INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwicklung eines Konzepts zur plattformübergreifenden Integration des Release-Managements bei der HVBInfo

Silvia Knittl

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Betreuer: Michael Brenner
Dr. Markus Garschhammer
Karin Betz, HVBInfo

Abgabetermin: 31. März 2006

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwicklung eines Konzepts zur plattformübergreifenden Integration des Release-Managements bei der HVBInfo

Silvia Knittl

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Betreuer: Michael Brenner
Dr. Markus Garschhammer
Karin Betz, HVBInfo

Abgabetermin: 31. März 2006

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 31. März 2006

. . . . . .

(Unterschrift des Kandidaten)

Zusammenfassung

Im Bankenbereich ist der Einsatz von IT von existenzieller Bedeutung. Transaktionen können hier ohne die Verwendung von Informationstechnologien nicht stattfinden. Durch die IT verursachte Fehler können hier schnell zu finanziellen Verlusten führen. Aus diesem Grund ist in diesem Umfeld die Sicherstellung der geforderten Funktionsfähigkeit der IT-Infrastruktur besonders wichtig. Das gilt vor allem, wenn neue oder geänderte Soft- oder Hardware eingesetzt werden soll.

Im Kontext dieser Diplomarbeit ist es die Aufgabe des Release-Managements, die produktive IT-Infrastruktur vor Fehlern beim Einführen von Softwareänderungen zu schützen. Diese Aufgabe erledigt das Release-Management durch die Koordination von vorab stattfindenen Abnahmetests des Kunden und durch eine kontrollierte Verteilung und Installation dieser Softwareänderungen beim Kunden. Diese Diplomarbeit findet in Zusammenarbeit mit der HVBInfo statt. Die HVBInfo betreibt die IT-Infrastruktur von Unternehmen innerhalb HVB-Group.

Bei der HVBInfo haben sich mehrere Release-Managementbereiche etabliert. Diese sind historisch entstanden und unterscheiden sich hauptsächlich durch ein Ausrichtung an der grundlegenden Technologie. Aus diesem Grund gibt es derzeit bei der HVBInfo je ein eigenes Release-Verfahren für die sogennanten Plattformen Client/Server, zentrale Server und Host. Durch diese Mehrgleisigkeit im Release-Management entstehen Nachteile durch hohe Koordinations- und Kommunikationsaufwände, durch unterschiedliche Prozessschnittstellen oder uneinheitliche Begriffswelten.

Im Rahmen dieser Arbeit wurde ein Konzept zur Integration des Release-Managements entwickelt. Dazu wurden zunächst ein Lebenszyklusmodell von Software eingeführt, mit dessen Hilfe verwandte Arbeiten zum Release-Management vorgestellt und klassifiziert wurden. Das Lebenszyklusmodell und die verwandten Arbeiten bildeten eine Grundlage zur Beschreibung und Analyse des bestehenden Release-Managements bei der HVBInfo. Des Weiteren bildete vor allem das Release-Management aus ITIL eine Basis zur Ableitung von Anforderungen an das zu konzipierende integrierte Release-Management und die Basis zu dessen Darstellung.

Das entstandene Konzept zur Integrations des Release-Managements bei der HVBInfo ist technologieunabhängig. Es vereinfacht durch klare Prozessschnittstellen die Koordination und Kommunikation der Prozessbeteiligten. Durch die Einführung einheitlicher Begriffe kann das Verständis der beteiligten Mitarbeiter verbessert werden. Aufgrund der Standardisierung des Release-Prozesses mithilfe der Ausrichtung an ITIL ist die Grundlage für weitere Optimierung des Release-Managements durch die Einführung von unterstützenden Werkzeugen oder durch Automatisierung von Teilaktivitäten gelegt.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Fokus und Aufgabenstellung der Diplomarbeit 1
1.2 Vorgehen bei der Diplomarbeit 3
1.3 Ergebnis der Diplomarbeit 5

2 Verwandte Arbeiten im Release-Management **6**
2.1 Lebenszyklusmodell und Klassifikationsschema 8
2.1.1 Kanonisches Lebenszyklusmodell des Softwareentwicklungsprozesses 8
2.1.2 Schema zur Klassifikation der Prozessmodelle 13
2.2 Release-Management als Bestandteil im Softwareengineering 14
2.2.1 V-Modell 97 15
2.2.2 V-Modell XT 17
2.2.3 Microsoft Solutions Framework 18
2.2.4 Rational Unified Process 21
2.2.5 Capability Maturity Model Integration 23
2.2.6 Weitere Beispiele für Release-Management innerhalb des Softwareengineerings 24
2.2.7 Software Engineering Body of Knowledge 35
2.3 Release-Management im IT-Servicemanagement 36
2.3.1 Microsoft Operations Framework 37
2.3.2 Solution Management and Request Tracking 39
2.3.3 SERVIAM Maintenance Framework 40
2.3.4 IT Infrastructure Library 43
2.3.5 Control Objectives for Information and Related Technology 50
2.3.6 Maturity Model for IT Operations 51
2.3.7 Software Maintenance Capability Maturity Model 53
2.3.8 IT Service Capability Maturity Model 54
2.3.9 Weitere Beispiele für Release-Management innerhalb von IT-Servicemanagement . . . 56
2.4 Zusammenfassung 58
2.4.1 Release-Managementbegriffe mit unterschiedlicher Bedeutung 61
2.4.2 Release-Management im Kontext der Diplomarbeit 62

3 Beschreibung des derzeitigen Release-Managements bei der HVBInfo **65**
3.1 Die HVB Informations-Verarbeitungs-GmbH und ihr Umfeld 66
3.2 Release-Management bei der HVBInfo 67
3.2.1 Plattformen bei der HVBInfo 68
3.2.2 Begriffe im Release-Management bei der HVBInfo 69
3.3 Motivation für die Integration der vorhandenen Release-Prozesse 75
3.4 Release-Management im dezentralen Bereich 77
3.4.1 Beteiligte Rollen und ihre Aufgaben 77
3.4.2 Release-Aktivitäten im dezentralen Bereich 79
3.4.3 Beschreibung des Release-Steuerungsprozesses 88
3.4.4 Bewertung des Release-Durchlaufes 97
3.4.5 Einordnung in das Lebenszyklusmodell 98
3.5 Release-Management bei zentralen Servern 99
3.5.1 Beteiligte Rollen und ihre Aufgaben 99
3.5.2 Release-Aktivitäten im Bereich zentraler Server 102
3.5.3 Beschreibung des Release-Steuerungsprozesses 109
3.5.4 Bewertung des Release-Durchlaufes 111

3.5.5 Einordnung in das Lebenszyklusmodell . . . 112
3.6 Release-Management im Host-Bereich . . . 113
3.6.1 Beteiligte Rollen und ihre Aufgaben . . . 113
3.6.2 Release-Aktivitäten im Host-Bereich . . . 114
3.6.3 Beschreibung des Release-Steuerungsprozesses . . . 121
3.6.4 Bewertung des Release-Durchlaufes . . . 125
3.6.5 Einordnung in das Lebenszyklusmodell . . . 125
3.7 Zusammenfassung . . . 126

4 Analyse des bestehenden Release-Managements und Ableitung von Anforderungen an das integrierte Release-Management 129
4.1 Analyse des Release-Managements im dezentralen Bereich . . . 130
4.1.1 Verbindung zum Change-Management . . . 131
4.1.2 Fehlendes Konfigurationsmanagement . . . 132
4.1.3 Allgemeine Defizite im bestehenden Release-Management im dezentralen Bereich . . . 133
4.2 Analyse des Release-Managements im Bereich zentraler Server . . . 135
4.2.1 Fehlendes Release-Management . . . 136
4.2.2 Allgemeine Defizite im bestehenden Release-Management im Bereich zentraler Server 137
4.3 Analyse des Release-Managements im Host-Bereich . . . 138
4.4 Ableitung von Anforderungen an das zu konzipierende, plattformübergreifende Release-Management . . . 140
4.4.1 Orientierung des RM an ITIL . . . 140
4.4.2 Qualitätsmanagement . . . 143
4.4.3 Umgebungsmanagement . . . 143
4.4.4 Anforderungen aufgrund der höheren Komplexität des integrierten Release-Managements . . . 144
4.5 Zusammenfassung . . . 147

5 Plattformübergreifendes Release-Management 149
5.1 Beteiligte Rollen und ihre Aufgaben . . . 150
5.2 Release-Aktivitäten . . . 154
5.2.1 Festlegen der Releasepolicy und des Release-Plans . . . 154
5.2.2 Entwurf, Release-Zusammenstellung und -Konfiguration . . . 159
5.2.3 Testen und Abnahme des Releases . . . 159
5.2.4 Planung des Einsatzes . . . 160
5.2.5 Kommunikation, Vorbereitung und Schulung . . . 160
5.2.6 Release-Verteilung und Installation . . . 161
5.2.7 Beschreibung des Release-Steuerungsprozesses . . . 161
5.2.8 Bewertung des Release-Durchlaufes . . . 165
5.3 Einordnung in das Lebenszyklusmodell . . . 167
5.4 Vorteile durch das integrierte Release-Management . . . 168
5.5 Zusammenfassung . . . 168

6 Zusammenfassung 170

1 Einleitung

Die Abhängigkeit von IT-Prozessen in Unternehmen steigt ständig. Das gilt besonders im Bankenbereich, wie folgendes Beispiel zeigt. „U.S. Federal Reserve System: In der ersten Nacht der Inbetriebnahme eines neuen Systems wurden 28 Milliarden Dollar an Banken falsch überwiesen. Zurücküberwiesen wurden allerdings nur 24 Milliarden Dollar, da das gesammte Geld nicht wieder auffindbar war“ [Mat05]. Hier leidet im Fehlerfall nicht nur das Image eines Unternehmens, sondern es entsteht sogar ein finanzieller Schaden. Dieser Vorfall zeigt die Abhängigkeit von Geschäftsprozessen von einer funktionierenden Informationstechnologie. Hier ist es besonders wichtig, dass die Einführung von Änderungen an der IT-Infrastruktur kontrolliert erfolgt. Das Release-Management (RM) ist zuständig für die Sicherung der produktiven IT-Infrastruktur bei der Einführung von Änderungen durch die Festlegung geeigneter Maßnahmen.

1.1 Fokus und Aufgabenstellung der Diplomarbeit

Die vorliegende Diplomarbeit wurde in Zusammenarbeit mit der HVBInfo durchgeführt. Die HVBInfo ist eine Tochterfirma der HVB Group. Sie ist verantwortlich für den Betrieb der IT-Infrastruktur ihrer Kunden. Kunden sind u.a. die Hypo Vereinsbank, die HVB Immobilien und weitere Tochtergesellschaften der HVB Group. Die HVBSystems ist ein weiterer IT-Dienstleister innerhalb der HVB Group. Sie entwickelt Software für ihre Kunden. Für die HVBInfo stellt sie den wichtigsten Softwarelieferanten dar. Die Zusammenarbeit mit ihr ist deshalb sehr eng.

Die HVBInfo bietet verschiedene Dienstleistungen für ihre Kunden an. Dazu gehören u.a. die Bereitstellung standardisierter PC-Arbeitsplätze, das Lizenzmanagement oder Output-Services (siehe Kapitel 3.1 ab Seite 66). Als eine der Dienstleistungungen wird auch die kontrollierte und gesteuerte Einführung von Änderungen, speziell Softwareänderungen, in die bestehende IT-Infrastruktur angeboten. Diese Dienstleistung wird durch das Release-Management erbracht. Es stellt hierfür organisatorische und technische Mittel und Methoden bereit. Dadurch können Änderungen an den IT-Beständen effektiv, sicher und nachvollziehbar durchgeführt werden.

Das Release-Management bei der HVBInfo hat den Fokus auf die kontrollierte und gesteuerte Einführung von Softwareänderungen in die IT-Infrastruktur. Diese geänderte Software wird häufig von der HVBSystems geliefert. Im Falle von anderen externen Softwarelieferanten werden die für den Einführungsprozess notwendigen Rollen entweder von Mitarbeitern der HVBSystems oder der HVBInfo wahrgenommen. Aus diesem Grund wird in dieser Diplomarbeit nicht mehr zwischen HVBSystems und anderen Softwarelieferanten unterschieden. Die Änderung, Einführung oder Aktualisierung jeglicher Hardware wird in dieser Arbeit nicht betrachtet.

Das Release-Management bei der HVBInfo ist nicht einheitlich. Es haben sich drei technologieorientierte Release-Verfahren entwickelt. Diese Dreiteilung hat sich historisch gebildet und betrifft die drei so genannten Plattformen den ***dezentralen Bereich*** bzw. ***Bereich Client/Server (C/S)***, den Bereich ***zentraler Server (zS)*** und den ***Host-Bereich***. Das Release-Management im dezentralen Bereich bzw. Bereich Client/Server ist zuständig für die Einführung von Softwareänderungen in die Infrastruktur, die beim jeweiligen Kunden vor Ort ist (Standortserver und Fat Clients). Bei den zentralen Servern handelt es sich um *UNIX*- und *Windows*-Server, die in den Räumen der HVBInfo stehen. Beim Host handelt es sich um *Mainframes* von IBM (zSerie). Sie befinden sich ebenfalls vor Ort bei der HVBInfo.

Das Release-Management dieser drei Bereiche unterscheidet sich neben der unterschiedlichen Technologieorientierung auch durch unterschiedliche Ausprägungen u.a. bezüglich der verantwortlichen Rollen und der Prozessabläufe. Durch diese technologieorientierte Aufteilung können derzeit keine plattformübergreifende Geschäftsprozesse der Kunden vor Einführung in die Produktion getestet werden. Dadurch bleiben Fehler

möglicherweise unentdeckt und treten erst im operativen Betrieb auf. Die verschiedenen Verantwortlichkeiten der jeweiligen Release-Verfahren, die entweder bei den Softwareentwicklern oder beim bereits etablierten Release-Management liegen, führen manchmal zu Unklarheiten über die konkrete Aufgabenaufteilung. Bei den jeweiligen Release-Prozessabläufen variieren der erwartete Fertigstellungsgrad der Software bei der Übergabe und die erwarteten Begleitdokumente. Im dezentralen Bereich wird z.B. davon ausgegangen, dass ein Produkt bereits fertig integriert ist, während im Bereich zS die Integration ein Bestandteil des Release-Prozesses ist. Die Art und der Umfang der durchzuführenden Tests unterscheidet sich bei den jeweiligen Verfahren. Im Host-Bereich werden beispielsweise die Mehrzahl der Softwareänderungen direkt von der Entwicklungsphase in die operative Betriebsphase übergeben, während im dezentralen Bereich umfangereiche Tests in dedizierten Testumgebungen vor Einführung in die Produktivumgebung vorgesehen sind. Diese drei Verfahren zur Einführung von Software haben sich unabhängig voneinander etabliert. Aus diesem Grund werden teilweise auch unterschiedliche Begriffe verwendet. Der Release-Prozess im dezentralen Bereich wird im Bereich zS Prozess zur Produktionsübergabe genannt. Aufgrund der unterschiedlichen Ausprägungen ist das Verständnis der beteiligten Mitarbeiter füreinander nicht ausgeprägt.

Das Release-Management im Bereich C/S war ursprünglich bei der HVBSystems angesiedelt. Vor ca. drei Jahren wurde es von der HVBSystems zur HVBInfo verlagert. Im dezentralen Bereich ist ein Release-Management organisatorisch durch eine eigene Abteilung (Inf1RQ) vertreten. Der dort stattfindende Release-Prozess ist seit längeren etabliert.

Im Bereich zentraler Server wird prinzipiell unterschieden zwischen zentralen *Windows*- und *UNIX*- (inkl. *Linux*-) Servern. Im Windows-Umfeld ist das Release-Management im Aufbau. Es orientiert sich im Vorgehen und Ablauf am Release-Management des dezentralen Bereichs. Allerdings gilt das nur für die Einführung von Änderungen an Betriebssystemen und betriebssystemnaher Software wie etwa Sicherheitsupdates. Für andere Anwendungen im Windows-Bereich und alle Anwendungen im Bereich zentraler Server ist derzeit kein Release-Management definiert. Die Einführung von Änderungen an der IT-Infrastruktur wird dort von den jeweiligen Softwareentwicklern gesteuert. Diese befinden sich i.d.R. bei der HVBSystems. Der Prozess zur Einführung von Softwareänderungen wird hier nicht Release-Prozess sondern Produktionsübergabeprozess genannt.

Im Host-Bereich wird seit Anfang 2004 durch ein Projekt ein Release-Management analog dem Vorgehen und Ablauf im dezentralen Bereich eingeführt. Hier erfolgt die Einführung von Änderungen derzeit pilothaft über einen durch das Release-Management gesteuerten Prozess. Die Mehrzahl der Einführungen von Softwareänderungen wird jedoch weiterhin über die jeweiligen Softwareentwickler gesteuert. Diese sind auch meist von der HVBSystems.

Die Aufgabe dieser Diplomarbeit ist das Entwickeln eines Konzeptes zur plattformübergreifenden Integration des Release-Managements. Durch diese Integration werden die bestehenden drei Verfahren zur Einführung von Software vereinheitlicht und erfolgen unter Steuerung und Kontrolle eines plattformübergreifenden Release-Managements. Durch diese Integration ergeben sich Vorteile durch Qualitätsverbesserungen und durch Standardisierung. Die Qualität verbessert sich durch definierte und kontrollierte Abnahmen von geänderter Software vor der Einführung in die Produktionsumgebung. Durch ein plattformübergreifendes Release-Management können diese Abnahmen in einer synchronisierten Testumgebung stattfinden. Diese kann produktionsähnlich gestaltet werden. Dadurch sind in Zukunft die Tests kompletter Geschäftsprozesse des Kunden möglich, was beim derzeit plattformspezifischen Vorgehen nicht bzw. nur mit hohem Koordinationsaufwand funktioniert. Die Abnahme kompletter Geschäftsprozesse wurde projekthaft im Rahmen der Integration der Vereins- und Westbank durchgeführt. Bei diesem Projekt konnten im Release-Prozess die Anzahl der *Incidents* vor Einführung in die Produktivumgebung deutlich gesenkt werden (siehe auch Abschnitt Motivation der Integration 3.3).

Durch ein standardisiertes Vorgehen beim Release-Prozess ergeben sich weitere Vorteile. Die Standardisierung legt die Grundlage für die Automatisierung von Aktivitäten. Das führt auch zu finanziellen Vorteilen. Automatisierungspotenzial ergibt sich vor allem bei technischen Tätigkeiten wie Installationaktivitäten oder bei der Softwareverteilung. Das standardisierte Release-Management verbessert weiterhin durch die Definition von einheitlichen Begriffen das Verständnis der Prozessbeteiligten untereinander.

1.2 Vorgehen bei der Diplomarbeit

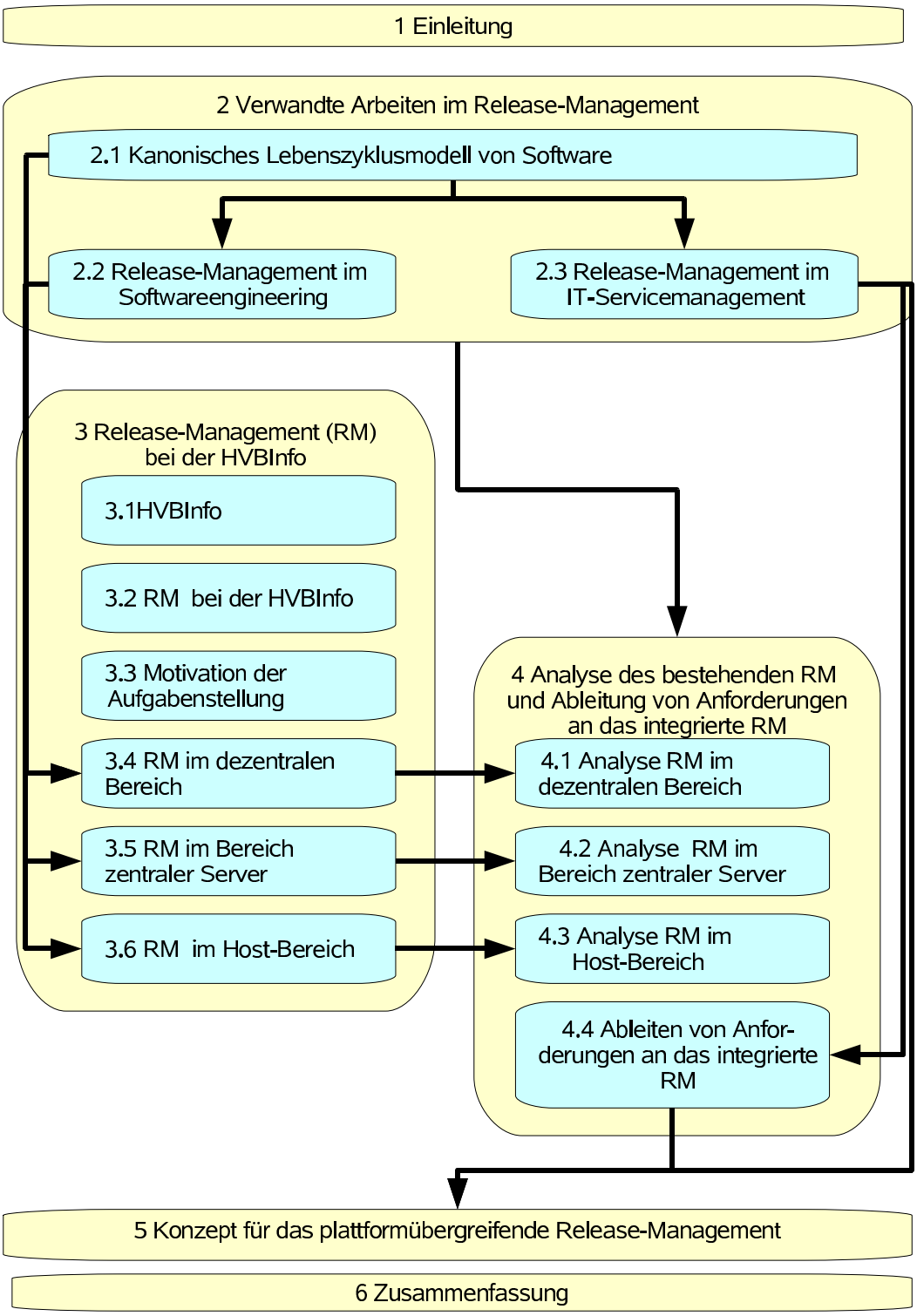


Abbildung 1.1: Vorgehen bei der Diplomarbeit

Abbildung 1.1 zeigt das Vorgehen bei der Erstellung der Diplomarbeit. In Kapitel 2 erfolgt eine Übersicht über verwandte Arbeiten zum Thema Release-Management. Hier wird das Release-Management bei bekann-

ten umfassenden Referenzmodellen wie etwa beim *Rational Unified Process* (vgl. Abschnitt 2.2.4) oder dem *Microsoft Operation Framework* (vgl. Abschnitt 2.3.1) und bei weniger bekannten und weniger umfassenden Forschungsarbeiten (siehe z.B. Abschnitt 2.2.6) untersucht.

