch nicht, dass die zuständig sind, neue Features in die Firma zu bringen.	Rolle
6	Also wir folgen jetzt den Kanban-Prozess, wo wir vordefinieren, wie groß der Umfang sein soll. Haben ein bisschen Scrum dabei. Über Kanban wird eine gut definierte Feature-Entwicklung abgebildet, weil	Arbeitsmethode

	es sehr abgrenzbar ist. Den regulären Sprint-Prozess verwenden wir für unsere Supportsprints. Man sagt dort je nachdem, wieviel Support Issues kann ich in der Woche einplanen und arbeiten diese in den Sprints ab. Wenn man Issues nicht schafft, nimmt man die dann einfach in den nächsten Support Sprint mit.	
6	Die Rollen sind vielfältig. Wir haben dort die Entwickler drinnen. Wir haben den Endkunden, der Abnehmer, der Software ist. Wir haben den Support drin und den Release Manager, Projektmanager, Quality Assurance. Die Rollen sind auch klar definiert im Prozess.	Rolle
6	Rein technisch ist die Releasedurchführung problemlos möglich.	Prozess
6	Es ist mehr ein inhaltliches Wachstumsproblem, weil unsere Software sehr spezifisch ist. Man braucht eine gewisse Kundenunterstützung, um diese Software zu liefern.	Probleme Team
6	Da versucht man mehr auf in die Richtung Automatisierung und Tooling zu arbeiten, damit die gleichen Leute mehr schaffen können. Das sind so mehr die Hürden. Also das Wachstum ist definitiv eine Hürde, wo das Tooling die Skalierbarkeit vor allem unterstützt.	Tool
6	Ja das war so eigentlich die Rolle von mir, um den Prozess dann so anzupassen, dass wir effizienter Releases parallel durchführen können.	Problembehandlung
6	Wir haben jetzt mittlerweile umgestellt auf eine Release Version. Die läuft dann über 4 Wochen und wird dann den Kunden ausgeliefert. Das sind so die Sachen, die wir aktiv in das Tagesgeschäft reinholen und die sind auch notwendig. Also die Rolle des Techleads und der Teamleads zusammen mit dem Q&A, die diese Informationen dann sammeln und entsprechend skalieren.	Problembehandlung
6	Nein also wir haben als ich gekommen bin sehr viel umstrukturiert.	Team
6	Wir haben zum Beispiel eine zuvor ein ausgelagertes externes Q&A Team gehabt, dass wir in die Entwicklungsmannschaft reingezogen, wo das Entwicklungsteam jetzt zusammenarbeitet, was ein wesentlicher Umbruch gewesen ist.	Team
6	Wir haben auch Support in den Entwicklungsprozess mit einbezogen. Es gab einen kleinen Zeitraum im Entwicklungsprozess, wo wir jetzt das Supportteam direkt mitnehmen.	Team
6	Wir haben sicher jedes Jahr zwei bis drei Anpassungen auf eine organisatorische Struktur, um mit dem Wachstum und dem Produktwachstums zurechtzukommen.	Team
6	Das ist ein ständiges Reviewen und Anpassen. Es werden Tuningmaßnahmen gemacht, wo es notwendig ist, das wird aber auch regelmäßig gemacht. Also wenn wir ein Releasefenster haben,	Prozess

	gibt es ein Post-Release-Meeting, wo wir darüber schauen, Feedback einsammeln, extern sowie intern und dann wird dieses Feedback nicht nur in den Release Prozess eingebunden, sondern auch in die Produktentwicklung eingetragen. Wenn man es eingrenzen würde, wäre das im 4-Wochen Zyklus, was unser Release Zyklus ist.	

Tabelle 12: Kodierung

ANHANG H: INTERVIEWLEITFADEN

Kategorie	Hypothese	Frage
Einleitung	-	1. Position, Firma, Berufserfahrung gesamt + im derzeitigen Unternehmen 2. Welches Tool verwendet Ihr in Euren Unternehmen für das Release Management? 3. Wie groß ist das Entwicklungsteam?
Erfahrung	1	1. Gibt es Erfahrung mit Devops/Jira? 2. Seit wann wird das Tool genutzt? 3. Wie war der erste Kontakt zum Tool? 4. Wie wurde die Entscheidung für das Tool getroffen?
Nutzung	2	1. Wieviel % der Funktionalität von dem Tool werden im Releasemanagement genutzt? 2. Gibt es Kennzahlen des Releasemanagements wie der Prozess sich über die Jahre entwickelt hat? Auch vor der Nutzung des Unternehmens? Bzw. kann gesagt werden, ob es besser geworden ist? 3. Welche Prozessteile werden bereits automatisiert? 4. Wieviel Stunden werden ca. pro Woche im Tool verbracht? 5. Wie wird das Tool genutzt? (Welche Teile im Prozess) 6. Braucht es zusätzliche Softwareprodukte, um einen guten Release durchzuführen?
Prozess	3	1. Gibt es einen visualisierten Release Prozess? 2. Kann der genutzte Prozess für mehrere Unternehmen verwendet werden? 3. Wurde das Tool auf den Prozess angepasst oder der Prozess auf das Tool? 4. Wie kommen (neue) Funktionen des Tools in den Prozess?
Rollen	4	1. Gibt es eine zuständige Person, die die Funktionalität des Tools testet und im Unternehmen einbringt? 2. Welche Rollen gibt es in Ihren Prozess? 3. Wird das Tool von allen Rollen benutzt?
Probleme	5	1. Welche Probleme gibt es in Ihren Unternehmen noch im Bereich des Releasemanagements? 2. Können diese Probleme mit den Tools behoben werden? 3. Werden die Probleme konkret aufgelistet und versucht zu lösen? 4. Wer ist für die Lösung der Probleme zuständig?