Bei den vorgestellten Arbeiten in Kapitel 2 gibt es prinzipiell zwei verschiedene Sichtweisen auf das Release-Management. Es kann dem Softwareengineering oder dem IT-Servicemanagement zugeordnet werden. Aus diesem Grund wird in Abschnitt 2.1 zunächst ein kanonisches Lebenszyklusmodell von Software eingeführt. Es besteht aus den Lebenszyklusphasen von Software von der Machbarkeitsstudie bis hin zur Einstellung eines Produktes. Mithilfe dieses Lebenszyklusmodells werden die vorgestellten verwandten Arbeiten zum Release-Management und die bei der HVBInfo bestehenden Release-Verfahren entweder dem Softwareengineering oder dem IT-Servicemanagement zugeordnet. Ein weiteres Unterscheidungsmerkmal der vorgestellten verwandten Arbeit ist es, ob sie jeweils zur Beschreibung von Prozessen oder zu ihrer Bewertung dienen. Die Einteilung der beschriebenen verwandten Arbeiten bezüglich dieser vier Unterscheidungsmerkmale, d.h. Softwareengineering vs. IT-Servicemanagement und Beschreibung vs. Bewertung erfolgt als Orientierungshilfe in einem Klassifikationsschema jeweils am Anfang jedes Abschnittes.

In Abschnitt 2.2 werden Referenzmodelle und Forschungsarbeiten vorgestellt, bei denen sich das Release-Management dem Softwareengineering zuordnen läßt. Das Softwareengineering hat den Schwerpunkt auf der Erstellung von Softwaresystemen. In diesem Kontext ist ein Release eine bestimmte Version einer Software. Das Aufgabengebiet des Release-Managements reicht hier vom Prozess zur Erstellung eines Releases und der Interaktion mit den Kunden (vgl. z.B. Release-Management als Teil des *Product Software Lifecycle* Management in Abschnitt 2.2.6.2) bis hin zur Unterstützung der Einführung dieses Releases beim Kunden (vgl. z.B. Release-Management als Bestandteil der Abnahme- und Einführungsphase in Abschnitt 2.2.6.8) oder der technischen Verteilung und Installation eines Releases (vgl. z.B. Release-Prozess als Bestandteil des Deployment-Prozesses in Abschnitt 2.2.6.6).

Ab Abschnitt 2.3 werden Referenzmodelle und Arbeiten aus der Forschung beschrieben, bei denen sich das Release-Management dem IT-Servicemanagement zuordnen läßt. Im IT-Servicemanagement liegt der Schwerpunkt auf dem operativen Betrieb von IT-Systemen. In diesem Kontext ist das Release-Management für die kontrollierte Einführung von Änderungen an u.U. mehreren Softwareprodukten in die bestehende IT-Infrastruktur zuständig (siehe z.B. *IT Infrastructure Library* in Abschnitt 2.3.4). Auch hier werden die technischen Aspekte der Release-Verteilung und -Installation (siehe Release-Prozess als Bestandteil des Deployment-Prozesses in Abschnitt 2.3.9.1) betrachtet.

Diese vorgestellten verwandten Arbeiten zum Release-Management dienen als Grundlage zur Einordnung, Beschreibung und Analyse der bei der HVBInfo bestehenden Release-Prozesse. Weiterhin dienen sie als Grundlage zum Ableiten von Anforderungen an das zu konzipierende integrierte Release-Management. Hierfür erfolgt in Kapitel 3 zunächst eine Beschreibung der HVBInfo mit ihrem Umfeld. Ab Abschnitt 3.2.2 werden der Fokus des Release-Managements bei der HVBInfo und wichtige releaserelevante Begriffe eingeführt. Die Beschreibung des bestehenden Release-Managements bzw. der bestehenden Prozesse zur Produktionseinführung erfolgt für den dezentralen Bereich in Abschnitt 3.4, für den Bereich zentraler Server in Abschnitt 3.5 und für den Host-Bereich in Abschnitt 3.6. Die bestehenden derzeitigen Verfahren bei der HVBInfo werden entsprechend des eingeführten Lebenszyklusmodell dem Softwareengineering zugeordnet.

Die HVBInfo ist eine reine Betreiberorganisation ohne nennenswerte eigene Softwareentwicklung. Aus diesem Grund wird als Grundlage für die Konzeption des integrierten Release-Managements die *IT Infrastructure Library* (ITIL) aus dem Bereich IT-Servicemanagement (siehe Abschnitt 2.3.4) gewählt. ITIL hat sich als Defacto-Standard in diesem Bereich etabliert. Weiterhin hat die HVBInfo bereits die Prozesse Change-, Incident und Problem-Management an ITIL orientiert (siehe Abschnitt Release-Management im Kontext der Diplomarbeit 2.4.2).

Die Gliederung der Beschreibung des bestehenden Release-Managements in Abschnitt 3.4 bis 3.6 entspricht der Gliederung von ITIL. Es gibt aufgrund der Zuordnung des bestehenden Release-Managements zum Softwareengineering nicht für jeden Punkt der ITIL-Gliederung entsprechende Beschreibungen bei den bestehenden Prozessen. Diese Gliederungsweise vereinfacht jedoch die Analyse der bestehenden Verfahren und das Ableiten von Anforderungen an das plattformübergreifende Release-Management. Fehlende Aspekte von ITIL, die aufgrund der gewählten Gliederung ersichtlich werden, sind gleichzeitig Anforderungen an das neu zu konzipierende Release-Management. Zu jedem der drei vorgestellen Bereiche werden die am Prozess beteiligten Rollen, ihre Aufgaben und die Release-Aktivitäten dargestellt. In Kapitel 4 erfolgt jeweils eine Analyse

von jedem betrachteten Bereich. In Abschnitt 4.4 werden daraus die Anforderungen an das neue plattformübergreifende Release-Management definiert.

Mithilfe dieser Anforderungen und der Beschreibung des Release-Managements in ITIL aus Abschnitt 2.3.4 erfolgt in Kapitel 5 die Beschreibung des Konzeptes zum integrierten Release-Managements. Hier werden ebenfalls die am Prozess beteiligten Rollen mit ihren Aufgaben und die Release-Aktivitäten dargestellt. Die Vorteile des neu konzipierten Release-Managements werden in Abschnitt 5.4 zusammengefasst. Abschließend erfolgt in Kapitel 6 eine Zusammenfassung der Diplomarbeit und ein Ausblick auf weitere offene Fragestellungen.

Hinweise zum Layout:
Englische Begriffe sind *kursiv* gesetzt. Ausnahmen bilden zentrale Begriffe des Release-Managements wie etwa alle „Release-“Begriffe, der Begriff „Change“ und „Deployment“.
ITIL- Begriffe und Begriffe der *HVBInfo* werden beim erstmaligen Auftreten farbig gekennzeichnet. Eine weitere farbige Kennzeichnung zur Unterscheidung erfolgt hier nur zur besonderen Hervorhebung des jeweiligen Aspektes. Im Text benannte *Produkte* verschiedener Hersteller oder Organisationen werden ebenfalls hervorgehoben. Am Ende der Diplomarbeit sind Verzeichnisse zu im Text enthaltenen Abbildungen, Tabellen und Abkürzungen, sowie ein Literaturverzeichnis und ein Index.

1.3 Ergebnis der Diplomarbeit

Das Ergebnis dieser Diplomarbeit ist das Konzept für ein integriertes Release-Management bei der HVBInfo. Das Konzept beschreibt die beteiligten Rollen mit ihren Aufgaben und die Aktivitäten. Die bestehenden plattformspezifischen und damit technologieorientierten Release-Verfahren werden durch ein einheitliches und standardisiertes Release-Management abgelöst. Das neue Release-Management ist an ITIL orientiert und hat klare Prozessschnittstellen. Durch einheitliche Begriffe kann das Verständnis der Prozessbeteiligten füreinander verbessert werden. Der neue Release-Prozess ist generisch angelegt. So können in der Diplomarbeit nicht betrachtete Release-Prozessinseln leicht zusätzlich integriert werden.

Durch die Standardisierung ist eine Grundlage für weiterführende Automatisierungsaktivitäten gelegt. Automatisierungspotenzial besteht vor allem bei den im Prozess anfallenden Testaktivitäten und bei den technischen Tätigkeiten im Deployment-Bereich (z.B. Installation, Verteilung, Aktivierung von Software). Dadurch kann der gesamte Prozessdurchlauf beschleunigt werden.

Durch die plattformübergreifende, gebündelte Einführung von Softwareänderungen können zukünftig Geschäftsprozesse des Kunden in einer produktionsnahen Testumgebung getestet werden. Das ist derzeit nicht möglich. Dadurch kann die Qualität der Softwareänderungen im Vorfeld überprüft werden.

Durch die Orientierung an ITIL wird das Release-Management zukünftig früher in den Änderungsprozess eingebunden. Besonders die Planung von Releases erfolgt dadurch in Zukunft proaktiv. Anstatt wie heute reaktiv auf übergebene Änderungen zu reagieren, kann so die Auslastung der am Prozess beteiligten Ressourcen besser gesteuert werden.

2 Verwandte Arbeiten im Release-Management

Inhaltsverzeichnis

2.1 Lebenszyklusmodell und Klassifikationsschema 8
2.1.1 Kanonisches Lebenszyklusmodell des Softwareentwicklungsprozesses 8
2.1.1.1 zu 1) Machbarkeitsstudie 9
2.1.1.2 zu 2) Projektinitiierung 9
2.1.1.3 zu 3 - 7) Systementwicklung 10
2.1.1.4 zu 8) Integration 10
2.1.1.5 zu 9 - 11) Tests 10
2.1.1.6 zu 12) Projektabschluss 11
2.1.1.7 zu 13) Betrieb inklusive Wartung 11
2.1.1.8 zu 14) Einstellung des Produktes 13
2.1.2 Schema zur Klassifikation der Prozessmodelle 13
2.2 Release-Management als Bestandteil im Softwareengineering 14
2.2.1 V-Modell 97 15
2.2.1.1 Release-Management im V-Modell 16
2.2.1.2 Einordnung des V-Modells in das Lebenszyklusmodell 16
2.2.2 V-Modell XT 17
2.2.2.1 Release-Management im V-Modell XT 17
2.2.2.2 Einordnung des V-Modells XT in das Lebenszyklusmodell 18
2.2.3 Microsoft Solutions Framework 18
2.2.3.1 MSF Grundprinzipien 19
2.2.3.2 MSF Modelle 19
2.2.3.3 MSF Disziplinen 20
2.2.3.4 Release-Management bei MSF 20
2.2.3.5 Einordnung von MSF in das Lebenszyklusmodell 20
2.2.4 Rational Unified Process 21
2.2.4.1 Release-Management beim Rational Unified Process 22
2.2.4.2 Einordnung vom Rational Unified Process in das Lebenszyklusmodell . . 22
2.2.5 Capability Maturity Model Integration 23
2.2.5.1 Release-Management bei CMMI 23
2.2.5.2 Einordnung von CMMI in das Lebenszyklusmodell 23
2.2.6 Weitere Beispiele für Release-Management innerhalb des Softwareengineerings . . . 24
2.2.6.1 Release-Management als Bestandteil der Softwarelogistik 24
2.2.6.2 Release-Management als Teil des Product Software Lifecycle Management 24
2.2.6.3 Release-Prozess als Prozess zwischen Entwicklungs- und Deployment-Prozess 26
2.2.6.4 Release-Management als Subprozess innerhalb der Wartung 26
2.2.6.5 Release-Management als Prozess zwischen Entwicklung und Installation . 29
2.2.6.6 Release-Prozess als Bestandteil des Deployment-Prozesses 29
2.2.6.7 Release-Management als Bestandteil des Konfigurationsmanagements . . 32
2.2.6.8 Release-Management als Bestandteil der Abnahme- und Einführungsphase 33
2.2.7 Software Engineering Body of Knowledge 35

2.2.7.1 Release-Management beim Software Engineering Body of Knowledge . . 35
2.2.7.2 Einordnung des Software Engineering Body of Knowledge in das Lebenszyklusmodell 36
2.3 Release-Management im IT-Servicemanagement 36
2.3.1 Microsoft Operations Framework 37
2.3.1.1 Release-Management beim Microsoft Operations Framework 38
2.3.1.2 Einordnung von MOF in das Lebenszyklusmodell 39
2.3.2 Solution Management and Request Tracking 39
2.3.2.1 Release-Management bei Solution Management and Request Tracking . . 39
2.3.3 SERVIAM Maintenance Framework 40
2.3.3.1 Evolution and Maintenance Maturity Model 40
2.3.4 IT Infrastructure Library 43
2.3.4.1 Release Management nach ITIL 43
2.3.4.2 Wichtige Begriffe des ITIL Release-Management 44
2.3.4.3 Release-Prozess in ITIL 47
2.3.4.4 Einordung von ITIL in das Lebenszyklusmodell 49
2.3.5 Control Objectives for Information and Related Technology 50
2.3.5.1 Release-Management bei CobIT 50
2.3.5.2 Einordnung von CobIT in das Lebenszyklusmodell 50
2.3.6 Maturity Model for IT Operations 51
2.3.6.1 Release-Management bei MITO 51
2.3.6.2 Einordnung von MITO in das Lebenszyklusmodell 52
2.3.7 Software Maintenance Capability Maturity Model 53
2.3.7.1 Release-Management beim Software Maintenance Capability Maturity Model 53
2.3.7.2 Einordnung von SM^{CMM} in das Lebenszyklusmodell 54
2.3.8 IT Service Capability Maturity Model 54
2.3.8.1 Release-Management beim IT Service Capability Maturity Model 55
2.3.8.2 Einordnung des IT Service Capability Maturity Model in das Lebenszyklusmodell 56
2.3.9 Weitere Beispiele für Release-Management innerhalb von IT-Servicemanagement . . 56
2.3.9.1 Release-Prozess bei der Object Management Group 56
2.3.9.2 Release-Management innerhalb des Wartungsmanagement bei Curth et. al. 57
2.4 Zusammenfassung 58
2.4.1 Release-Managementbegriffe mit unterschiedlicher Bedeutung 61
2.4.1.1 Konfigurationsmanagement 61
2.4.1.2 Release-Planung 61
2.4.1.3 Änderungsmanagement 61
2.4.2 Release-Management im Kontext der Diplomarbeit 62

Dieses Kapitel liefert eine Übersicht über bestehende Ansätze zum Release-Management. Die Sichtweise auf das Release-Management (RM) ist breit gefächert. Das RM wird sehr oft als Teildisziplin im Softwareengineering (SWE) aufgefasst. Hier wird eine neue Version eines Produktes als „das Release“ bezeichnet (siehe z.B. *Rational Unified Process* im Abschnitt 2.2.4). Aus diesem Grund sind die Tätigkeiten innerhalb des RM ausgerichtet an der Bereitsstellung oder Verfügbarmachung dieses Produktes. Die Prozesse des RM haben hier einen Fokus auf das Release als Produkt. Der Softwareengineeringprozess endet in der Regel mit der Release-Bereitstellung.

Eine andere Sichtweise ist die Einordnung des RM als Prozess im operativen IT-Betrieb als Teildisziplin des IT-Servicemanagements (ITSM) (siehe z.B. *IT Infrastructure Library* im Abschnitt 2.3.4). In dieser Einordnung liegt der Fokus auf der Bereitstellung und Einführung von Produkten in eine operative IT Umgebung. Die Aktivitäten dieses Prozesses starten hier nach der Bereitstellung eines Produktes, also wenn der Prozess aus Sicht des Softwareengineerings im Prinzip beendet ist. Die Prozesse des RM haben hier den Fokus auf RM als Dienst zur Einführung von Produkten in die operative IT Umgebung.

Auf Basis dieser beiden Sichtweisen wird nachfolgend ein Schema zur Klassifikation von RM eingeführt. Mithilfe dieses Klassifikationsschemas erfolgt eine Einteilung der untersuchten Quellen. Durch die verschiedenen Sichtweisen auf das RM ergibt sich eine Vielzahl von Release-Begriffen mit unterschiedlichen Bedeutungen oder Verwendungen. Ebenfalls gibt es Einordnungen von Release-Begriffen, die sich jeweils nur auf ein begrenztes Teilgebiet des SWE oder ITSM beziehen.

Aus diesem Grund wird im nachfolgenden Abschnitt zunächst ein kanonisches Lebenszyklusmodell von Software vorgestellt. Anhand dieses Modells erfolgt eine Einordnung der untersuchten Begriffe und Prozesse. Das Lebenszyklusmodell beinhaltet die Prozessschritte von der Machbarkeitsstudie bis zur Einstellung eines Produktes. Dieses Vergleichsschema wird verwendet, weil die gefundenen Quellen teilweise nur einen bestimmten Aspekt des Lebenszyklus abdecken. Die Zuordnung soll die Einteilung erleichtern.

Mithilfe dieser Klassifikation soll die Analyse der bestehenden Ist-Prozesse (siehe ab Kapitel 3) unterstützt werden. Die Aufteilung in die SWE- und ITSM-Disziplin soll helfen, die Prozessschnittstellen für den in der Diplomarbeit zu konzipierenden Prozess zu definieren. Aus diesen verschiedenen Prozess- und auch Begriffsdefinitionen werden die für die Thematik der Diplomarbeit am besten geeigneten verwendet.

2.1 Lebenszyklusmodell und Klassifikationsschema

In diesem Abschnitt wird ein Lebenszyklusmodell von Software eingeführt. Dieses Lebenszyklusmodell ist in verschiedene Phasen unterteilt. Mithilfe dieses Lebenszyklusmodell erfolgt die Einteilung der ab Abschnitt 2.2 vorgestellten verwandten Arbeiten zum Release-Management anhand der Überdeckung der Phasen in entweder die SWE oder die ITSM Sicht. Zur Veranschaulichung dieser Einteilung wird weiterhin ein Klassifikationsschema eingeführt. Dieses ist in die vier Achsen SWE, ITSM, Prozessbeschreibung oder Prozessbewertung eingeteilt. Beschreibende Modelle definieren Verfahren und Tätigkeiten (siehe z.B. V-Modell im Abschnitt 2.2.1). Bewertende Modelle dienen zur Bewertung von Prozessen bzw. als Leitfaden für das Management (siehe z.B. *Software Maintenance Capability Maturity Model* im Abschnitt 2.3.7 oder *Control Objectives for Information and Related Technology* im Abschnitt 2.3.5). Die vorgestellten verwandten Arbeiten zum Release-Management werden jeweils in einen der Quadranten des Klassifikationsschemas eingeordnet. Das Klassifikationsschema soll als Orientierungshilfe dienen und wird deshalb immer am Anfang eines Abschnittes eingefügt.

2.1.1 Kanonisches Lebenszyklusmodell des Softwareentwicklungsprozesses

Als Grundlage zur Einordnung der Prozessaktivitäten dient das kanonische Lebenszyklusmodell von Software nach Mc Dermid [MD 93]. Dieses Lebenszyklusmodell besteht aus den in Abbildung 2.1 dargestellten Phasen. Die Phasen des Lebenszyklusmodells bestehen sowohl aus technischen Aufgaben als auch aus Managementaktivitäten. Die Anordnung der Phasen impliziert nicht notwendigerweise einen sequentiellen Ablauf der Tätigkeiten. Es sind auch inkrementelle, iterative oder parallele Abläufe dieser Phasen denkbar.

Da im Fokus dieser Diplomarbeit nicht die eigentliche Softwareentwicklung steht, werden die Phasen Anforderungsspezifikation bis *Unit* Test zu einer einzigen Phase Systementwicklung zusammengefasst. Die Integrationsphase wurde hier nicht der Systementwicklung zugeordnet. Grund hierfür sind neuere Trends in der Softwareentwicklung wie z.B. durch Webservices oder komponentenbasierte Softwaresysteme. Hier kann nicht mehr davon ausgegangen werden, dass alle Bestandteile eines *Webservices* oder eines komponentenbasierten Systems vom gleichen Hersteller kommen. Die Integration findet in diesen Fällen beim Kunden und nicht mehr beim Hersteller statt. Die Phasen Integrationstest, Akzeptanztest und operativer Test wurden aus Übersichtsgründen in einer Phase Tests zusammengefasst. Aus dem selben Grund wurde die Betriebsphase mit den Teilphasen Anforderungs-, Änderungs- und Release-Steuerung zu der Phase Betrieb inkl. Wartung zusammengefasst.

Die resultierende Darstellung (siehe Abbildung 2.2) wird in der Diplomarbeit verwendet, um Referenzmodelle und Beispiele aus den zitierten Quellen ab Abschnitt 2.2 einzuordnen. Bei Bedarf wird jedoch auf die nicht

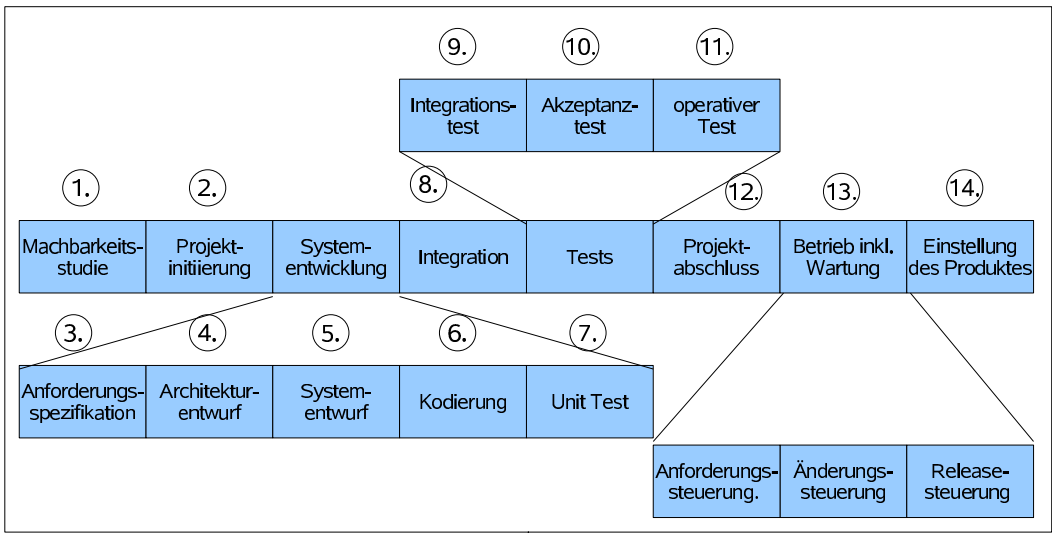


Abbildung 2.1: kanonisches Lebenszyklusmodell nach [MD 93]

komprimierten Phasen zurückgegriffen. Nachfolgend erfolgt die Beschreibung der Kernaufgaben der Phasen des Lebenszyklusmodells. Die Nummerierung bezieht sich hierbei auf die in Abbildung 2.1 dargestellten Nummern.

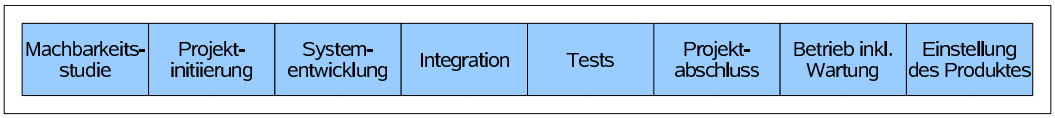


Abbildung 2.2: kanonischer Lebenszyklus im Kontext der Diplomarbeit

2.1.1.1 zu 1) Machbarkeitsstudie

Mithilfe der Machbarkeitsstudie werden technische Probleme erkannt und Entwurfsentscheidungen validiert. Sie dient als Grundlage der technischen Steuerung des Projektes. In der Machbarkeitsstudie werden die Bedürfnisse des Kunden untersucht und die Grenzen bestimmt, was mit den gegebenen Ressourcen umsetzbar ist. Das Ergebnis dieser Phase ist ein Abschlussbericht der Studie als Entscheidungsgrundlage für das weitere Vorgehen.

2.1.1.2 zu 2) Projektinitiierung

In dieser Phase werden durch das Projektmanagement die Arbeitsmethoden innerhalb des Projektes (Standards, Prozeduren, Tools) und die Projektsteuerungsmethoden (Metriken, Qualitätskriterien, Berichtswesen) festgelegt. Ergebnisse dieser Phase sind funktionale Anforderungen, Vertrag, Qualitätsplan, Konfigurationsmanagementplan, Erstentwurf, Teststrategie, die akzeptierte Arbeitsmethode innerhalb des Projektes und der Projektplan.

2.1.1.3 zu 3 - 7) Systementwicklung

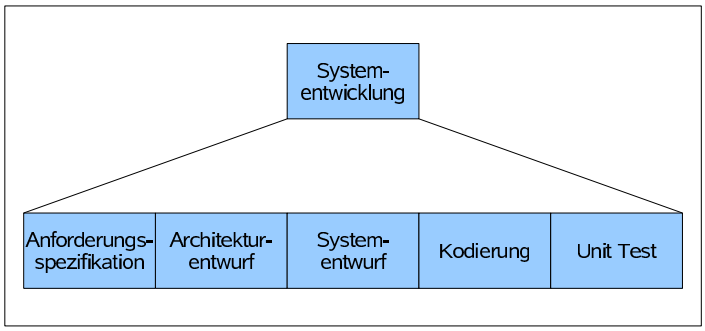


Abbildung 2.3: Aktivitäten innerhalb der Systemerstellung

Die Aufgabe der Systementwicklung ist die Erstellung eines fertigen Softwaresystems. Als Teilaufgaben fallen hier die Anforderungsspezifikation, der Architekturentwurf, der Systementwurf, die Kodierung und der Unit Test an (siehe Abbildung 2.3).

Als Ergebnis dieser Phase entsteht ein Modul. Dieses wird im nachfolgenden Schritt integriert.

2.1.1.4 zu 8) Integration

Voraussetzung dieser Phase sind entsprechende Testdaten und eine Menge von getesteten Modulen. Diese getestete Module werden anhand der festgelegten Integrationsstrategie (*Top-down*, kritischer Prozess zuerst, *Bottom-up*, etc.) zeitlich fertiggestellt und integriert. Ergebnisse dieser Phase sind ein integriertes Produkt zum Testen für ein unabhängiges Testteam und für das Kundensupportteam, Installationsmitteilungen und Testdaten zum unabhängigen Testen. Eine Ausgabe des Betriebs- und des Wartungshandbuches und die Analyse des Schulungsbedarfes muss bis zum Ende dieser Phase fertiggestellt sein.

2.1.1.5 zu 9 - 11) Tests

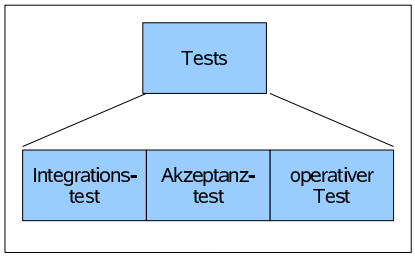


Abbildung 2.4: Testarten innerhalb der Testphase

In dieser Phase finden Integrationstests, Akzeptanztests und operative Tests statt (siehe Abbildung 2.4). Aus Gründen der Übersichtlichkeit wurden diese im Lebenszyklusmodell zusammengefasst. Nachfolgend werden die jeweiligen Testarten ausführlicher beschrieben.

2.1.1.5.1 Integrationstest Hier wird das Produkt durch ein unabhängiges Testteam getestet. Grundlagen für die Tests sind u.a. das Benutzerhandbuch. Diese Tests werden noch vor der endgültigen Installation beim Kunden durchgeführt. Ergebnis dieser Aktivität sind unabhängige Testergebnisse, Qualitätsberichte und ein lieferfähiges Produkt (*initial release*) inklusive Installationsdokumentation, Benutzerhandbuch, Betriebshandbuch und Wartungshandbuch.

2.1.1.5.2 Akzeptanztest Hier erfolgt die Evaluation und Abnahme des Produktes durch den Kunden anhand seiner festgelegten Abnahmekriterien. Die Akzeptanztests werden aus den Festlegungen in der Anforderungsspezifikation abgeleitet. Das Ergebnis dieser Schritte sind Abnahmeergebnisse, ein lieferbares Softwareprodukt (*packaged software product*), Installationsanweisungen und geschultes Personal für die Installation, abgenommene Benutzerhandbücher und Schulungsunterlagen.

2.1.1.5.3 Operativer Test In dieser Phase unterstützt das Entwicklungsteam den Kunden beim Einsatz neuer Software in der Anfangsphase. Es werden auch Qualitäts- und Zuverlässigkeitsmessungen durchgeführt. Diese dienen zur Bewertung des Erfolges des Produktes, ehe das Projekt abgeschlossen wird. Die durch die Umstellung notwendigen Installations- und Schulungsaktivitäten werden abgeschlossen. Anschließend wird sichergestellt, dass die Software und Hardware in Betrieb ist. Wenn der Softwareeinsatz zufriedenstellend ist, wird das Produkt vollständig dem Supportteam für Support und Wartung übergeben.

2.1.1.6 zu 12) Projektabschluss

Durch den Projektabschluss wird sichergestellt, dass alle Informationen für den Übergang zu nachfolgenden Projekten vorhanden sind, wenn diese Projekte im Zusammenhang mit dem abgeschlossenen stehen. Solche Informationen können nachfolgende Wartungsaufgaben oder Weiterentwicklungsprojekte sein. In dieser Phase werden auch quantitative Projektdaten zur weiteren Verbesserung für nachfolgende Projekte ausgewertet.

2.1.1.7 zu 13) Betrieb inklusive Wartung

In dieser Phase sind die Produkte im produktiven Einsatz beim Benutzer. Im Gegensatz zur Phase der Softwareentwicklung, die nach einer bestimmten Zeit abgeschlossen ist, ist die Betriebsphase in der Regel ohne Zeitgrenze.

Die Betriebsphase beinhaltet auch die Wartung. Sie ist eine eigene Teildisziplin. Durch die Wartung werden Softwareprodukte nach der Auslieferung modifiziert, um Fehler zu korrigieren (korrigierende Wartung), neue Funktionalitäten zu liefern (perfektible Wartung) oder um das Produkt an geänderte Bedingungen (adaptive Wartung) anzupassen. Oft wird hierfür auch der Begriff Softwareevolution benutzt. Dieser Begriff betont, dass nach erster Auslieferung eines Produktes meist eine Reihe von neuen Versionen dieses Produktes nachfolgen wird. Grundlage von Änderungen sind *Software Problem Reports* (SPR) . Gesteuert werden die Änderungen durch eine Reihe von Releases oder Updates. Jedes nachfolgende Release basiert ebenfalls auf den vorgestellten Entwicklungsschritten. Die Entwicklungsschritte können nun jedoch an die durch die Veränderung entstehenden Auswirkungen und Risiken angepasst werden.

Eine besondere Herausforderung besteht hier beim Einsatz von althergebrachter Software. Diese wurde möglicherweise vor langer Zeit ohne moderne SWE-Methoden produziert und hat dadurch evtl. Dokumentationslücken. Bei solcher so genannter geriatrischer Software muss speziell der *ripple effect* ausgeschlossen werden. Bei diesem Effekt wirken sich Änderungen in einem Teil des Systems unerwartet auf andere Teile aus [MD 93, Kap. 20]. Deswegen hat das Testen und Validieren in dieser Phase eine hohe Relevanz.

2.1.1.7.1 Anforderungsgesteuertes Wartungsmodell Abbildung 2.5 zeigt die Prozesse der Wartung als Teilprozesse der Betriebsphase.

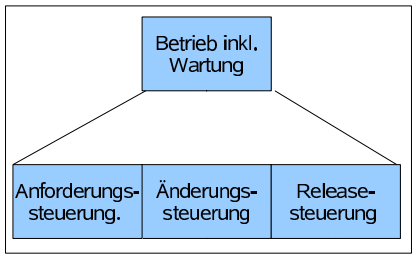


Abbildung 2.5: Betrieb und Wartung

Dieses Modell geht von der Initiierung des Wartungsprozesses durch den Benutzer mithilfe eines *Request for Change* (RfC) aus. Das Modell besteht aus den drei Prozessen Anforderungssteuerungsprozess, Änderungssteuerungsprozess und dem Release-Steuerungsprozess (*Request control, Change control, Release control*). Diese Prozesse werden nachfolgend detaillierter beschrieben.

Anforderungssteuerungsprozess Die Hauptaktivitäten im Anforderungssteuerungsprozess sind:

- die Informationsgewinnung zu jeder neuen Anforderung
- das Festlegen eines Vorgehens zur Kategorisierung der Anforderungen
- die Analyse der Auswirkungen, um jede Anforderung bezüglich Kosten und Nutzen zu bewerten
- die Zuteilung einer Priorität für jede Anforderung

Die Sammlung der Änderungsanforderungen bildet eine Grundlage zur Messung und Beurteilung der Wartungsaktivitäten. Zuständig für diese Informationsgewinnung ist laut Mc Dermid ein *Helpdesk*. Dieser soll bereits die RfC der Benutzer nach Änderungswünschen und Fehlern etc. vorab einteilen. Der Prozess sollte möglichst durch ein System (*Problem Monitoring System* (PMS)) unterstützt werden. Dieses überwacht jederzeit den momentanen Status des RfC. Das PMS liefert außerdem Statistiken zu Zeitverbrauch bis zur

Durchführung der Änderung, zu den meist betroffenen Programmen und zu den noch offenen Änderungsanforderungen. Jeder RfC wird entweder als perfektibel, adaptiv oder korrigierend (*perfective, adaptive or corrective* [MD 93, Kap. 20-5]) eingestuft.

Änderungssteuerungsprozess Im Änderungssteuerungsprozess werden folgende Aktivitäten durchgeführt:

- Auswahl bestimmter Änderungen aus der Prioritätsliste
- Nachstellen des Problems, falls eines vorliegt
- Analyse des Codes
- Entwurf der Änderungsanforderung inklusive Tests
- Qualitätssicherung

Mit der Analyse des Codes oder der Spezifikation soll ein Verständnis für das momentane System geschaffen werden, um die Änderungen entwerfen zu können. Die nötigen Änderungen werden dann im Entwurf festgehalten und Testfälle werden erstellt. Nach Genehmigung des RfC durch eine Qualitätssicherungsprozedur darf die eigentliche Änderung erst eingesetzt werden. Die Qualitätssicherung führt ähnlich wie im ursprünglichen Entwicklungsprozess Reviews oder Inspektionen durch.

Release-Steuerungsprozess Im Release-Steuerungsprozess erfolgen folgende Aktivitäten:

- Festlegung des Release-Umfangs
- Zusammenstellung eines neuen Releases
- Konfidenztests
- Verteilung
- Akzeptanztests

Innerhalb dieses Prozessschrittes wird festgelegt, welche Änderungen in das Release eingefügt werden. Auch hier wird der Prozess durch die Qualitätssicherung begleitet.

2.1.1.7.2 Management des Wartungsprozesses Die Managementaktivitäten im Wartungsprozess unterscheiden sich nicht von Managementaktivitäten in anderen Prozessen. Die Teilbereiche des Managements sind hier die Planung, die Steuerung und Kontrolle, die Organisation und Personalfragen.

Die Planung umfasst zum einen die Einhaltung der vom Management vorgegebenen Anforderungen und zum anderen die Planung, wie der Wartungsprozess an sich durchgeführt wird. Der Wartungsprozesses kann in einen so genannten Überlebensmodus (*survival mode*) oder in einem Produktionsmanagementmodus (*product management mode*) durchgeführt werden [MD 93, S.20/7]. Bei ersteren wird erwartet, dass das Personal die steigenden Änderungshäufigkeit ohne nennenswerte Probleme abfängt. Im Vordergrund steht das Zeitmaß. Das Wartungspersonal wird anhand der Anzahl der bewältigten Änderungen bewertet und im Vordergrund steht die Reduzierung des enormen (Arbeits-) Rückstandes. Bei letzteren steht die strategisch geplante Softwareevolution im Vordergrund. Hier werden Änderungen antizipiert und eingeplant. Die Wartbarkeit von Software wird aktiv aufrechterhalten, anstatt dass sie allmählich degeneriert. Das Wartungspersonal wird nicht mehr an der Anzahl der durchgeführten Änderungen, sondern anhand ihrer Unterstützung der Verbesserung der Wartbarkeit und Qualität von Software bewertet.

In beiden Fällen ist der Prozess jedoch änderungsgetrieben. Im letzteren Fall steht jedoch ein Lebenszyklusmodell mit definierten Qualitätssicherungmeilensteinen und Änderungssteuerung zur Verfügung. Auch wird die Verwendung von Technologien adressiert. Das kann zum Beispiel eine Umgebung mit vollständigem Konfigurationsmanagement und Versionskontrolle sein. Durch die Steuerung und Kontrolle des Wartungsprozesses werden grundlegende Informationen zur Bestimmung der Effektivität der Prozesse und zur Überwachung der Qualität der Software erzeugt. Schlüsselparameter sind hier u. a. Zeitdauer der Durchführung einer Änderung, durchschnittliche Prozessfehler je Lauf eines Programms, Anzahl der verbrauchten Personenstunden je

Wartungskategorie, durchschnittliche Bearbeitungsdauer je RfC etc. Durch Festlegen der Organisation werden klare Zuständigkeiten definiert. Ein wichtiger Bestandteil dieser Organisation ist das *Change Control Board* (CCB) . Es ist zuständig für die Priorisierung von RfC.

2.1.1.7.3 Qualitätssicherung in der Wartung In der Wartungsphase stellen geeignete Qualitätssicherungsmaßnahmen den Schutz der Systeme vor Qualitätsverschlechterungen aufgrund von Änderungseinführungen sicher. Das Aufgabenspektrum der Qualitätssicherung (QS) bezieht sich nicht nur auf die reine Fehlerbeseitigung im Rahmen von Testaktivitäten, sondern auch auf die Gestaltung aller Einflussmöglichkeiten auf die Qualität wie etwa verbesserte Spezifikations- und Entwurfstechniken oder Managementeinflüsse. Mithilfe von Reviews sollen möglichst viele Probleme frühstmöglich erkannt und aussortiert werden können. Vor allem bei geplanten Änderungen ist die Beurteilung von Auswirkungen auf das System wichtig. Für den Änderungsprozess ist eine klare Änderungsrichtlinie nötig. Beim Einführen neuer Funktionalitäten im Rahmen der Änderungen müssen neue Testfälle erzeugt werden. Die Testfälle müssen selbst ebenfalls überprüft werden. Regressionstests stellen sicher, dass durch die eingeführten Änderungen Bestehendes unverändert bleibt. Testwerkzeuge unterstützen an dieser Stelle den Qualitätssicherungsprozess.

2.1.1.8 zu 14) Einstellung des Produktes

In dieser Phase werden die zur Einstellung eines Produktes notwendigen Schritte festgelegt. Dazu wird ein Einstellungsplan erstellt. Bei erfolgter Einstellung wird ein Abschlussbericht erstellt. Er dokumentiert die zur erfolgreichen Einstellung des Produktes unternommenen Tätigkeiten, wie z.B. die Archivierung. Mit dem Abschlussbericht soll ein sauberer Übergang am Lebensende eines Produktes erfolgen.

2.1.2 Schema zur Klassifikation der Prozessmodelle

Abbildung 2.6 zeigt, wie sich das SWE und das ITSM in das vorgestellte Lebenszyklusmodell einordnen lassen. In dieser Abbildung werden die Aktivitäten des SWE rot und die Aktivitäten des ITSM blau dargestellt. Aktivitäten, die in beiden Bereichen durchgeführt werden, sind in der Abbildung farbig schraffiert dargestellt. Aus dieser Grafik ist ersichtlich, dass das SWE den Hauptfokus auf die Systementwicklung legt, während das ITSM den Schwerpunkt im operativen Betrieb hat. Managementaufgaben, wie etwa Machbarkeitsstudien oder Projektabschluss fallen sowohl im SWE als auch im ITSM an. Ebenso werden die verschiedenen Testaktivitäten ebenfalls von beiden Bereichen adressiert.

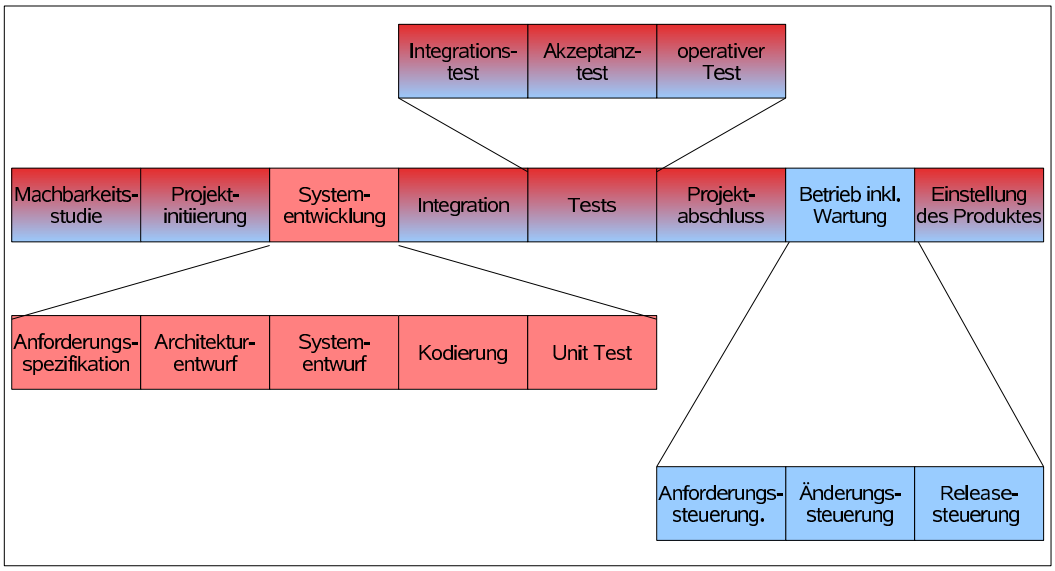


Abbildung 2.6: Einordnung von SWE und ITSM in das Lebenszyklusmodell

Zur allgemeinen Klassifikation wird Abbildung 2.7 herangezogen. Es erfolgt eine Unterteilung in die SWE- (Produkt- bzw. Projektfokus) und die ITSM-Sicht (Prozessfokus). Grund hierfür ist, dass ein Teil der untersuchten verwandten Arbeiten zum RM das RM als Bestandteil des SWE betrachtet. Die Prozesse des SWE enden üblicherweise mit der Erstellung eines auslieferbaren Softwareproduktes. Die IT-Servicesicht ist eine neuere Betrachtungsweise. Der Betrieb von Software und damit anfallende Wartungsaktivitäten werden von vielen SWE-Modellen nur oberflächlich adressiert. Die Aktivitäten des ITSM starten, wenn die Aktivitäten des SWE beendet sind.

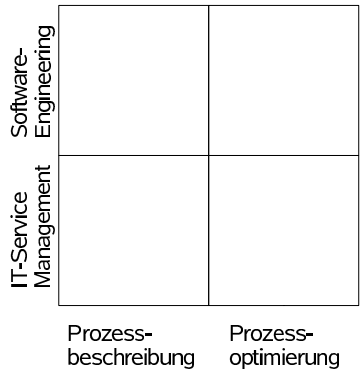


Abbildung 2.7: Schema zur Klassifikation von RM

Mit der Einordnung der beschriebenen Quellen aufgrund des Schemas in Abbildung 2.7 soll ein erster Überblick gewonnen werden. Die Einteilung in die SWE- und ITSM-Sicht erfolgt auf Basis des vorher beschriebenen Lebenszyklusmodell. Des Weiteren wird unterteilt, ob die beschriebenen Vorlagen zur Beschreibung von Prozessen dienen oder zur Bewertung von Prozessen. Aus Gründen der Übersichtlichkeit wird dieses Bild immer am Anfang der jeweiligen Beschreibung eingefügt.

Die Einordnung der verwandten Arbeiten hilft weiterhin beim Finden einer geeigneten Vorlage zur Konzeption des integrierten Release-Managements bei der HVBInfo. Die HVBInfo ist eine reine Betreiberorganisation. Aus diesen Gründen eignen sich für das zu konzipierende Release-Management im Klassifikationsschema links unten eingeordnete Referenzmodelle oder Verfahren. Anhand der rechts unten eingeordneten Referenzmodelle kann dann zukünftig das integrierte Release-Management kontinuierlich bewertet und dadurch optimiert werden. Es wird sich weiterhin bei der Beschreibung des derzeitigen RM bei der HVBInfo ab Abschnitt 3.4 zeigen, dass das momentane RM der HVBInfo dem SWE zuzuordnen ist, also im Klassifikationsschema rechts oben.

2.2 Release-Management als Bestandteil im Softwareengineering

In diesem Abschnitt werden Modelle vorgestellt, die sich dem SWE zuordnen lassen. Es erfolgt sowohl eine Beschreibung von klassischen SWE-Modellen, wie etwa dem V-Modell (siehe Abschnitt 2.2.1) oder *Rational Unified Process* (siehe Abschnitt 2.2.4), als auch von Prozessen, die Gegenstand der Forschung sind. Die klassischen Modelle decken meist einen größeren Teilbereich des Lebenszyklusmodells ab, als die Arbeiten aus der Forschung. Es wird gezeigt, dass der Release-Begriff im SWE gleichbedeutend ist mit der Version eines neuen Produktes. Allerdings ist das Release auch hier nicht eindeutig definiert. Bei Andersson [Ande 00] ist ein Release z.B. ein Software System, dass aus Kompontenen von verschiedenen Herstellern besteht, während es bei Ramakrishnan [Rama 04] eine vollständig konstruierte Version eines Softwareproduktes ist. Selbst innerhalb des SWE gibt es kein einheitliches Verständnis zum RM.

2.2.1 V-Modell 97

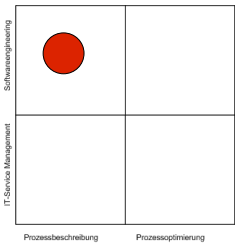


Abbildung 2.8: Einordnung V-Modell

Das V-Modell 97 (kurz V-Modell) [V-Modell, V-Modell-a, V-Modell-b, V-Modell-c] dient als Leitfaden für die Entwicklung von IT-Systemen. Es ist unterteilt in das Vorgehensmodell, die Methodenzuordnung und in funktionale Werkzeuganforderungen. Das Vorgehensmodell beschreibt Tätigkeiten und Ergebnisse des Softwareentwicklungsprozesses. Diesen Tätigkeiten sind jeweils die verantwortlichen Rollen zugeordnet. Jede Rolle beinhaltet eine Beschreibung der zur Erfüllung der Aufgaben geforderten Kenntnisse. Diese Festlegungen im Vorgehensmodell sind verbindlich für Entwicklungsvorhaben im militärischen Bereich und im Bereich der Bundesverwaltung. In der Methodenzuordnung werden Methoden vorgeschlagen, mit deren Hilfe die Aktivitäten des Vorgehensmodells durchgeführt werden können. Die funktionalen Werkzeuganforderungen beschreiben die geforderten Eigenschaften von Werkzeugen für die Systementwicklung.