Tabelle 13: Interviewleitfaden

ABKÜRZUNGSVERZEICHNIS

CICD	Continuos Intgegration and continuous delivery
ITIL	Information Techology Infrastructury Library
DoD	Definition of Done
DoR	Definition of Ready
DBIST	Development & Build Integration & System Tests

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LITERATURVERZEICHNIS

Amir, Seyed Danesh; Mahmoud, Reza Saybani; Seyed, Yahya Seyed Danesh (2011): Software release management challenges in industry: An exploratory study. In *Afr. J. Bus. Manage.* 5 (20), pp. 8050–8056. DOI: 10.5897/AJBM10.1098.

Atlassian (2021a): Bitbucket-Kurzübersicht. Available online at https://www.atlassian.com/de/software/bitbucket/guides/getting-started/overview, updated on 10/21/2021, checked on 10/25/2021.

Atlassian (2021b): Confluence-Grundlagen. Available online at https://www.atlassian.com/de/software/confluence/guides/get-started/confluence-overview#about-confluence, updated on 10/21/2021, checked on 10/25/2021.

Atlassian (2021c): Kurzübersicht über Jira. Available online at https://www.atlassian.com/de/software/jira/guides/getting-started/overview, updated on 10/21/2021, checked on 10/25/2021.

Atlassian (2022): Jira Software-Features. Available online at https://www.atlassian.com/de/software/jira/features.

Barqawi, Neda; Syed, Kamran; Mathiassen, Lars (2016): Applying service-dominant logic to recurrent release of software: an action research study. In *JBIM* 31 (7), pp. 928–940. DOI: 10.1108/JBIM-02-2015-0030.

Beck, Kent et al. (2001): Manifest für Agile Softwareentwicklung. Available online at https://agilemanifesto.org/iso/de/manifesto.html, updated on 6/9/2020, checked on 10/25/2021.

Chen, Lianping (2017): Continuous Delivery: Overcoming adoption challenges.

Daly, Laura (2021): Der Weg zu stressfreien Software-Releases | Atlassian. Available online at https://www.atlassian.com/de/agile/software-development/stress-free-release, updated on 10/21/2021, checked on 10/25/2021.

Dhaya, Sindhu Battina (2021): The challenges and mitigation strategie of using DevOps during software development.

Dileepkumar, S. R.; Mathew, Juby (2021): Optimize Continuous Integration and Continuous Deployment in Azure DevOps for a controlled Microsoft.NET environment using different techniques and practices. In *IOP Conf. Ser.: Mater. Sci. Eng.* 1085 (1), p. 12027. DOI: 10.1088/1757-899X/1085/1/012027.

Ebert, Christof (2016): DevOps Technologies.

Elsässer, Wolfgang (2005): ITIL einführen und umsetzen. Leitfaden für effizientes IT-Management durch Prozessorientierung. München: Hanser.

Groß, Torsten (2021): Release-Management mit JIRA Software, Bitbucket und Bamboo. Available online at https://blog.seibert-media.net/blog/2016/08/09/release-management-mit-jira-software-bitbucket-und-bamboo/, updated on 10/25/2021, checked on 10/25/2021.

Hughes, Greg (2017): Achtung - Lücke. Sechs Schritte zum Überbrücken der Lücke zwischen der Softwareentwicklung und dem Betriebsteam durch Release Management.

Humble, Jez; Farley, David (2010): Continuous delivery. Reliable software releases through build, test, and deployment automation. Upper Saddle River NJ: Addison-Wesley.

Joby, P. P. (2019): Exploring DevOps. Challenges and benefits. In *JITDW* 01 (01), pp. 27–37. DOI: 10.36548/jitdw.2019.1.004.

Johansson, Anthon (2017): How can Atlassian products be modified to reduce the average time usage for common tasks.

Kajko-Mattsson M., Yulog F. (2005): Microsoft Word - Delivery IDPT.doc.

Khan, Muhammad Owais (2020): Fast Delivery, Continuously Build, Testing and Deployment with DevOps Pipeline Techniques on Cloud. In *IJST* 13 (5), pp. 552–575. DOI: 10.17485/ijst/2020/v13i05/148983.