Das Vorgehensmodells ist in folgende vier Submodelle unterteilt:

Qualitätssicherung: Maßnahmen des Submodells QS dienen zum Nachweis der Erfüllung von vorgegebenen Anforderungen, zur präventiven Vermeidung von Mängeln und zur Sicherstellung der Prozessqualität. Adressiert werden sowohl konstruktive als auch analytische QS-Maßnahmen. Eine konstruktive Maßnahme ist z.B. die Unterstützung der Softwareentwicklung durch Werkzeuge. Mit analytischen Maßnahmen wird die Qualität von Prüfgegenständen geprüft und bewertet.

Konfigurationsmanagement: Das Konfigurationsmanagement (KM) umfasst die Planung des KM, die Katalogisierung und Versionsverwaltung von Produkten und das Änderungsmanagement. Durch das KM wird die eindeutige Identifizierbarkeit eines Produktes sichergestellt. Unterschiede durch Änderungen zu früheren Versionen sind jederzeit erkennbar und es kann jederzeit auf frühere Versionen zurückgegriffen werden. Änderungen werden so nachvollziehbar und die Integrität ist sichergestellt.

Projektmanagement: Im Submodell Projektmanagement (PM) werden die Aktivitäten von der Projektinitialisierung bis zum Projektabschluss durchgeführt. Weitere Aufgaben in diesem Modell sind z.B. Kosten- und Nutzenanalyse, Risikomanagement, Informations- und Berichtswesen, Schulung und Einarbeitung, Ressourcenbereitstellung und der Projektabschluss.

Systemerstellung: Das Submodell Systemerstellung (SE) ist aufgegliedert in eine Systemebene und in eine Ebene der Software- und Hardwareeinheiten. Die Abstimmung von Anwender und Realisierer erfolgt auf Basis der Systemebene. Die Umsetzung dieses Systems erfolgt auf der Ebene der Software- und Hardwareeinheiten. Aktivitäten innerhalb dieses Submodells sind u.a. System-Anforderungsanalyse, Systementwurf, Systemintegration und Überleitung in die Nutzung (vgl. Abbildung 2.9).

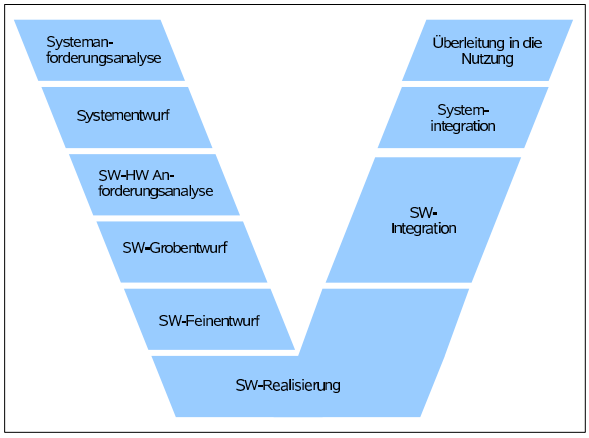


Abbildung 2.9: Submodell Systemerstellung des V-Modells nach [V-Modell-c]

2.2.1.1 Release-Management im V-Modell

Im V-Modell [V-Modell] ist das RM ein Bestandteil des Submodells Konfigurationsmanagement. Durch die Prozesse des RM werden Konfigurationseinheiten nach bestimmten Kriterien bzw. Optionen konfiguriert und Produkte für die Auslieferung vorbereitet. Das beinhaltet auch die Erstellung eines Datenträgers und einer Installationsprozedur. RM ist hierbei die einzige Auslieferstelle. Somit wird gewährleistet, dass nur Produkte mit abgestimmten Änderungen ausgeliefert werden. Ebenso hat das Release-Management den Überblick über die gesamten Auslieferungen.

Die Tätigkeit der Einführung von fertiggestellten Systemen ist dem Submodell Systemerstellung „SE 9 Überleitung in die Nutzung“ zugeordnet. Hier wird dafür gesorgt, dass ein fertiggestelltes System an der vorgesehenen Einsatzstelle installiert und in Betrieb genommen wird (siehe Abbildung 2.10). Dafür sind folgende Subaktivitäten definiert:

SE 9.1 Beitrag zur Einführungsunterstützung leisten: Hierzu gehören z. B. Schulungsmaßnahmen, Planung des Vorgehens bei der Überleitung z.B. Parallelbetrieb, Übergangsregelungen. Verantwortlich für diese Subaktivität ist der Systembetreuer. Er wird unterstützt durch den technischen Autor und den Anwender.

SE 9.2 System installieren: Hier wird das System durch den Systembetreuer durch folgende Aktionen in einen betriebsbereiten Zustand gebracht: System entpacken, Verbinden der Systemteile, Netzwerke anschließen, Funktionsprüfung des fertig zusamengestellten Systems etc. Der Systemumgebungsbetreuer ist bei dieser Aufgabe für die Umgebung zuständig.

SE 9.3 In Betrieb nehmen: Bei dieser Aktivität wird das installierte System durch den Systembetreuer in Betrieb genommen. Vor der Inbetriebnahme erfolgt eine Abnahmeprüfung und eine Einweisung des Personals hinsichtlich der Inbetriebnahme.

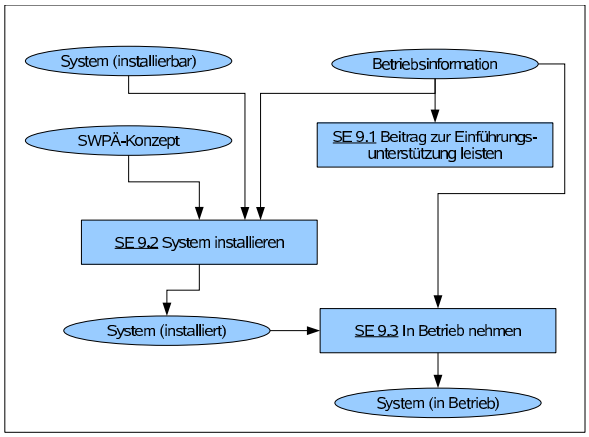


Abbildung 2.10: Überleitung in die Nutzung, [V-Modell]

2.2.1.2 Einordnung des V-Modells in das Lebenszyklusmodell

Das V-Modell ist als SWE-Modell einzuordnen. Es beschreibt Tätigkeiten zur Erstellung von Software-/Hardware-Systemen. Für die Machbarkeitsstudie ist im V-Modell keine eigene Phase vorgesehen. Innerhalb des Submodells PM gibt es die Kosten- und Nutzenanalyse, bei der für die möglichen Lösungsvorschläge eine Kosten- und Nutzenrechnung erfolgt. Außerdem werden durch das Risikomanagement mögliche Risiken im Projekt gehandhabt. Das V-Modell unterstützt noch die Einführung von fertigen Systemen in den Betrieb. Die eigentliche Betriebsphase wird allerdings nicht mehr beschrieben. Die Phase Einstellung von Produkten ist im V-Modell nicht vorgesehen. Der Release-Begriff wird in diesem Modell nicht explizit verwendet. Das Produkt der Subaktivität SE ist ein installierbares System einschließlich zugehöriger Dokumentation wie etwa Betriebsinformationen. Abbildung 2.11 zeigt, welche Phasen des Lebenszyklusmodells vom V-Modell abgedeckt werden.

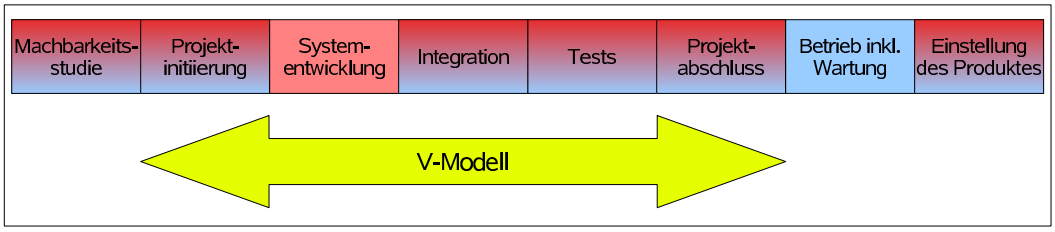


Abbildung 2.11: Einordnung des V-Modells in das Lebenszyklusmodell

2.2.2 V-Modell XT

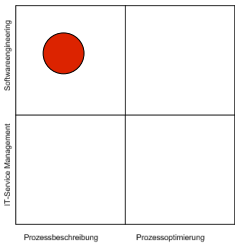


Abbildung 2.12: Einordnung V-Modell XT

Das V-Modell XT [V-Modell XT] ist eine Überarbeitung und Erweiterung des V-Modells (siehe Abschnitt 2.2.1). XT steht hier für *extreme tailoring* oder *extendable*. Abbildung 2.13 zeigt Entscheidungspunkte und Projektdurchführungsstrategien des V-Modells XT. Erweitert wurde es im Vergleich zum V-Modell hinsichtlich Anpassungsmöglichkeiten an verschiedene Projekte und Organisationen, hinsichtlich der Adaptionsmöglichkeiten des Modelles selbst und hinsichtlich der Berücksichtigung neuer Technologien, Vorschriften oder Normen.

Anstatt der Aufteilung des Vorgehensmodells in monolithische Submodelle stehen jetzt aufeinander aufbauende Vorgehensbausteine im Zentrum. In einem Vorgehensbaustein werden alle inhaltlich zusammengehörenden Produkte, Aktivitäten und Rollen vereint. Für alle Projekte gleich ist der definierter Kern mit den Bausteinen Projektmanagement, Qualitätssicherung, Konfigurationsmanagement und Problem- und Änderungsmanagement. Alle anderen Projektarten werden durch die geeignete Auswahl der entsprechenden Bausteine unterstützt. Diese Zusammenstellung wird als *Tailoring* bezeichnet.

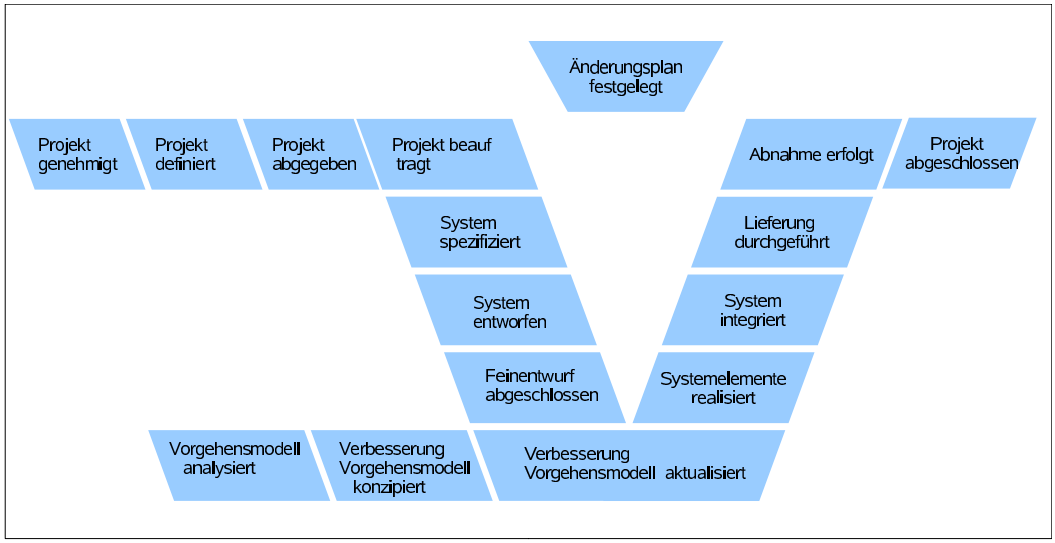


Abbildung 2.13: Entscheidungspunkte und Projektdurchführungsstrategien im V-Modells XT nach [V-Modell XT]

2.2.2.1 Release-Management im V-Modell XT

Beim V-Modell XT ist wie im V-Modell der Vorgehensbaustein Konfigurationsmanagement dafür zuständig, dass jedes Produkt anhand funktionaler oder äußerer Merkmale identifiziert werden kann. Hier ist jedoch das Konfigurationsmanagement nicht auf die Dauer der Systemerstellung begrenzt, sondern bezieht sich auch auf

den Zeitraum der Nutzung. Für Änderungen an der Konfiguration ist der Vorgehensbaustein Problem- und Änderungsmanagement zuständig. Durch diesen werden Änderungswünsche, Fehler und Probleme behandelt und gelöst. Auslöser für diesen Prozess sind Problemmeldungen oder Änderungsanträge von Nutzern, Entwicklern oder Auftraggebern. Im Rahmen des Vorgehensbausteins Problem- und Änderungsmanagement werden diese Meldungen beurteilt. Dann wird über die Durchführung von Änderungen in einer Änderungsentscheidung entschieden. Die zuständige Rolle hierfür wird Änderungssteuerungsgruppe oder auch *Change Control Board* genannt..

Innerhalb der Aktivität Systementwurf wird das Migrationskonzept erstellt. Das Migrationskonzept wird vom Systemarchitekt mithilfe des Systemintegrators erstellt. Das Konzept legt fest, wie die Migration durchgeführt wird, was zu migrieren ist und wer dafür zuständig ist. Ebenso wird eine Roll-back-Strategie erstellt.

2.2.2.2 Einordnung des V-Modells XT in das Lebenszyklusmodell

Abbildung 2.14 zeigt die Abdeckung des vorgestellten Lebenszyklusmodell mit dem V-Modell XT. Im Vergleich zum V-Modell adressiert das V-Modell auch die Einstellung eines Systems und den Prozess der Anforderungs- und Änderungssteuerung. Das V-Modell XT betrachtet allerdings nicht den eigentlichen Betrieb des Systems. Der Prozess der Anforderungs- und Änderungssteuerung verstehen sich hier als Steuerung von Anforderungen und Änderungen im (Weiter-)Entwicklungsprozess beim jeweiligen Softwarehersteller. Diese neuen Anforderungen und Änderungswünsche resultieren dann u.a. vom Anwender dieser Software durch die Benutzung derselben.

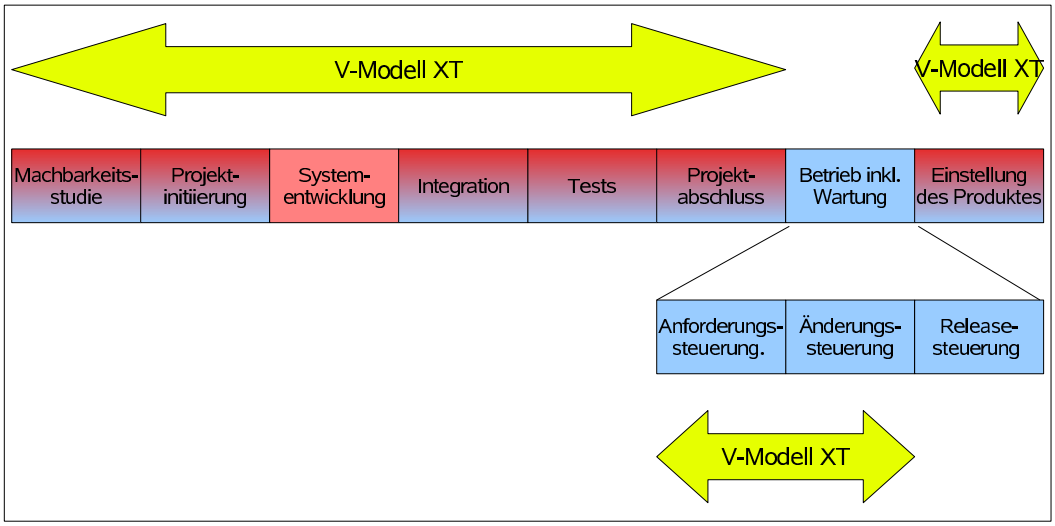


Abbildung 2.14: Einordnung des V-Modells XT in das Lebenszyklusmodell

2.2.3 Microsoft Solutions Framework

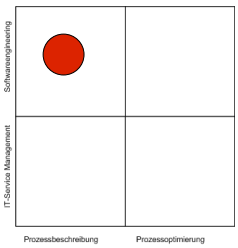


Abbildung 2.15: Einordnung von MSF

Das *Microsoft Solutions Framework* (MSF) [MSF] ist ein Rahmenwerk für die projektbasierte Erstellung von Softwarelösungen der Firma Microsoft.

Es besteht aus mehreren Komponenten, die je nach Bedarf zusammengestellt werden können. Diese Komponenten sind die MSF Grundprinzipien, MSF Modelle, MSF Disziplinen, MSF Grundkonzepte, MSF bewährte Praktiken und MSF Empfehlungen. Die wichtigsten werden nachfolgend detaillierter beschrieben.

Schlüsselqualitätsziel	Rolle
Lieferung innerhalb der Projektrahmenbedingungen	Programm-Management
Lieferung innerhalb der Produktspezifikationen	Entwicklung
Release nach Abhandlung aller Aspekte	Test
Reibungsloses Deployment und kontinuierliches Management	RM
Erweiterte Benutzerperformanz	*User Experience*
Zufriedene Kunden	Produkt-Management

Tabelle 2.1: Schlüsselqualitätsziele und zugeordnete Rollen in MSF

2.2.3.1 MSF Grundprinzipien

Das MSF Rahmenwerk beruht auf allgemeingültigen Grundprinzipien (Werte und Standards). Dazu gehören z.B. das Fördern einer offenen Kommunikation, das Arbeiten an einer gemeinsamen Vision oder das Motto „bleib beweglich, erwarte Änderungen“.

2.2.3.2 MSF Modelle

Diese Komponente beinhaltet das Team- und Prozessmodell zur schematischen Beschreibung der Organisation der Projektteams und der Prozesse.

2.2.3.2.1 MSF Teammodell Das MSF Teammodell [MSFb] definiert die Rollen und Verantwortlichkeiten eines Teams.

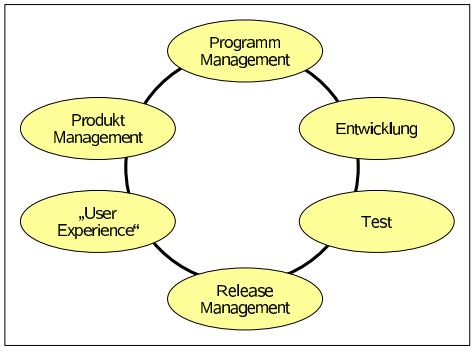


Abbildung 2.16: Rollen des MSF Teammodells

Es basiert auf der Voraussetzung, dass jedes Technologieprojekt definierte Schlüsselqualitätsziele erreichen muss, damit es erfolgreich ist. Dazu sind bestimmte Qualifikationen nötig. Diese werden einer Rolle zugeordnet (siehe Tabelle 2.1). Die zugehörigen Wissensgebiete werden *Functional Areas* genannt und bestimmen die Domäne jeder Rolle. Die Rollen im Teammodell zeigt Abbildung 2.16.

2.2.3.2.2 MSF Prozess Modell Das MSF Prozess Modell [MSFa] definiert Aktivitäten des Projektlebenszyklus von der Initiierung bis zum Einsatz. Diese Komponente kombiniert das Konzept der Meilensteine des traditionellen Wasserfallmodells und das Konzept der inkrementellen iterativen Entwicklung des Spiralmodells. Jede Phase endet mit einem Meilenstein. Dadurch kann überprüft werden, ob das Phasenziel erreicht wurde. Innerhalb jeder Phase können auch Meilensteine als Zwischenergebnis definiert werden. Diese dienen als Fortschrittsindikatoren im Prozess.

Der Zyklus startet mit der Festlegung einer Vision als Meilenstein. Dann folgt die Planungsphase mit dem Meilenstein genehmigter Projektplan, die Entwicklungsphase mit dem Meilenstein fertiger *Scope*, die Stabilisierungsphase mit dem genehmigten Release-Meilenstein und die Deployment-Phase mit dem Meilenstein abgeschlossenes Deployment. Am Ende eines Zyklus steht eine bestimmte Version eines Release (siehe Abbildung 2.17).

Dieses wird durch die nachfolgenden Iterationen weiterentwickelt. Das Konfigurationsmanagement steuert und kontrolliert den Zustand der verschiedenen Projektelemente wie z.B. Code, Dokumente, Pläne etc. Dadurch wird sichergestellt, dass jederzeit das Zurücksetzen auf eine frühere Version möglich ist.

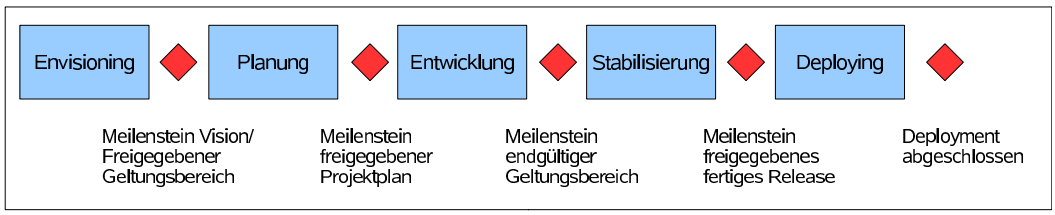


Abbildung 2.17: MSF Prozessmodell

2.2.3.3 MSF Disziplinen

Die MSF Disziplinen Projektmanagement, Risikomanagement und Readiness-Management stellen Methoden, Terminologien und Ansätze bereit. Dadurch wird die Funktionsfähigkeit des MSF Team- und Prozessmodells optimiert.

2.2.3.4 Release-Management bei MSF

Das RM ist eine Rolle des MSF Teammodells. Es ist zuständig für eine reibungslose Versorgung und einen unterbrechungsfreien Betrieb. Das RM fungiert als Schnittstelle zwischen der Entwicklung und der Betriebsgruppe. Es handhabt die Werkzeugauswahl für Release-Tätigkeiten, die Schulung des Betriebspersonals, den Support für die Pilotversorgung und plant die Versorgung der Lösung in die Produktion.

Im MSF Prozessmodell beinhaltet die Release-Planung die Festlegung der Funktionalitäten, die durch eine bestimmte Version ausgeliefert werden. Änderungen an der Spezifikation werden durch die Änderungskontrolle organisiert. In der Stabilisierungsphase wird eine Lösung getestet, deren *Features* komplett sind. Die Tests werden in einer realitätsnahen Umgebung durchgeführt. Hauptaufgaben sind hier das Beseitigen von Fehlern und die Vorbereitung des Releases. Wenn die Lösung als stabil genug betrachtet wird, erfolgt die Versorgung einer Pilotgruppe. Die Deployment-Phase beinhaltet Prozeduren für die Installation und Deinstallation von Hard- und Software, für automatische Deployment-Werkzeuge und für Notfall-Roll-backs.

2.2.3.5 Einordnung von MSF in das Lebenszyklusmodell

Das MSF ist ein SWE Modell. Es adressiert nicht die eigentliche Betriebsphase. Abbildung 2.18 zeigt, welche Bereiche des Lebenszyklusmodells vom MSF abgedeckt werden. Für die Betriebsphase wurde von Microsoft das Microsoft Operations Framework (siehe Abschnitt 2.3.1) als Ergänzung entwickelt.

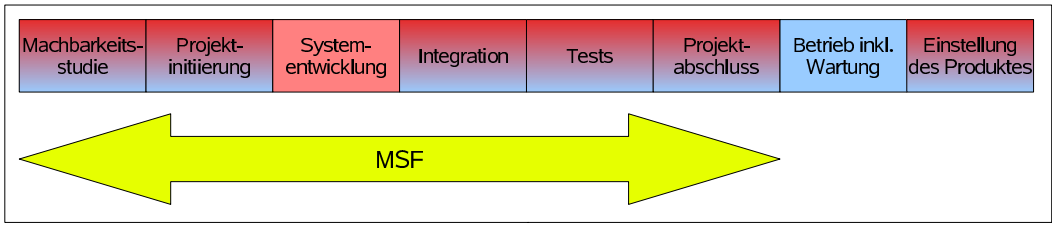


Abbildung 2.18: Einordnung des MSF in das Lebenszyklusmodell

2.2.4 Rational Unified Process

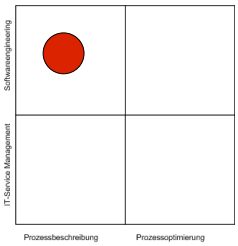


Abbildung 2.19: Einordnung RUP

Der *Rational Unified Process* (RUP) [RUP, Kruc 99] ist der SWE-Prozess der Firma Rational. Rational wurde im Jahr 2003 von IBM übernommen. RUP basiert auf dem *Unified Process* von Booch, Rumbaugh und Jacobson. Diese legten auch die Grundlage für die *Unified Modelling Language* (UML) . Aus diesem Grund verwendet RUP UML als Methode [WSK 05]. RUP versteht sich selbst als Prozessprodukt indem das Produkt „RUP“ selbst von Rational weiterentwickelt und gepflegt wird.

Der Prozess basiert auf sechs *Best Practices* der Softwareentwicklung. Diese sind die iterative Softwareentwicklung, Verwendung komponentenbasierter Architekturen, Prüfung der Softwarequalität, kontrolliertes Änderungsmanagement, Verwalten von Anforderungen und die visuelle Softwaremodellierung (mit UML). RUP beinhaltet als Aufgaben die Unterstützung von Teammitgliedern bei der Abwicklung ihrer Aktivitäten. Des Weiteren werden die zu entwickelnden Artefakte spezifiziert. RUP bietet außerdem Kriterien zur Überwachung und Bewertung der Produkte und Aktivitäten an.

Der komplette Prozess wird in neun Handbüchern beschrieben. Das Prozesshandbuch etwa beschreibt, was, wann, wie und von wem zu tun ist. Das Artefaktehandbuch beschreibt die Input- und Output-Produkte des Prozesses.

Die Gesamtstruktur von RUP hat zwei Dimensionen (siehe Abbildung 2.20). In der ersten Dimension werden die Aspekte des Lebenszyklusses dargestellt. Der Lebenszyklus beinhaltet die Konzeptphase (*Inception*), die Entwurfsphase (*Elaboration*), die Konstruktionsphase und die Übergangsphase (*Transition*). Jede dieser Phasen wird mit einem definierten Meilenstein abgeschlossen. In der zweiten Dimension werden die statischen Aspekte des Prozesses dargestellt. Sie werden durch Komponenten, *Workflows*, *Worker* und Artefakte beschrieben. Diese Modellierungselemente stellen im Prozess dar, wer (*Worker*), was (Artefakt), wann (*Workflows*) und wie (Aktivität) tut.

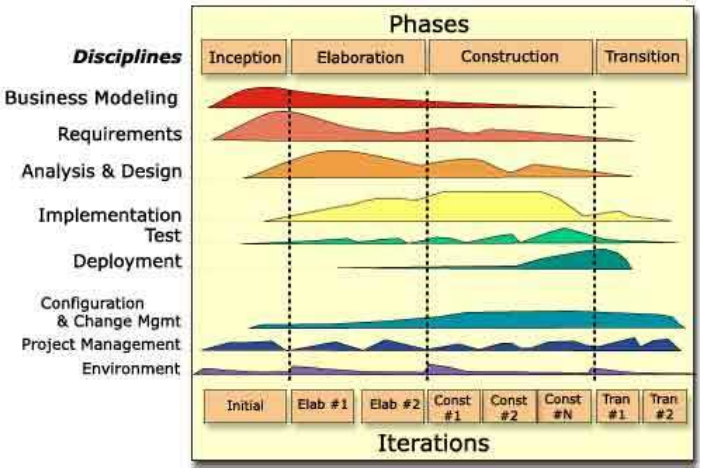


Abbildung 2.20: Dimensionen von RUP aus [RUP]

Ein Artefakt ist in diesem Zusammenhang ein konkretes Produkt oder eine konkrete Information. Diese Artefakte werden durch den jeweiligen Prozess erstellt, geändert oder genutzt. Die Artefakte sind Gegenstand der Versionsverwaltung und des Konfigurationsmanagements. Die *Workflows* werden wiederum unterteilt in *Core-Workflows*, *Iterations-Workflows* und *Workflow-Details*. Die *Core-Workflows* sind ebenfalls wieder unterteilt in z.B. Realisierungs-, Test- oder Verteilungs-Workflow und in unterstützende Workflows wie Konfigurations- und Änderungsmanagement-Workflow und Umgebungs-Workflow.

RUP basiert auf einer iterativen Softwareentwicklung. Die fertiggestellten Komponenten werden schrittweise integriert und getestet. Oft werden bei der iterativen Entwicklung die entstandenen Produkte wieder modifiziert. Aus diesem Grund hat das Änderungsmanagement hier die Aufgabe, alle Änderungen im Hinblick auf

Anforderungen, Entwurf und Realisierung zu verwalten. Des Weiteren hat es die Aufgabe, aufgetretene Fehler oder Missverständnisse innerhalb des Projektes zu verfolgen.

2.2.4.1 Release-Management beim Rational Unified Process

Zwischen den Iterationen werden zur Messung des Projektfortschrittes Meilensteine festgelegt. Die Konzeptphase schliesst mit dem *Life-Cycle Objective-* (LCO) Meilenstein, die Entwurfsphase mit dem *Life-Cycle Architecture-* (LCA) Meilenstein, die Konstruktionsphase mit dem *Initial Operational Capability-* (IOC) Meilensein, und die Übergangsphase mit dem Produkt-Release- (PR) Meilenstein ab.
Nach dem ersten Durchlauf dieser vier Phasen ist eine Softwaregeneration fertiggestellt. Diese bildet die Grundlage für evtl. nachfolgende Iterationen. Dann jedoch mit einer anderen Gewichtung der Phasen.

Der PR der ersten Iteration wird hier als internes Release bezeichnet. Diesem folgen in jeder nochfolgenden Iteration weitere PR-Meilensteine. Ein weiteres speziell gekennzeichnetes Release ist das erste externe Release oder Beta-Release. Diesem folgt nach der letzten Iteration das so genannte fertige Release. Das Beta-Release entsteht als Meilenstein am Ende der Konstruktionsphase. Es dient zur Evaluation des Einsatzes des Produktes beim Benutzer. Die so genannte *Baseline* entspricht einem Release, welches einem Review unterworfen und genehmigt wurde Die *Baseline* darf anschließend nur noch durch die Änderungs- oder Konfigurationskontrolle verändert werden. Das Ziel der Entwurfsphase ist die Festlegung einer *Baseline*.

In der Übergangsphase wird das Produkt in die Umgebung des Benutzers integriert. Hier wird noch mit auftretenden Fehlern oder fehlenden Funktionalitäten gerechnet. Dadurch wird ein weiteres Release nötig sein. Wenn die *Baseline* die vollständige Version erlangt hat, ist diese Phase mit dem PR als Meilenstein abgeschlossen. Der Verteilungs-Workflow sorgt für die Auslieferung des Produktes beim Endanwender. Aktivitäten hierzu sind:

- das Erstellen und das Zusammenführen eines externen Software-Releases
- die Paketbildung
- die Verteilung und Installation der Software
- die Planung und die Durchführung von Beta-Tests
- der Parallelbetrieb mit den alten Systemen als Vergleichsgrundlage
- das Training der Benutzer und Administratoren
- die formelle Akzeptanz durch den Benutzer.

Zuständige *Worker* sind hier der Verteilungsmanager für das Planen und Managen der Verteilung, der Realisierer für die Produktion des Software-Releases, der technische Autor für die Erstellung von Handbüchern und der Autor für die Erstellung der Schulungsunterlagen. Artefakte in diesem Prozess sind der Verteilungsplan, die Benutzerhandbücher und Schulungsunterlagen.

2.2.4.2 Einordnung vom Rational Unified Process in das Lebenszyklusmodell

RUP lässt sich wie in Abbildung 2.21 in das Lebenszyklusmodell einordnen. Der Prozess endet mit der Einführung eines Produktes beim Kunden. Der eigentliche Betrieb wird hier nicht beschrieben.

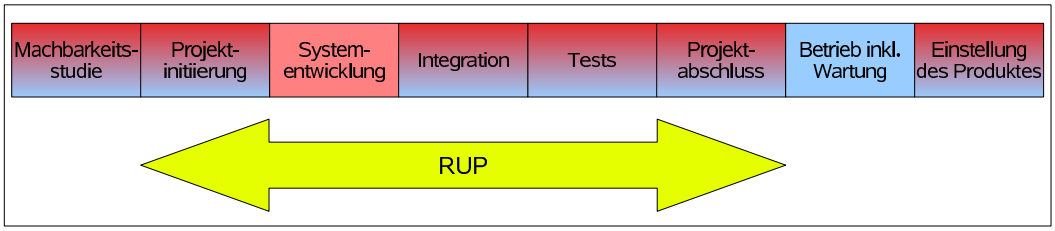


Abbildung 2.21: Einordnung von RUP in das Lebenszyklusmodell

2.2.5 Capability Maturity Model Integration

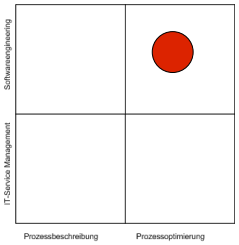


Abbildung 2.22: Einordnung CMMI

Das *Capability Maturity Model Integration* (CMMI) des Software Engineering Institute (SEI) der Carnegie Mellon University ist ein Modell zur Verbesserung von Software- und Systementwicklungsprozessen [Cmu02]. Mithilfe von CMMI haben Auftraggeber die Möglichkeit Prozesse beim Hersteller zu bewerten. Der Hersteller selbst kann CMMI zur Optimierung seiner eigenen Entwicklungsprozesse verwenden.

Der Entwicklungsprozess wird in Prozessgebiete strukturiert. Es gibt die Prozessgebiete Prozessmanagement, Projektmanagement, *Engineering* und Support. Diese Prozessgebiete sind weiter unterteilt. Das Support-Prozessgebiet ist z.B. weiter unterteilt in das Konfigurationsmanagement, Prozess- und Produktqualitätssicherung, Messen und Analyse, *Decision Analysis and Resolution*, organisatorische Integrationsumgebung und *Kausal Analysis and Resolution*. Die Realisierung eines Prozessgebietes wird durch das Erreichen messbarer Ziele festgestellt. Bei der Bewertung kann ein Unternehmen eine von fünf möglichen Reifegradstufen erreichen. CMMI definiert hierfür die jeweils notwendigen Prozessgebiete zur Erreichung einer bestimmten Stufe.

2.2.5.1 Release-Management bei CMMI

Beim CMMI ist innerhalb des Support-Prozessgebietes das Konfigurationsmanagement angesiedelt. Aufgaben des Konfigurationsmanagements sind hier u.a. die Steuerung von Änderungen an *Configuration Items* (CI), die Erstellung und Bereitstellung von Spezifikationen für das Erstellen von Produkten aus dem Konfigurationsmanagement-System oder die Integritätserhaltung von *Baselines*.

Die Änderung an *Baselines* und das Freigeben von Produkten, die aus den Konfigurationsmanagement-System erstellt werden, wird durch die Konfigurationslenkung, das Änderungsmanagement und durch Auditierung des Konfigurationsmanagements systematisch gesteuert und überwacht.

Eine *Baseline* ist hier eine Menge von Spezifikationen oder Produkten, die einem formalen Review unterlagen und als Basis für die weitere Entwicklung dienen. Änderungen dürfen hier nur noch durch den Change-Prozess erfolgen. Eine *Baseline* stellt die Zuordnung eines Identifikators an ein CI und seiner zugehörigen Entitäten dar.

2.2.5.2 Einordnung von CMMI in das Lebenszyklusmodell

Der Produktlebenszyklus beginnt bei CMMI mit der Vision und endet mit der Einstellung eines Produktes. CMMI geht davon aus, dass ein Hersteller für verschiedene Kunden entwickelt. Aus diesem Grund ist evtl. die Definition verschiedener Produktlebenszyklen nötig. Ein Lebenszyklus kann so z.B. aus den Phasen Konzeption bzw. Vision, Machbarkeitsstudie, Entwurf, Entwicklung, Produktion und Produkteinstellung bestehen. Nachdem die Definition der Lebenszyklusphasen dem Hersteller überlassen bleibt, lässt sich CMMI im Ganzen wie in Abbildung 2.23 dargestellt, in das Lebenszyklusmodell einordnen.

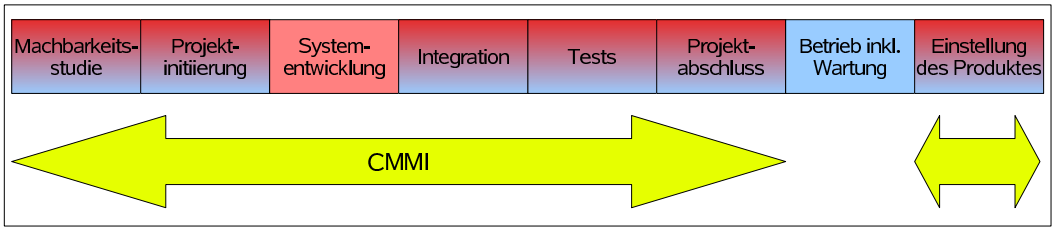


Abbildung 2.23: Einordnung von CMMI in das Lebenszyklusmodell

2.2.6 Weitere Beispiele für Release-Management innerhalb des Softwareengineerings

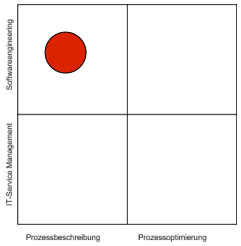


Abbildung 2.24: Einordnung der Prozesse

In diesem Abschnitt erfolgen weitere Beispiele von Definitionen des RM als als Bestandteil innerhalb des SWE. Alle aufgeführten Beispiele sind beschreibende Prozesse und somit wie in Abbildung 2.24 zu klassifizieren.

Bei einem Teil der Beispiele erfolgt keine Einordnung in das Lebenszyklusmodell, weil sie nur einen minimalen Ausschnitt des Lebenszyklusmodells adressieren.

2.2.6.1 Release-Management als Bestandteil der Softwarelogistik

Bei Greefhorst [Gree 00] ist das Release-Management ein Bestandteil der Softwarelogistik. Sie ist eine Disziplin im Softwareengineering und beinhaltet zusätzlich das *Derivation Management* und das Konfigurationsmanagement. Abbildung 2.25 zeigt den Prozessablauf mit den zugehörenden Datenbanken.

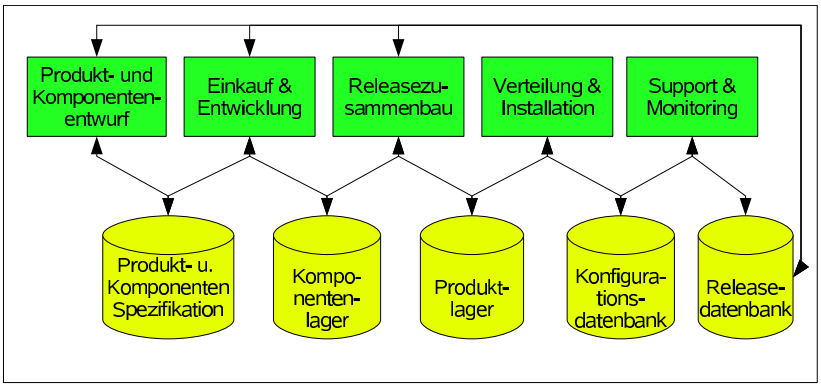


Abbildung 2.25: Softwarelogistikprozesse und Datenbanken, [Gree 00]

2.2.6.2 Release-Management als Teil des Product Software Lifecycle Management

Bei Jansen et. al. [Jans 05, Jan05] ist der Release-Prozess ein Teilprozess des *Product Software Lifecycle Management*. Der Release-Prozess umfasst hier die Erstellung eines Releases als neue Version eines Softwareproduktes eines bestimmten Herstellers und die Interaktionen mit dessen Kunden. Tätigkeiten sind hier das Management des Release-Prozesses und das *Produktknowledgemanagement*. Zum Release-Prozessmanagement gehört wiederum ein formeller Release-Ablauf. Dieser beschreibt schrittweise die Erzeugung eines Releases.

Es wird davon ausgegangen, dass die Kunden mit jeweils eigenen Varianten einer Applikation beliefert werden müssen. Daraus resultiert eine hohe Komplexität des Release- und Deployment-Prozesses. Das führt in der Praxis oft zu einem höheren Zeit- und Ressourcenverbrauch als ursprünglich geplant. Zur Unterstützung des

Prozesses wird ein Rahmenwerk auf der Basis von Zustandsmodellen vorgeschlagen. Eine Komponente kann dann z.B.*source, built, deployed and running* als Zustandsabfolge haben. Das Problem von möglichen inkompatiblen Komponentenversionen kann durch eine geeignete Logik gelöst werden. Sollten dennoch Probleme auftreten, werden durch ein Kommunikationsrahmenwerk die Softwareentwickler automatisch per E-Mail informiert.

Der Informationsfluss zwischen Softwarehersteller und dessen Kunden wird durch das *Customer Configuration Updating* (CCU) geregelt (siehe Abbildung 2.26). Prozesse innerhalb von CCU sind *Delivery, Deployment und Activation & Usage*.

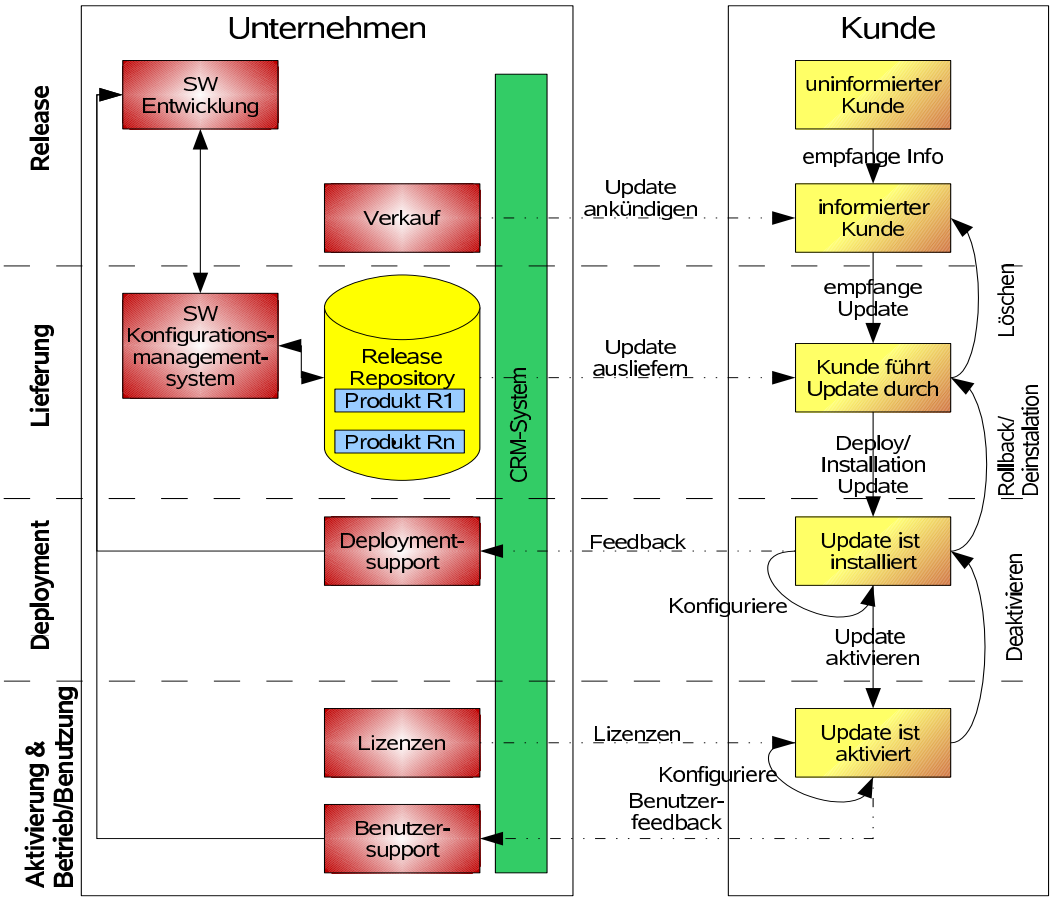


Abbildung 2.26: Customer Configuration Updating (CCU) aus [Jan05]

Das *Produktknowledgemanagement* informiert alle beteiligten Mitarbeiter über neue Releases. Durch transparente Informationsprozesse soll die Beziehung zwischen den Entwicklungs- und Verkaufsabteilungen verbessert werden. *Produktknowledgemanagement* ist auch für die Speicherung von freigegebener Software in einer Datenbank zuständig. Dadurch können Kunden jederzeit auf eine alte Version zurücksetzten. Laut den Autoren sind folgende Voraussetzungen für einen hohen Reifegrad des CCU-Prozesses nötig:

Die Software muss durch ein *Product Data Management (PDM)*-System gemanagt werden: Das zwingt den Softwarelieferanten dazu, auch alle sekundären Artefakte wie Handbücher etc. genauso wie das Prdodukt an sich zu managen.

Das *Produktknowledgemanagement* ist zwingend nötig: Das bedeuted, dass die Beziehungen des Produktes zu anderen sowohl in menschenlesbarer Form für den Vertrieb oder Support, als auch in maschinenlesbarer Form zum automatischen Entdecken und Auflösen von Konflikten vorhanden sein müssen. Die Verfügbarkeit von alten Versionen muss ebenso gewährleistet sein.

2.2.6.3 Release-Prozess als Prozess zwischen Entwicklungs- und Deployment-Prozess

Bei Ballintijn et. al. [BJv 04, Ball 05] erfolgen die Release- und Deployment-Prozesse beim Hersteller von Anwendungssoftware. Der Release-Prozess beginnt nach dem Softwareentwicklungsprozess und endet vor dem Deployment-Prozess. Auch sie gehen in ihrem Szenario von einer Vielzahl von Kunden aus, die mit einer Vielfalt von Varianten und Versionen versorgt werden müssen.