Leite, Leonardo; Rocha, Carla; Kon, Fabio; Milojicic, Dejan; Meirelles, Paulo (2020): A Survey of DevOps Concepts and Challenges. In *ACM Comput. Surv.* 52 (6), pp. 1–35. DOI: 10.1145/3359981.

Lienemann, Gerhard (2006): ITIL-Change-Management. Hinweise und Vorgehensweisen aus der Praxis. 1. Aufl. Hannover: Heise. Available online at http://deposit.dnb.de/cgi-bin/dokserv?id=2750366&prov=M&dok_var=1&dok_ext=htm.

Mayring, Philipp (2021): Qualitative Content Analysis. A Step-by-Step Guide. Available online at https://books.google.at/books?hl=de&lr=&id=hCdLEAAAQBAJ&oi=fnd&pg=PA116&dq=mayring+qualitative+content+analysis+a+step+by+stp+guide&ots=XjQ6iYiOkl&sig=yL3p3XsMV_rAjKyeNMX4n_GmNzQ#v=onepage&q&f=false.

Michlmayr, Martin; Hunt, Francis; Probert, David (2005): Quality Practices and Problems in Free Software Projects.

Michlmayr, Martin; Hunt, Francis; Probert, David (2007): Release Management in Free Software Projects: Practices and Problems. In Joseph Feller, Brian Fitzgerald, Walt Scacchi, Alberto Sillitti (Eds.): Open Source Development, Adoption and Innovation, vol. 234. Boston, MA: Springer US (IFIP — The International Federation for Information Processing), pp. 295–300.

Microsoft (2022): What is Azure Boards? Available online at https://docs.microsoft.com/en-us/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops.

Miikka, Eloranta (2018): Continuous development and release automation of web applications.

Noll, John; Razzak, Mohammad Abdur; Bass, Julian M.; Beecham, Sarah (2017): A Study of the Scrum Master's Role 10611 (6), pp. 307–323. DOI: 10.1007/978-3-319-69926-4_22.

Rance, Stuart (2011): 6-Tips-to-Help-You-Improve-Change-Management.

Rasa G., Dr. RSD Wahida Banu (2019): The Roles and Responsibilities of ITIL Release Management Process.

Rasa G., Jagadheesh Kumar, Banu Wahida (2010): Release and Deployment Management using ITIL.

Roopesh, Nair (2021): Releasepipelines - Azure Pipelines. Available online at https://docs.microsoft.com/de-de/azure/devops/pipelines/release/?view=azure-devops, updated on 10/25/2021, checked on 10/25/2021.

Rossberg, Joachim (2019): Agile Project Management with Azure DevOps: Concepts, Templates, and Metrics: Apress.

Ruhe, G.; Saliu, M. O. (2005): The Art and Science of Software Release Planning. In *IEEE Softw.* 22 (6), pp. 47–53. DOI: 10.1109/MS.2005.164.

Saleem, Hussain; Burney, S. M. Aqil (2019): Imposing Software Traceability and Configuration Management for Change Tolerance in Software Production.

Samer, I. Mohamed (2016): Software Release Management Evolution. Comparative Analysis across Agile and DevOpsContinuous Delivery.

Schwartz, Lisa (2020): ITIL 4 - Decoupling deployment from release management. Available online at https://www.linkedin.com/pulse/itil-4-decoupling-deployment-from-release-management-lisa-schwartz/.

Senepathi, Mali; Buchan, Jim; Osman, Hady (2018): DevOps Capabilities, Practices, and Challenges: Insights from a Case Study.

Shanmugasundaram, Stuart; Sojini, B. (2018): An Overview on Release and Deployment Management Strategy, pp. 5–12.

Sikender, Mohammad Mohsienuddin (2019a): DevOps Automation Advances I.T. Sectors with the Strategy of Release Management. In *IJCTT* 67 (12), pp. 82–88. DOI: 10.14445/22312803/IJCTT-V67I12P114.

Sikender, Mohsienuddin Mohammad (2018): Improve Software Quality through practicing DevOps Automation.

Sikender, Mohsienuddin Mohammad (2019b): DevOps Automation Advances I.T. Sectors with the Strategy of Release Management.

Stacy, D., Prudnikov, M., Khan, A., & Shen, X. (2017): Practicing Continuous Integration and Continuous Delivery on AWS. Accelerating Software Delivery with DevOps.

Umme, A. K.; Shadikur, R.; Fakhrul, H.; Aras, M. I.; Karmand H. (2020): SW Release Challenges. A Case Study with Mitigation Plan Using Sem-Automated Process.

Wikström, Axel (2019): Benefits and challenges of Continuous Integration and Delivery - A Case Study.

WKO (2022): Klein- und Mittelbetriebe (KMU): Definition - WKO.at. Definition: Was versteht man unter KMU?