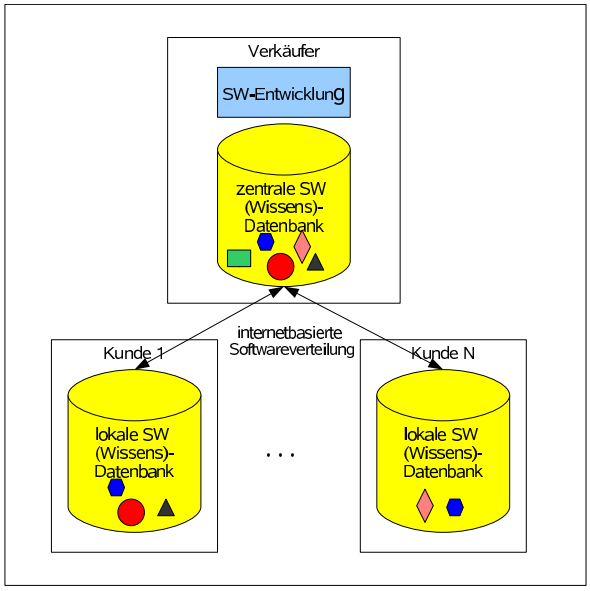


Abbildung 2.27: *Software Delivery Model* [Ball 05]

Zur Bewältigung der Komplexität des Release- und Deployment-Prozesses steht auch hier das Managen von Produktinformationen im Fokus. Die so genannte *Intelligent Software Knowledge Base* (ISKB) unterstützt den Release- und Deployment-Prozess. Sie ist eine Art von PDM System.

In ihr werden alle Informationen über alle Artefakte gespeichert, die Bestandteil einer Anwendung über ihren gesamten Lebenszyklus sind. Durch Erzwingen von Konsistenzanforderungen kann dann der Prozess zum Aktualisieren von Software beim Hersteller verbessert werden. Mithilfe der Informationen aus der ISKB können dann auch Auswirkungen von Softwareänderungen beim Kunden im Vorfeld analysiert werden.

Die Informationen in der ISKB werden durch so genannte*Feature Diagrams* dargestellt. Als weiterer Punkt steht hier die dynamische Auslieferung von Software über das Internet im Fokus. Diese wird durch die ISKB unterstützt, indem Softwareupdates durch Bildung der Differenz zu der beim Kunden vorhandenen Software erstellt werden (siehe Abbildung 2.27). Die ISKB ist deshalb aufgeteilt auf Datenbanken beim Hersteller und bei den Kunden.

2.2.6.4 Release-Management als Subprozess innerhalb der Wartung

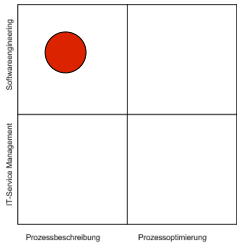


Abbildung 2.28: Einordnung Wartung

In diesem Abschnitt werden verschiedene Quellen zusammengefasst, die den Release-Prozess als Bestandteil des Wartungsmanagements einordnen. Auch hier gibt es eine grosse Bandbreite von Sichtweisen auf das RM. Aus diesem Grund ist keine einheitliche Einordnung in das Lebenszyklusmodell möglich.

Die Einordnung in das Klassifikationsschema ist hier jedoch dem SWE zugeordnet. Bei allen nachfolgenden dargestellten Modellen handelt es sich um beschreibende Prozesse (siehe Abbildung 2.28).

2.2.6.4.1 Release-Management bei Stark et. al. Bei Stark et. al. [SOSA 99] ist die Ausgangslage das Anforderungsmanagement eines Systemherstellers. Es ist zuständig für die Vereinbarungen über den Release-Inhalt mit seinen Kunden. Der Release-Inhalt wird spezifiziert aufgrund der technischen Anforderungen, Performanzkriterien und geforderten Funktionalitäten.

Nach der erstmaligen Auslieferung eines Release an den Kunden ergeben sich in der Wartungsumgebung neue Anforderungen im Sinne von Änderungen oder Fehlerkorrekturen durch *Change Requests*. Eine Anforderung ist hier gleichbedeutend mit einem genehmigten *Change Request*. Der Release-Prozess wird hier als der Prozess aufgefasst, der sich mit der Handhabung dieser Anforderungen beschäftigt.

So ist hier die Aufgabe der Release-Planung, die Anforderungen zu analysieren, zu entwerfen oder zu priorisieren. Die Release-Planung findet hier beim Hersteller statt. Er stimmt zusammen mit dem Kunden ab, wel-

Anforderungsart	Kategorie
Berechnungen	Falscher Operand in Gleichung
	Falsche Klammerung
	Falsche Gleichung
Eingabe	Falsches Format
	Dateiende fehlt
Schnittstelle	Software- / Hardwareschnittstelle
	Software- / Benutzerschnittstelle
	Software- / Datenbankschnittstelle
	Software- / Softwareschnittstelle
Performanz	Zeitlimit überschritten
	Speichergrenze überschritten

Tabelle 2.2: Auszug der Taxonomiy der Anforderungsarten aus [SOSA 99]

che Anforderungen umgesetzt werden. Durch die Volatilität der Änderungswünsche wird es für die Hersteller schwierig, verlässliche Release-Terminpläne und Budgetpläne zu erstellen.

Auf der Basis von empirischen Untersuchungen entwickeln Stark et. al. eine Methode, anhand derer die Release-Termine zuverlässiger bestimmt werden können. Dazu haben sie die Anforderungsarten klassifiziert (siehe Tabelle 2.2). Jeder Anforderungskategorie wurde der gemessene Zeitverbrauch zwischen dem Zeitpunkt der Genehmigung und dem Zeitpunkt der Abnahme des Releases beim Kunden gegenübergestellt. Des Weiteren wurden die Anforderungen unterteilt in neue, zu löschende und zu ändernde Anforderungen bezüglich des ursprünglich mit dem Kunden übereingekommenen Anforderungskataloges. Die Summe dieser Anforderungsänderungen, bezogen auf die Anzahl der ursprünglichen Anforderungen, wird als Anforderungsvolatilität (*Requirement Volatility*) bezeichnet.

Abbildung 2.29 zeigt, wie sich die Häufigkeitsverteilung von Anforderungen bezogen auf ihre Kategorie verteilt. Die Grafik zeigt, dass aufgrund von logischen Fehlern die meisten Änderungen resultieren. Mithilfe dieses Wissens werden z.B. bei Reviews von nachfolgenden Entwurfs- und Kodierungsvorgängen die logischen Gesichtspunkten detaillierter betrachtet.

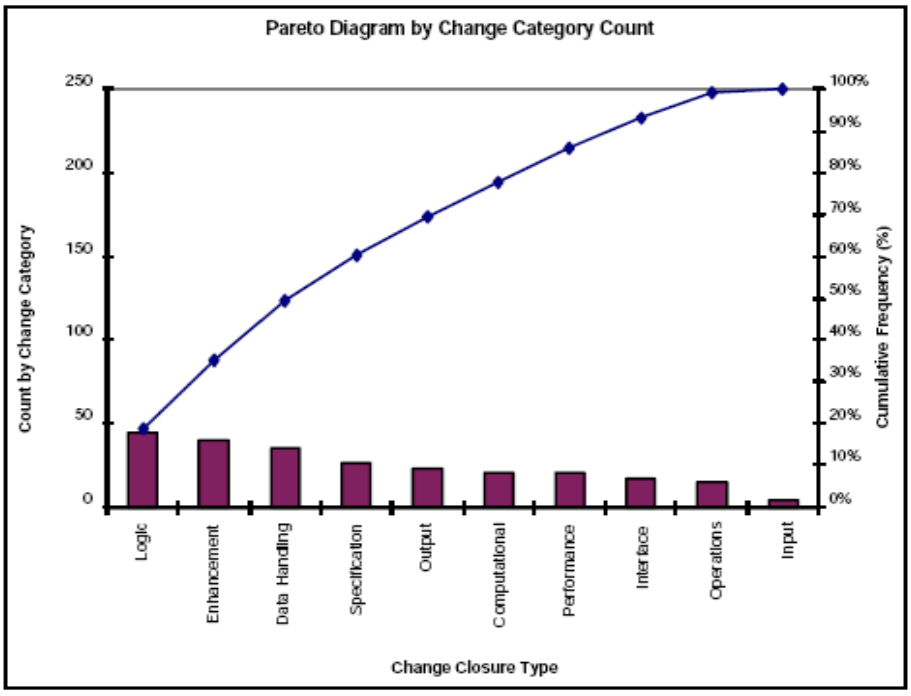


Abbildung 2.29: Häufigkeit der Anforderungen nach Kategorie (aus [SOSA 99])

Das Ergebnis ihrer Arbeit ist die nachfolgende Formel. Mit ihr kann die Auswirkung von risikobewerteten Änderungen auf die Zeitplanung berechnet werden. Das Risiko wird als Anforderung/Personentage gemessen.

$$PercentPlannedSchedule = 0.97 + 0.41 * RequirementVolatility^{1/2} + 0.23 * Risk$$

Ein neues Release ohne Anforderungsänderungen und damit ohne erwartetes Risiko wird laut dieser Gleichung zu 97% innerhalb der eingeplanten Zeit ausgeliefert.

Einordnung in das Lebenszyklusmodell Abbildung 2.30 zeigt, welche Bereiche das Modell von Stark et. al. im Lebenszyklusmodell abdeckt. Die Anforderungen für Änderungen ergeben sich nach der ersten Auslieferung eines Releases beim Kunden im Betrieb. Diese Anforderungen werden gesammelt und gemeinsam mit dem Kunden priorisiert. Aufgrund dieser Priorisierung wird ein neues Release zusammengestellt. Die Sichtweise von Stark ist dem SWE zuzuordnen. Er beschreibt nicht, welche Prozesse beim Kunden eingeführt sind, sondern wie ein Hersteller Änderungswünsche seiner Kunden einteilen und planen kann.

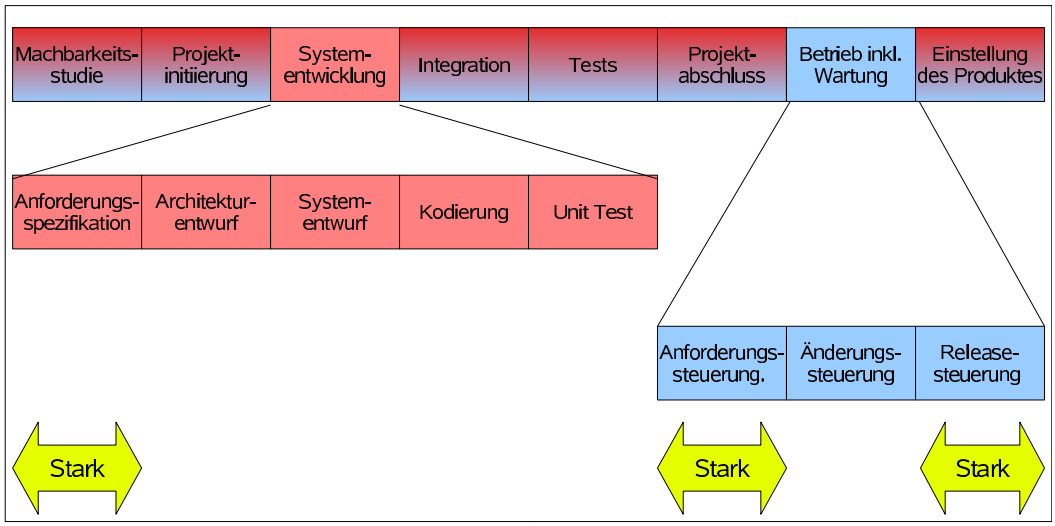


Abbildung 2.30: Einordnung des Modells von Stark et. al. in das Lebenszyklusmodell

2.2.6.4.2 Release-Management bei Brinkkemper et. al. Bei Brinkkemper et. al. [JBBvN 05, A Case Study in Mass Market ERP Software] wird sowohl der Release-, als auch der Deployment-Prozess als Subprozess des Wartungsprozesses eingeordnet. Hier wird die ISKB als integriertes *SCM/PDM/CRM* System aufgefasst. Sie speichert alle Informationen über alle Artefakte, die Bestandteil über den gesamten Applikationslebenszyklus sind. Auf Verkaufsseite speichert die ISKB alle Informationen über alle verfügbaren Anwendungen in allen je verfügbaren Versionen. Auf Kundenseite speichert sie Informationen über die installierten Anwendungen, Anwendungseinstellungen und Konfigurationen. Abbildung 2.31 zeigt den Verteilprozess der Releases. Dieser Prozess wird aufgrund der geforderten Qualitätskriterien gesteuert.

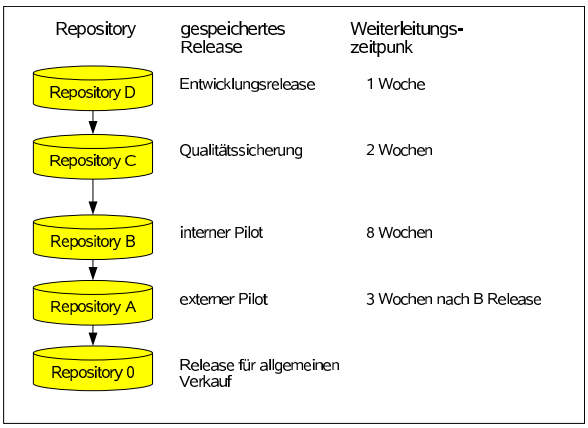


Abbildung 2.31: Repository- und Weiterleitungs-Schema aus [JBBvN 05]

Die Entwickler arbeiten mit dem D *Repository*. Nach Fertigstellung der Entwicklungsarbeiten und Überprüfung durch die Programmierer wird das Release in das C *Repository* weitergegeben. Alle vorherigen Daten werden dann durch die neuen überschrieben. Nachdem das Release im C *Repository* durch die Qualitätsicherung überprüft worden ist, wird es einmal in zwei Wochen in das B *Repository* kopiert. Dort soll das Release vom internen Personal benutzt und dadurch weiter getestet werden. Nach ca. zwei Monaten des internen Testens wird davon ausgegangen, dass das Release stabil genug ist. (Anmerkung: Es handelt sich hier um *ERP* Software, die auch selbst beim Hersteller eingesetzt wird.) Dann wird sie in das A *Repository* kopiert. Dort wird die Software für externe Pilotkunden zur Verfügung gestellt. Nach weiteren drei Wochen der Pilotphase wird das Release in das NULL *Repository* kopiert. Dort steht es für alle Kunden zur Verfügung.

2.2.6.5 Release-Management als Prozess zwischen Entwicklung und Installation

Ramakrishnan [Rama 04] definiert das Software Release-Management (SRM) als Prozess, durch den die Softwareentwickler die Software den Benutzern bereitstellen. Die Aktivitäten des Prozesses starten, wenn eine Komponente fertig entwickelt wurde und enden mit der Installation. Hier werden im SRM jedoch die Installation, Konfiguration oder Laufzeitkonfigurationen nicht berücksichtigt. SRM sieht der Autor als Bindeglied zwischen der Organisation, die die Komponenten erstellt hat und der Organisation, die sie zu einer einzigen Anwendung zusammenstellt. Ein wichtiges Kriterium im SRM Prozess sind auch hier das Managen der Abhängigkeiten, die zwischen verschiedenen Komponenten bestehen. Dieser Faktor wird noch verstärkt, wenn die Komponenten von verschiedenen Organisationen kommen.

2.2.6.6 Release-Prozess als Bestandteil des Deployment-Prozesses

Bei nachfolgenden Modellen ist der Release-Prozess ein Bestandteil des so genannten Deployment-Prozesses. Die beiden vorgestellten Quellen haben jeweils die SWE-Sichtweise. Die Aktivitäten des Deployment-Prozesses werden unterschiedlich beschrieben. Deshalb ist keine einheitliche Einordnung in das Lebenszyklusmodell möglich.

2.2.6.6.1 Release-Management bei Carzaniga et. al. Nach Carzaniga et. al. [Carz 97, vHC+ 98, CFH+ 98] ist die Release-Tätigkeit ein Bestandteil des so genannten *Software Deployment Process*.

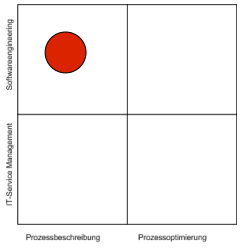


Abbildung 2.32: Einordnung RM bei Carzaniga et. al.

Darunter werden alle Aktivitäten verstanden, die ein Softwaresystem für die Benutzer verfügbar machen. Dieser Prozess beinhaltet neben der Release-Aktivität die Aktivitäten Installation, Aktivierung, Deaktivierung, Update und Entfernen von Komponenten (siehe Abbildung 2.33).

Die Release-Aktivität ist hier die Schnittstelle zwischen dem Entwicklungs- und dem Deployment-Prozess. Sie beinhaltet wiederum alle Tätigkeiten zur Bereitstellung eines Systems, damit es korrekt zusammengestellt und an den Kunden ausgeliefert werden kann. Dafür müssen alle für das Softwaresystem beim Kunden benötigten Ressourcen bestimmt werden. Ebenso müssen Informationen zu den nachfolgenden Tätigkeiten des Deployment-Prozesses gesammelt werden. Das Paketieren des Systems und die Bereitstellung von Release-Informationen für interessierte Parteien gehört ebenso noch zu den Aufgaben innerhalb dieses Release-Prozesses.

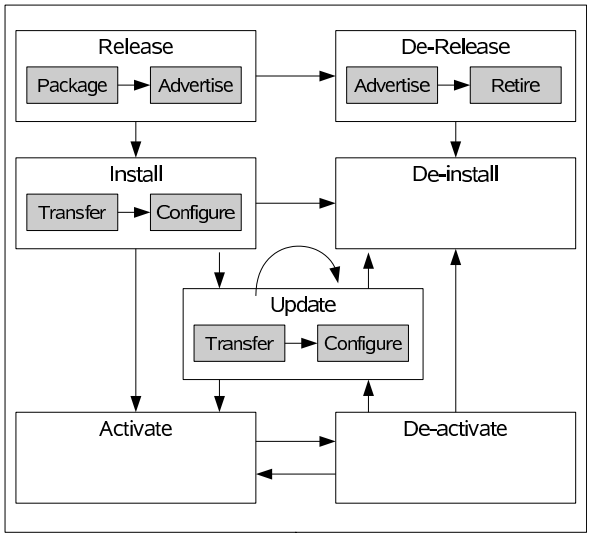


Abbildung 2.33: Deployment-Prozess aus [Carz 97, S. 3]

Auf der Grundlage dieser Definitionen legen sie ein dreiteiliges Modell fest, welches die Basis für ein Rahmenwerk zur Einordnung von Deployment-Prozessen bildet [CFH+ 98]. Das Modell besteht aus der Deployment-Prozessspezifikation und der Architektur, die auf der Abstraktion von Produkt, Umgebung und Regelwerk basiert. (siehe Abbildung 2.34)

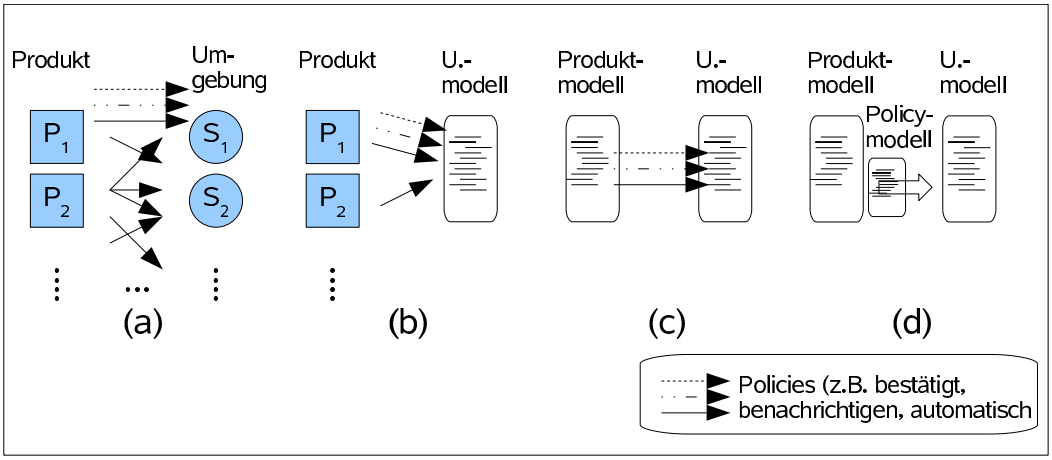
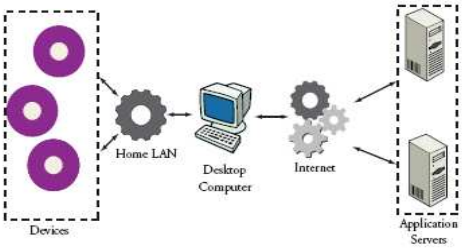


Abbildung 2.34: Framework für Deploymentprozesse nach [CFH+ 98]

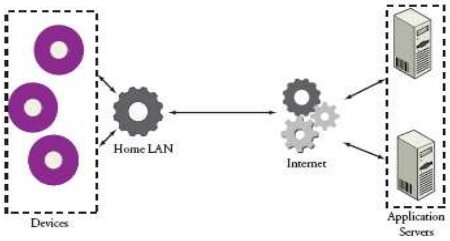
Die Abbildung zeigt eine Aktivität des Deployment-Prozesses, z.B. die Installation. Auf der linken Seite muss diese etwa für *m* Produkte auf *n* Umgebungen mit drei Regeln definiert werden. Bei den Schritten von a) nach d) in der Abbildung erfolgt eine Reduzierung bzw. Abstraktion der speziellen Prozeduren. Mit diesem Rahmenwerk beurteilen sie eine Auswahl von verfügbaren oder vorgeschlagenen Deployment-Techniken, wie etwa durch das Werkzeug *OpenView* der Firma HP oder dem *Red Hat Package Manager (RPM)* .

Release-Management bei Hall et. al. Hall et. al. [Hal99] entwerfen auf der Grundlage der Definition von Carzaniga et. al. das so genannte *Software Dock*. Das *Software Dock* ist ein Rahmenwerk für einen verteilten, agentenbasierten Deployment-Prozess. Es unterstüzt die Kooperation zwischen Softwareproduzenten und Softwarebenutzern und ist unterteilt in ein *Release Dock* beim Produzent und ein *Field Dock* beim Benutzer.

Die Agenten docken sich beim Field Dock an und können sich für Ereignisse des *Release Docks* registrieren. Dann können sie als Antwort auf diese Ereignisse Tätigkeiten ausführen, wie etwa die Installation einer Neue-



(a) Upgrade von Software nach Pull-Prinzip aus [Ande 00]



(b) Upgrade von Software nach Push-Prinzip aus [Ande 00]

rung. Mithilfe eines eingeführen standardisierten Deployment-Schemas (DSD Format) stehen den Agenten die Informationen bezüglich der Beschreibung des Softwaresystems zur Verfügung.

Release-Management bei Thanheiser et. al. Thanheiser et. al. [TTS 04] entwerfen auf der Grundlage der Definition des Deployment-Prozesses von Carzaniga die so genannte Softwaretankstelle. Sie erlaubt berechtigten Benutzern die Software auf der eigenen oder geliehenen Hardware zu installieren. Die Software liegt in einer Softwaredatenbank vor. Die Softwaretankstelle soll heterogene Geräte mit unterschiedlicher, evtl. sogar unbekannter Hard- und Softwarekonfiguration zuverlässig und sicher mit Software versorgen können.Ein Prototyp wurde hiervon bereits in Betrieb genommen.

Einordnung der Modelle von Carzaniga, Hall und Thanheiser et. al. Die Beschreibungen dieses Abschnitts sind speziell ausgerichtet auf die Schnittstelle zwischen Softwarehersteller und Softwarenutzer. Sie haben den Fokus auf Aktivitäten der technischen Umsetzung des Deployment-Prozesses. Innerhalb dieses Prozesses ist die Release-Tätigkeit gleichbedeutend mit der Verfügbarmachung einer neuen Softwareversion und der technischen Einführung der Software beim Benutzer. Das RM wird hier beim Softwarehersteller ausgeführt. Aus diesem Grund ist es der SWE-Sicht zuzuordnen. Dieser technische Aspekt des RM kann im Lebenszyklusmodell nicht abgebildet werden. Deshalb wird hier auf die Darstellung verzichtet.

2.2.6.6.2 Release-Management bei Andersson

Auch bei Andersson [Ande 00] sind die Tätigkeiten im Deployment-Prozess das Paketieren, Ausliefern, Installieren, Konfigurieren und Aktivieren von Software.

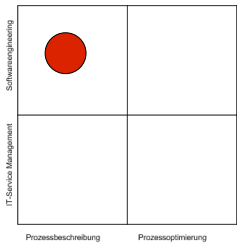


Abbildung 2.35: Einordnung RM bei Andersson

Der Fokus hier liegt speziell im Bereich des *pervasive computing*. Hier bestehen besondere Herausforderungen. Es kann z.B. nicht davon ausgegangen werden, dass es eine administrative Rolle gibt, die sich um den Prozess kümmert.

Im normalen Haushalt etwa wird kaum der Kühlschrankbesitzer für die nötigen Updates zur Verantwortung gezogen werden können. Vorgeschlagen werden hier Deployment-Prozesse sowohl als Pull- als auch als Push-Modell. (siehe Abbildung 2.36(a) und 2.36(b))

Einordnung des Modells von Andersson in das Lebenszyklusmodell Der Prozess von Andersson wird aus der Sichtweise des Herstellers beschrieben. Der Hersteller unterstützt die automatische Einführung und Deaktivierung von Software. Die eigentliche Betriebsphase wird in dieser Beschreibung nicht adressiert. Der Fokus liegt auch hier auf der technischen Umsetzung der Softwareeinführung. Diese kann im Lebenszyklusmodell nicht abgebildet werden.

2.2.6.7 Release-Management als Bestandteil des Konfigurationsmanagements

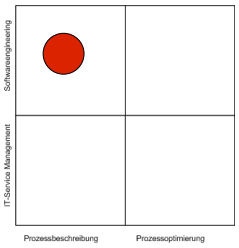


Abbildung 2.36: Einordnung RM im KM

Nicht nur im V-Modell (vgl. Abschnitt 2.2.1) wird das RM dem Konfigurationsmanagement (KM) zugeordnet, sondern es werden nachfolgend noch weitere Beispiele vorgestellt, die das RM als Bestandteil des KM beschreiben.

Die aufgezeigten Beispiele zeigen, dass auch hier keine einheitliche Sichtweise auf die Aufgaben des KM und damit auch keine einheitliche Sichtweise auf die Aufgaben des RM bestehen. Dadurch ist bei nachfolgenden Beispielen keine allgemeingültige Einordnung in das Lebenszyklusmodell möglich. Das einzige gemeinsame Merkmal nachfolgender Beschreibungen ist, dass es sich hierbei um beschreibende Prozesse handelt.

2.2.6.7.1 Release-Management bei Crnkovic et. al. Bei Crnkovic et. al. [CAPD 03, Crn99] ist das Release-Management ein Bestandteil des *Software Configuration Management* (SCM) . SCM ist eine gesteuerte Methode, um die Entwicklung und Änderung von Software während des gesamten Produktlebenszyklusses zu managen. Als Produktlebenszyklus wird hier ein generischer Produktlebenszyklus definiert, der im Prinzip dem hier eingeführten Lebenszyklusmodell entspricht. Dieser generische Produktlebenszyklus wird bei Crnkovic et. al. zur Gegenüberstellung der Prozessschritte in der Hardware- und Softwareentwicklung benutzt. Ziel ist eine Integration des SCM und des *Product Data Management* (PDM) .

Neben RM ist hier auch das Change-Management eine Aufgabe des SCM. RM ist in der Abbildung 2.37 die letzte Phase innerhalb der Softwareentwicklung vor der Phase Produktion. Das RM beinhaltet die Paketbildung der Software zu einem auslieferbaren Format inklusive der Erzeugung der Dokumentation über die Änderungen.

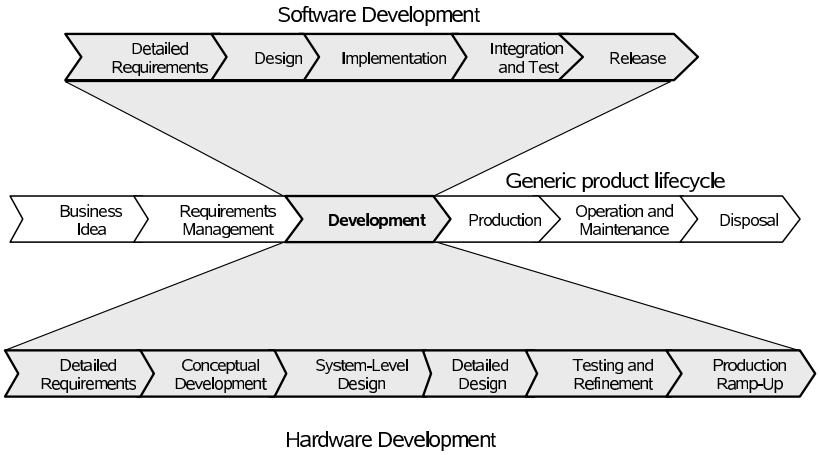


Abbildung 2.37: Phasen in Software- und Hardwareentwicklung, aus [CAPD 03, S. 7]

Einordnung in das Lebenszyklusmodell Bei Crnkovic et. al. werden die SCM Prozesse beim Softwarehersteller und das PDM beim Hardwarehersteller beschrieben. Als Vergleichsgrundlage dient ein eigenes generisches Lebenszyklusmodell. Dieses entspricht dem hier vorgeschlagenen Lebenszyklusmodell. Aus diesem Grund wird hier auf eine Darstellung der Einordnung verzichtet.

2.2.6.7.2 Release-Management bei Gillis et. al. Bei Gillis et. al. [GiFr 05] werden die Release-Aktivitäten ebenfalls dem Konfigurationsmanagement zugeordnet. Die Autoren bemerken, dass das Managen von Komponenten ab dem Zeitpunkt des Erstellens, Verteilens und Außerbetriebnehmens des Releases genauso wichtig ist, wie das Managen der Softwareentwicklung selbst. Zudem steigen die Herausforderungen an das Konfigurationsmanagement und damit auch an das Release-Management. Gründe dafür sind u.a. die

zunehmende Modularisierung von Software. Durch diese Aufteilung von Software in mehrere kleinere Komponenten steigt die Häufigkeit von Softwareverteilaufgaben. Die Zunahme von Sicherheitsbedrohungen durch Viren etc. führt zu häufigeren Verteilzyklen von Sicherheits-Patches.

Das Release beinhaltet kompilierte und zu einem Paket zusammengestellte Softwarekomponenten. Die Komponenten und ihre Abhängigkeiten werden diesem Release eindeutig zugeordnet. Dadurch soll die Reproduzierbarkeit der Applikation gewährleistet werden. Änderungen an den Komponenten dürfen dann im Anschluß nur durch das Erzeugen einer neuen Version durchgeführt werden.

Einordnung in das Lebenszyklusmodell Gillis et. al. sehen in ihrem Vorschlag das Konfigurationsmanagement für den gesamten Lebenszyklus vor. Ihr Lebenszyklus beinhaltet die Entwicklung von Softwarekomponenten, die Integration inkl. Integrationstests zu einem Release, die Verteilung dieses Releases und die Einstellung von Produkten. Der Lebenszyklus wird bei ihnen allerdings nicht detaillilerter beschrieben. Die Sichtweise hat hier den Fokus auf das SWE. Der Lebenszyklus endet mit dem Deployment von Software. Das Einstellen bezieht sich hier auf die Einstellung der Entwicklung dieser Software und nicht auf die Einstellung des Betriebes dieser Software. Abbildung 2.38 zeigt die Einordnung in das Lebenszyklusmodell.

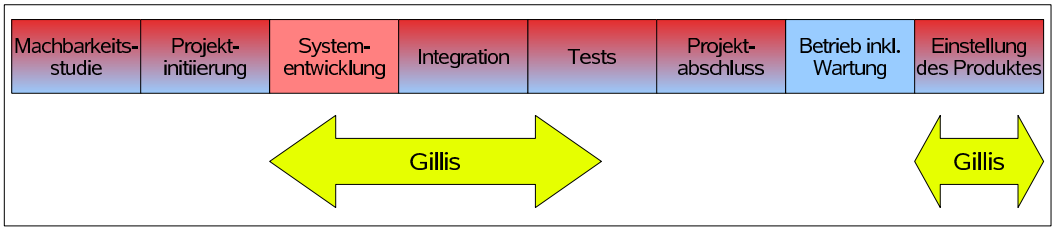


Abbildung 2.38: Einordnung in das Lebenszyklusmodell (Gillis et. al.)

2.2.6.8 Release-Management als Bestandteil der Abnahme- und Einführungsphase

In diesem Abschnitt werden Beispiele aufgezeigt, in denen das RM als Bestandteil der Abnahme- und/oder Einführungsphase aufgefasst wird. Die aufgezeigten Beispiele sind im SWE angegliedert und beziehen sich auf die Phasen Akzeptanztest, operativer Test und Betrieb.

2.2.6.8.1 Abnahme- und Einführungphase bei Balzert Balzert [Balz 96] benutzt nicht den Begriff Release.

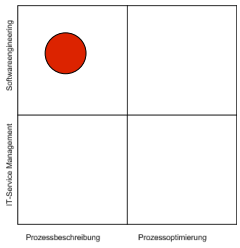


Abbildung 2.39: Einordnung Modell von Balzert

Er bezeichnet die Abnahme- & Einführungsphase als Phase zwischen der Realisierungsphase, wo das Softwareprodukt fertiggestellt wird und der Wartungs- & Pflegephase. In der Abnahmephase erfolgt die Übergabe des Gesamtproduktes an den Auftraggeber. Bei dieser Übergabe finden Abnahmetest und ggf. auch Belastungs- oder Stresstests statt. Nach erfolgreicher Abnahme erfolgt die Einführungsphase. In dieser Phase wird das Produkt in der Zielumgebung installiert, die Benutzer geschult und das Produkt in Betrieb genommen. Wichtig hier ist besonders die Zeitplanung der Umstellung. Je nach Umfang der geplanten Einführung kann die Inbetriebnahme entweder als direkte Umstellung, Parallelbetrieb oder als Versuchslauf erfolgen. Die verschiedenen Produktversionen werden in einem so genannten Wartungsarchiv aufbewahrt. In der anschließenden Wartungs- & Pflegephase erfolgen korrektive und progressive Tätigkeiten, wie Korrektur von Fehlern oder Anpassungen an neue Anforderungen. Auch Notfall-Reparaturen zur Sicherung des laufenden Betriebes werden dieser Phase zugeordnet. Das Konfigurations- und Änderungsmanagement unterstützt hier das Wartungsmanagement bei der geordneten Abwicklung von Wartungsaufgaben.

Einordnung in das Lebenszyklusmodell Abbildung 2.40 stellt dar, welchen Bereich die von Balzert vorgeschlagene Abnahme- und Einführungphase im Lebenszyklusmodell abdecken. Bei den Tests werden in der Abnahme- und Einführungsphase die Akzeptanz- und operative Tests adressiert. In den beiden Bänden von Balzert gibt es darüber hinaus noch eine Beschreibung des SWE-Prozesses. D. h. die eigentliche Lebenszyklusabdeckung ist umfassender, als es hier beschrieben wurde. Auf eine extra Ausführung wird an dieser Stelle verzichtet.

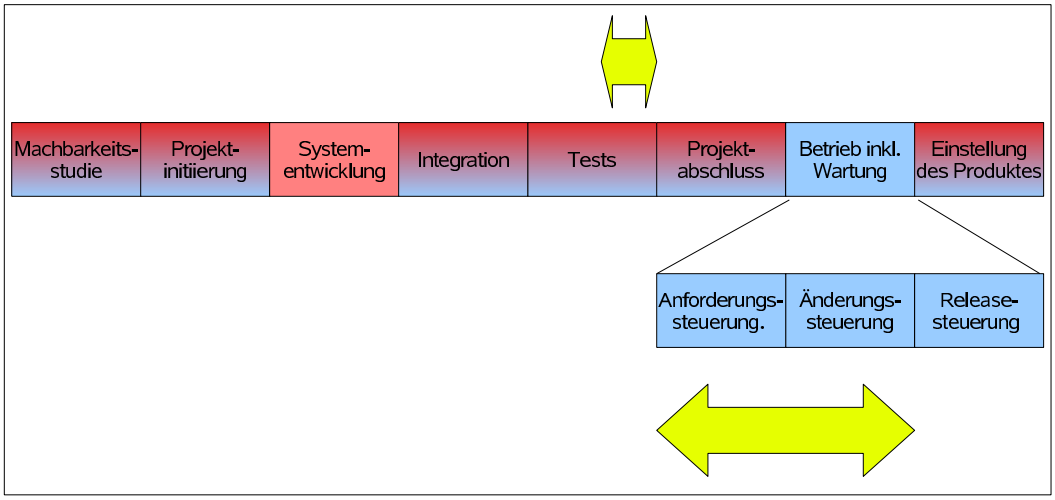


Abbildung 2.40: Einordnung der Abnahme- und Einführungphase bei Balzert in das Lebenszyklusmodell

2.2.6.8.2 Einführungsphase bei Steinweg Steinweg [Stei 04] setzt in seiner Beschreibung ebenfalls ein Phasenmodell voraus.

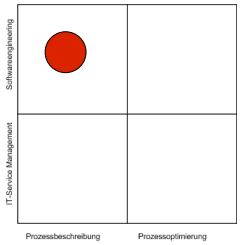


Abbildung 2.41: Einordnung Modells von Steinweg

Dieses besteht aus der Geschäftsanalyse, der Konzeption, dem Design, der Realisierung, der Einführung und dem Betrieb.

Der Release-Begriff wird von Steinweg ebenfalls nicht verwendet. Bei ihm wird bei der Einführung das System beim Kunden „implantiert“. Unterschieden wird hierbei zwischen standardisierter Software und individuell für den Kunden entwickelter Software. In der Einführungsphase muss der Kunde das System annehmen und die nötige Infrastruktur bereitstellen. Durch lokales *Customizing* wird das System an Kundenanforderungen angepasst. Möglicherweise entstehen hier wiederum Änderungsanforderungen. Bei der Anpassung des Systems werden vier Dimensionen tangiert. Die Geschäftsstrategie, die Personalstrategie, die IT-Strategie und die Organisationsstrategie. Bei der Geschäftsstrategie erfolgen Geschäftsanpassungen und Konvertierungen. Oft gehört dazu eine Übernahme von Altdaten in ein neues System. Bei Organisationsanpassungen werden Änderungen an Prozessen, Rollen oder der Aufbauorganisation durchgeführt. Das Personal wird für die Umstellungen entsprechend geschult. Schließlich muss noch das IT-System an die geänderten technischen Gegebenheiten angepasst werden.

Bei der Einführung stehen dem Einführungsteam alle Dokumente der vorherigen Phasen zur Verfügung. Vorgeschlagen wird hierzu eine gemeinsame Versionsverwaltung, die nach der Einführung nahtlos weitergepflegt werden kann. Der Ablauf der Einführung beinhaltet die Werkabnahme & Übernahme des Systems, den Pilotbetrieb, die offizielle Abnahme des Systems, den Roll-out, die Schulung und das *Going Live*. Bei der Werkabnahme & Übernahme wird das System installiert und in einer Testumgebung in Betrieb genommen. Anhand von Abnahmekriterien erfolgt ein vorläufiger Abnahmetest. Die Inhalte für den Abnahmetest werden bereits in der Konzeptionsphase erstellt. Durch den anschließenden Pilotbetrieb soll die Stabilität und Performanz des Systems getestet werden. Zusammen mit dem Kunden wird das Roll-out-Verfahren festgelegt. Das Roll-out-Verfahren beinhaltet die Beschreibung der notwendigen Systemeinstellungen, des Verteilungsverfahrens, der Termine und der zuständigen Personen. Bei der offiziellen Abnahme wird anhand der Abnahmekriterien für

oder gegen eine Übernahme des Systems entschieden. Die Abnahmekriterien sind im Wesentlichen die gleichen wie bei der Werkabnahme. Der Fokus liegt hier auf dem offiziellen Charakter der Abnahme. Durch diese erhält der Hersteller die Abnahmebescheinigung. Durch den Roll-out wird die produktive Inbetriebnahme vorbereitet. In dieser Phase erfolgt die Softwareverteilung durch möglichst automatische Verteilungsvorgänge und die Installation. Parallel dazu kann schon die Schulung durchgeführt werden. Hierfür muss eine Schulungsumgebung mit den notwendigen Daten, Berechtigungen und Unterlagen bereitgestellt werden. Durch den *Going Live* wird das System im Betrieb produktiv geschaltet.

Anschließend erfolgt die Phase Betrieb inklusive Wartung. Steinweg sieht für diesen Abschnitt die Prozesse von ITIL Service Support jedoch ohne das RM und ITIL Service Delivery vor (vgl. ITIL Abschnitt 2.3.4).

Einordnung in das Lebenszyklusmodell Das Lebenszyklusmodell von Steinweg deckt das vorgeschlagene Lebenszyklusmodell voll ab (siehe Abbildung 2.42). Steinweg schlägt selbst Vorgehensweisen für alle SWE-Phasen vor. Er definiert den Abschnitt Einführung, bei der ein neues System in den Betrieb implantiert wird. Für die eigentliche Betriebsphase hat er selbst keine Vorgehensweise entwickelt, sondern sieht den Einsatz der ITIL Prozesse vor.

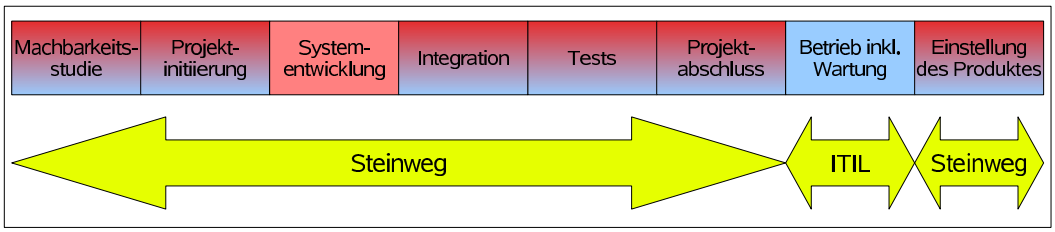


Abbildung 2.42: Einordnung des Modells von Steinweg in das Lebenszyklusmodell

2.2.7 Software Engineering Body of Knowledge

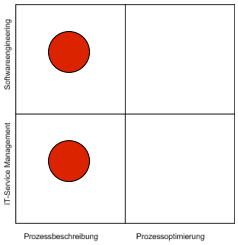


Abbildung 2.43: Einordnung SWE-BOK

Hilburn et. al. [SWE-BOK] haben im *Software Engineering Body of Knowledge* (SWE-BOK) eine Wissensbasis für das SWE zusammengestellt. Damit soll ein klarer Konsens über die Aufgaben innerhalb des Softwareengineerings hergestellt werden. Dazu organisieren und katalogisieren sie den *Body of Knowledge*. Dessen Ziel ist eine systematische, einheitliche, knappe und vollständige Beschreibung der SWE-Disziplin.

Die Architektur von SWE-BOK ist in die drei hierarchischen Ebenen Wissenskategorien, Wissensgebiete und Wissenseinheiten aufgeteilt. Wissenskategorien sind IT Grundlagen, SWE, Softwaremanagement und Softwaregebiete. Diese werden jeweils weiter unterteilt in die entsprechenden Wissensgebiete. Bei SWE sind Anforderungserwerb, Design, Kodierung, Testen und Betrieb und Wartung die Wissensgebiete. Diese sind jeweils weiter unterteilt in die Wissenseinheiten (siehe z.B. Abbildung 2.44).

2.2.7.1 Release-Management beim Software Engineering Body of Knowledge

Der Release-Begriff wird in der vorliegenden Beschreibung nicht explizit verwendet. SWE-BOK deckt alle Bereiche des klassischen SWE ab. Darüber hinaus beinhaltet das Wissensgebiet Betrieb und Wartung Konzepte, Methoden und Prozesse, damit ein Softwaresystem sich ändern und weiterentwickeln kann. Dieses Gebiet kann auch bei einer anderen Organisation als der Entwicklerorganisation eingesetzt werden. In SWE-BOK finden sich daher Merkmale der vorher beschriebenen Abnahme- und Einführungsphase und des Ablaufes, wie er nachfolgend bei z.B. ITIL (vgl. Abschnitt 2.3.4) noch dargestellt werden wird. Die Phase Wartung und Betrieb beinhaltet bei SWE-BOK folgende Wissenseinheiten [SWE-BOK, Pigo]:

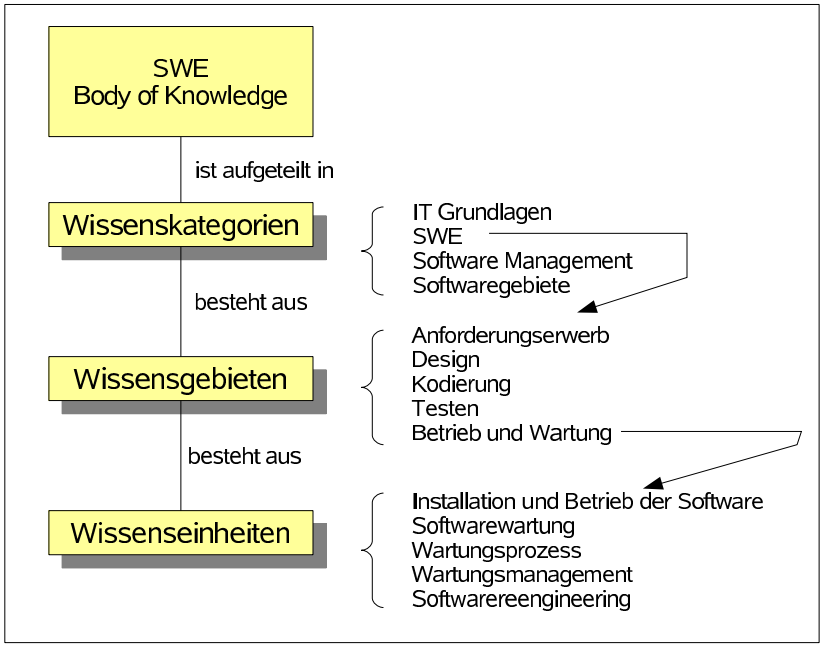


Abbildung 2.44: Ebenen von SWE-BOK

2.2.7.1.1 Installation und Betrieb von Software In dieser Einheit werden Methoden und Techniken für die Installation und den kontinuierlichen Betrieb der Software festgelegt. Weiterhin wird für einen reibungslosen Übergang der Systeme von der Entwicklungsorganisation zur Benutzerorganisation gesorgt. Das beinhaltet z.B. nötige Schulungen und Handbücher. Dieser Ansatz entspricht der Sichtweise einer Abnahme- und Einführungsphase von Balzert und Steinweg im Abschnitt 2.2.6.8.

2.2.7.1.2 Wartung In der Wartungseinheit wird zwischen korrigierender, adaptiver, perfektionierender und vorbeugender Wartung unterschieden. Der Wartungsprozess beinhaltet die Beschreibung der nötigen Wartungsphasen. Der Wartungsprozess startet mit einem *Change Request* und einer vorherigen Problemanalyse. Der C*hange Request* kann mit dem *Request for Change* in ITIL verglichen werden (siehe Abschnitt 2.3.4.1). Anschließend erfolgt eine Bewertung der möglichen Lösungswege. Hierfür fordert Pigoski [Pigo], dass das Wartungspersonal Wissen über die Codestruktur und den Inhalt hat. Die gewählte Lösung wird realisiert und getestet und schließlich dem Kunden zur Verfügung gestellt. Das Konfigurationsmanagement unterstützt die Steuerung und Kontrolle der Wartungsaktivitäten. Weitere unterstützende Aktivitäten innerhalb des Wartungsprozesses sind Verfifizieren, Validieren, Qualitätsicherung, Betrieb eines Helpdesks und Schulung von Benutzern. Die Einheit Wartungsmanagement ist zuständig für die organisatorischen und wirtschaftlichen Angelegenheiten innerhalb des Wartungsprozesses. Durch das Softwarereengineering werden Softwaresystemen restrukturiert und rekonstruiert. Damit soll die Qualität, Verständlichkeit und Wartbarkeit verbessert werden.

2.2.7.2 Einordnung des Software Engineering Body of Knowledge in das Lebenszyklusmodell

Bis auf die Machbarkeitsstudie, die von SWE-BOK nicht explizit adressiert wird, deckt es alle Phasen des Lebenszyklusmodells ab. Abbildung 2.45 zeigt die Einordnung in das Lebenszyklusmodell.

2.3 Release-Management im IT-Servicemanagement

In diesem Abschnitt wird RM aus der Sicht von ITSM beschrieben. IT-Services sind geschäftsprozessunterstützende IT-Funktionen. Sie stellen sich dem Benutzer als geschlossene, anwendungsorientierte Einheiten dar

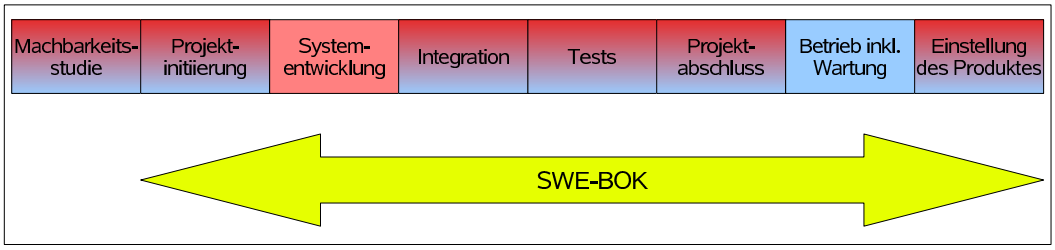


Abbildung 2.45: Einordnung von SWE-BOK in das Lebenszyklusmodell

[KHP 04]. Beim SWE lag der Fokus auf der Entwicklung von Systemen und die Sichtweise des RM lag daher in der Bereitstellung eines neuen Produktes. Beim ITSM hingegen liegt das Augenmerk auf dem operativen Betrieb einer IT-Infrastruktur.. Im nachfolgenden Abschnitt werden bekannte ITSM-Modelle (siehe ITIL Abschnitt 2.3.4 oder MOF Abschnitt 2.3.1) und auch weniger verbreitete Ansätze aus dem Forschungsumfeld vorgestellt.

2.3.1 Microsoft Operations Framework

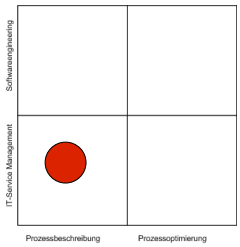


Abbildung 2.46: Einordnung von MOF

Das *Microsoft Operations Framework* (MOF) [Micr , MSF, Somm 04, PuQu 03] ist ein Referenzmodell für den IT-Service Betrieb der Firma Microsoft. Es basiert auf den Best-Practices von ITIL (vgl. Abschnitt 2.3.4) und erweitert es um Anleitungen für den Betrieb von IT-Umgebungen. Hierfür sind eigene Prozesse z.B. zum Speichermanagement oder zur Verzeichnisdienstadministration [Somm 04] definiert. MOF bietet Hilfestellung, um zuverlässige, verfügbare, wartbare und handhabbare Systeme auf der Basis von Microsoft Produkten und Technologien zu erhalten. Unterteilt ist das MOF-Modell in das MOF-Prozessmodell, das Teammodell und das Risikomodell.

Das Prozessmodell stellt den Lebenszyklus von IT-Dienstleistungen dar und enthält dazu die vier Quadranten *Changing*, *Operating*, *Support* und *Optimizing* [PuQu 03] (siehe Abbildung 2.47). Mithilfe der im Teammodell definierten Richtlinien können Mitarbeiter und Verantwortungen innerhalb einer Organisation strukturiert werden. Durch die Anwendung des Risikomodells können bestehende Risiken gemanagt werden. Im Gegensatz zu MSF (siehe Abschnitt 2.2.3), dass sich auf die Lieferung von Softwarelösungen im Rahmen von Projekten konzentriert, hat MOF den Fokus auf das IT-Servicemanagement.

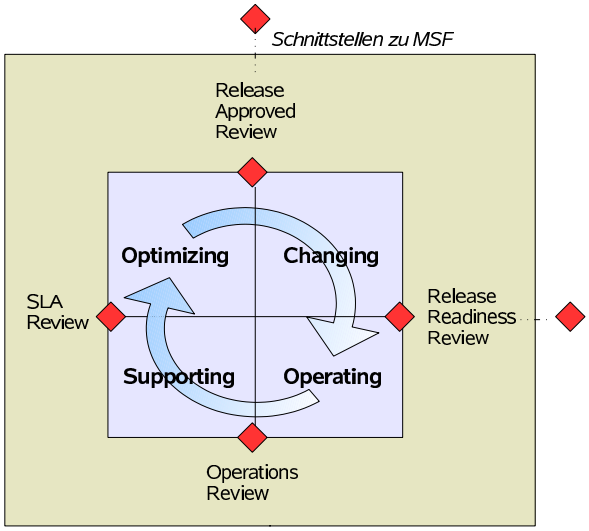


Abbildung 2.47: MOF Quadranten und Reviews

2.3.1.1 Release-Management beim Microsoft Operations Framework

Bei MOF ist das RM eine *Service Management Function* (SMF) innerhalb des MOF *Changing Quadrant*. SMF beschreiben Prozesse und Richtlinien zur Erbringung von IT-Dienstleistungen. Der *Changing Quadrant* beschreibt Prozesse, Verantwortlichkeiten, Reviews und Anleitungen, die Organisationen bei Änderungen ihrer IT Infrastruktur unterstützen.

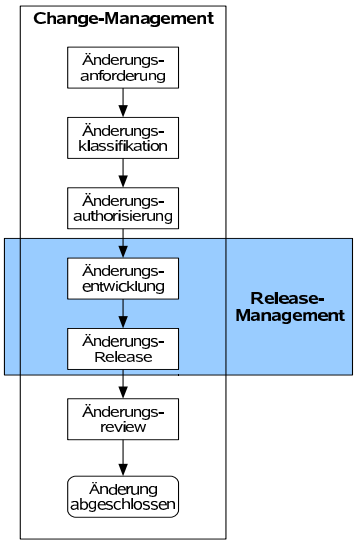


Abbildung 2.48: Zusammenhang zwischen Change- und Release-Management bei MOF nach [Mof05]

Neben dem RM ist auch das Change-Management und Konfigurationsmanagement eine SMF des *Changing Quadrant*. Abbildung 2.48 zeigt den Zusammenhang zwischen dem Change- und dem Release-Mangament bei MOF.

Das RM stellt den erfolgreichen Einsatz von Änderungen in der Produktion sicher. Es koordiniert und managed alle Releases für die IT Umgebung. Konzepte hierfür sind die Release-Strategie, der Roll-out, der *Release Approved Review* und der *Release Readiness Review* (RRR) . Eine Release ist hierbei jede Änderung oder Bündelung von Änderungen zum Einbringen in eine gemanagte IT-Umgebung.

Abbildung 2.49 zeigt die Prozessschritte beim Release-Management. Schnittstellen zum MSF bestehen hier durch den *Release Approved Review* und dem RRR. Der *Release Approved Review* ist der Auslöser für die Softwareentwickler, die Entwicklung der Änderung zu beginnen. Der RRR entspricht dem Meilenstein freigegebens fertiges Release beim MSF. Erst durch diese Genehmigung darf eine Softwaränderung in die Produktionsumgebung eingeführt werden.

Der Release-Manager bestimmt bei der Release-Planung, welche Änderungen nötig sind, um das Release in die Produktivumgebung einzuführen. Aufgrund dieser Änderungen wird der Release-Plan erstellt. Der Release-Plan listet die nötigen Aufgaben und Aktivitäten auf.

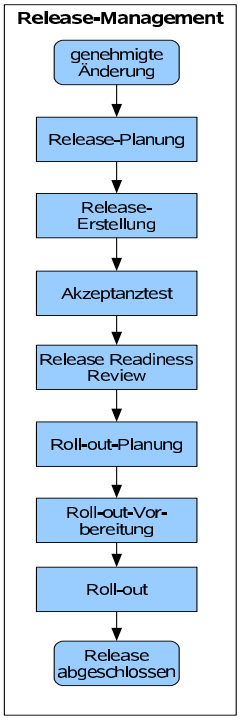


Abbildung 2.49: Release-Prozess bei MOF nach [Mof05]

Bei der Release-Zusammenstellung legt der Release-Manager die notwendigen Prozesse, Tools und Technologien für die Einführung in die Produktivumgebung fest. Akzeptanztests sollen das Auftreten negativer Auswirkungen durch das geplante Release verhindern. Möglicherweise entspricht die Testumgebung nicht ganz der Produktivumgebung. Aus diesen Grund kann es im Einzelfall nötig sein, bestimmte Tests in der Produktivumgebung durchzuführen.

Der RRR ist der letzte Meilenstein vor Beginn der eigentlichen Roll-out-Planung- und Vorbereitung. Hier wird für oder gegen den Release-Roll-out entschieden. Betrachtet wird u.a. die Funktionsfähigkeit des Release an sich und der Stand der Produktivumgebung. Die eigentliche Roll-out-Strategie wird bei der Roll-out-Planung festgelegt. Die Roll-out-Reihenfolge wird durch den Release-Manager begutachtet. Damit wird die Konformität mit den Geschäftsanforderungen und Prioritäten sichergestellt.

Vor dem eigentlichen Roll-out werden noch Vorbereitungen in der Produktiv- oder evtl. Pilotumgebung getroffen. Diese Vorbereitungen beinhalten die Information von Benutzern oder anderen Personal, die Schulung des *Service Desk* und technischer Mitarbeiter und die Sicherung kritischer IT Komponenten.

Beim Roll-out wird schließlich das Release in die Produktivumgebung gebracht. Der Release-Manager wird über den jeweiligen Status informiert. Die Änderungen werden dann noch in der *Configuration Management Database* (CMDB) nachgeführt. Beim Scheitern des Releases oder bei schwerwiegenden Problemen wird u. U. das Problem-Management zur Ursachensuche hinzugezogen. Falls der Fehler behoben werden kann, wird das dokumentiert und ein *Request for Change* (RfC) daraus erstellt. Ansonsten wird ein Roll-back durchgeführt.

2.3.1.2 Einordnung von MOF in das Lebenszyklusmodell

Abbildung 2.50 zeigt die Abdeckung von MOF und MSF mit dem Lebenszyklusmodell. MOF ergänzt die bereits vorhandene Abdeckung des Lebenszyklusmodells von MSF um den Bereich Betrieb und Wartung. Das Release-Management bildet hierbei den Koordinationspunkt zwischen dem Entwicklerteam und dem Betriebsteam [Somm 04].

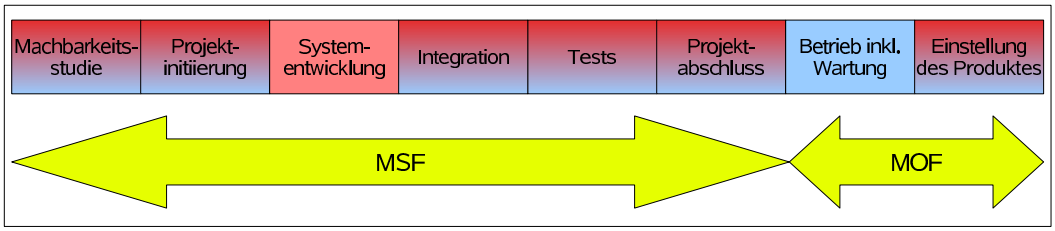


Abbildung 2.50: Einordnung von MOF und MSF in das Lebenszyklusmodell

2.3.2 Solution Management and Request Tracking

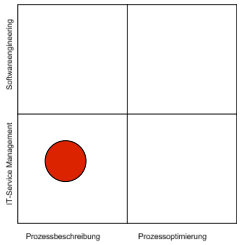


Abbildung 2.51: Einordnung SMART

Solution Management and Request Tracking (SMART) [McNi 04] ist eine Sammlung von in der Praxis bewährten Lösungen zum Anwendungssupport und zur Wartung (*Application Support and Maintenance* (ASM)).

ASM besteht aus den Geschäftspraktiken *Transition Management*, *Solution Management*, *Support Management*, *Change Control Management* und *Application Enhancement Management*.

2.3.2.1 Release-Management bei Solution Management and Request Tracking

Bei SMART wird ein so genanntes *Application Enhancement* als Release bezeichnet. Innerhalb der Managementdisziplin *Application Enhancemant* wird das Release entworfen, zusammengestellt, konfiguriert und getestet. Die Roll-out-Planung wird vom *Change Control Management* durchgeführt. Das *Application Enhancement Management* ist wiederum zuständig für die Kommunikation, Vorbereitung und Schulung.

Abbildung 2.52 zeigt die Einordnung in das Lebenszyklusmodell. SMART deckt hierbei die komplette Betriebsphase ab.

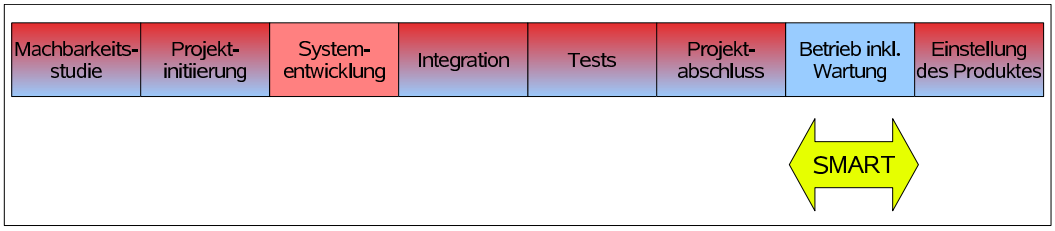


Abbildung 2.52: Einordnung von SMART in das Lebenszyklusmodell

2.3.3 SERVIAM Maintenance Framework

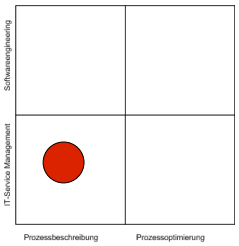


Abbildung 2.53: Einordnung SERVIAM

SERVIAM [KMTe 05, KM 04] ist ein Projekt der Swedish Computer Society in Verbindung mit der Stockholm University and Royal Institute of Technology, University of Skövde und veschiedenen Industriepartnern. SERVIAM behandelt die Entwicklung und Wartung servicebasierter Systeme.

Das SERVIAM Maintenance Framework [KMTe 05] ist ein Rahmenwerk für die Evolution und Wartung speziell von *Webservices*. *Webservices* bringen durch lose Kopplung verschiedener Komponenten diverser Hersteller oder durch schnellere Änderungszyklen eine erhöhte Komplexität von Wartungs- und Evolutionsprozessen mit sich. Durch den Einsatz von Komponenten verschiedener Hersteller reichen die Wartungs- und Evolutionsprozesse bis zu diesen Herstellern und sind deshalb nicht mehr im eigentlichen Fokus des eigenen Unternehmens. Aus diesem Grund unterziehen Kajko-Mattsson et. al. die Prozesse Change-Management, Problemmanagement, *Emergency Management* und RM einer Revision. [1]

Beim Change-Mangagement z.B. wird bei einem Änderungsbedarf des *Webservices* der nötige Service zugekauft, anstatt selbst entwickelt. Das hat speziell Auswirkungen auf die Anforderungsspezifikation. Das Change-Management bekommt dadurch einen globaleren Charakter, weil es Subprozesse bei den jeweiligen Unternehmen koordiniert. Das Problemmanagement hat evtl. beim Einsatz verteilter Anwendungen keine Übersicht mehr über die Systemstruktur. Wenn überhaupt kann der Zuständige nur noch die betroffenen Komponenten identifizieren. Die Herstellerorganisation muss dann weiteren Support leisten. Auch beim Emergency-Prozess müssen die Zuständigkeiten geklärt werden. Der Release-Prozess bei *Webservices* varriiert in der Größe der betroffenen Systeme, von einzelnen Komponenten bis hin zum kompletten *Webservice*. Des Weiteren resultiert aus einem schnellebigen Geschäftsumfeld eine hohe Anzahl möglicherweise sogar paralleler Releases.

Um diese Komplexität handhaben zu können, muss das Unternehmen einen ende-zu-ende Release-Prozess mit Beteiligung aller Betroffenen etablieren. Dadurch fällt ein erhöhter Kommunikations- und Koordinationsaufwand an. In manchen Fällen kann das sogar dazu führen, dass ein globales *Release Advisory Board* eingerichtet werden muss. Das RM kann eine solche Dynamik des Geschäftsumfeldes dann nur durch einen weniger starren Prozess abwickeln. Das RM wird detailierter im nachfolgenden Teilprojekt von SERVIAM beschrieben.

2.3.3.1 Evolution and Maintenance Maturity Model

Das *Evolution and Maintenance Maturity Model* (EM^3) [KMMe 05] ist ein Teilprojekt von SERVIAM. Das Aufgabengebiet des Projektes ist die Evolution und Wartung von *Webservices*.

2.3.3.1.1 Release-Management beim Evolution and Maintenance Maturity Model Das RM ist ein Bestandteil von EM^3. Es basiert auf der SMF von Microsoft und dem Release-Prozess, der bei Oracle im Einsatz ist. Bei EM^3 wird unterschieden zwischen der Herstellerseite und der Kundenseite. Der Hersteller ist für die Erstellung eines Releases verantwortlich und der Kunde ist verantwortlich für dessen Akzeptanz.

Ein Relase ist hier ebenfalls eine Menge von genehmigten Änderungen, die gemeinsam in die Produktivumgebung eingeführt werden (vgl. ITIL Abschnitt 2.3.4).

Als Release-Zyklus wird die Zeit ab der Release-Planung bis zur Auslieferung definiert. Unterschieden wird hier noch zwischen einem strategischen langfristigen und einem individuellen kurzfristigen RM. Im ersteren werden eine Anzahl von Release-Zyklen, ihre Inhalte, Verantwortlichen und Zeiten geplant. Betrachtet werden hier vor allem strategische Unternehmensziele. Diese strategischen Release-Pläne werden durch individuelle Releases realisiert. Bei diesen wird entschieden, welche Anforderungen durch das Release umgesetzt werden. Des Weiteren wird unterteilt in funktionalitätsgetriebene und zeitplangetriebende Releases. Bei den

[1] Aus den vorliegend zitierten Quellen ist nicht ersichtlich, ob diese Prozesse auf einem bestehenden Referenzmodell (z.B. ITIL) beruhen oder selbst definiert wurden.

funktionalitätsgetriebenen ist die Umsetzung einer definierten Änderung ausschlaggebend. Bei den zeitplangetriebenden ergibt sich die *Deadline* der Einführung durch vorab festgelegte Termine. An diesen Terminen wird dann alles eingeführt, was bis zu diesem Zeitpunkt fertiggestellt ist. Die Änderung an sich wird noch in eine fehlerkorrigierende oder funktionalitätserweiternde Korrektur oder ein Mischtyp von beiden eingeteilt.

Bei EM^3 wird ein Rollen- und ein Prozessmodell für das RM eingeführt. Im Rollenmodell werden 15 verschiedene Rollen auf Hersteller- und Kundenseite identifiziert (siehe Abbildung 2.54)

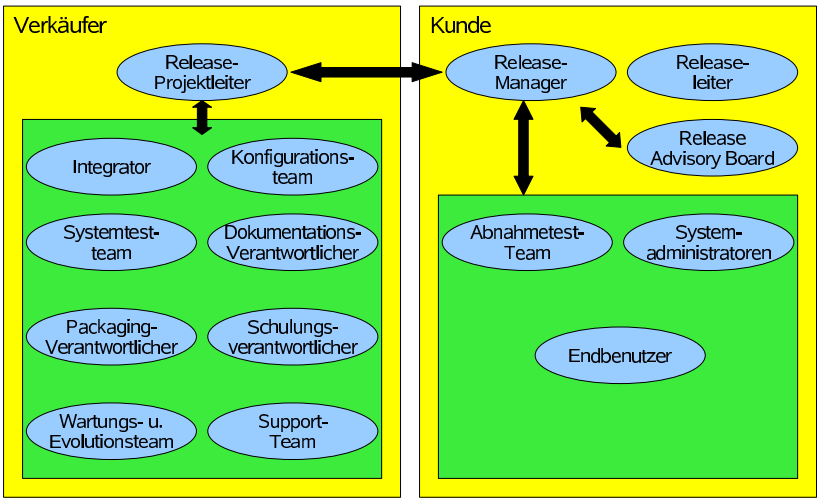


Abbildung 2.54: Rollenmodell von EM^3

Der Prozess besteht aus sieben Prozessphasen. Diese sind verteilt auf die Hersteller- und Kundenseite. Der Auslöser des Release-Prozesses ist ein genehmigter *Change Request*. Der Change-Prozess wird von EM^3 nicht beschrieben. Die Phasen des Release-Prozesses sind in ihrer zeitlichen Abfolge in Abbildung 2.55 dargestellt. Diese werden nachfolgend detaillierter beschrieben.

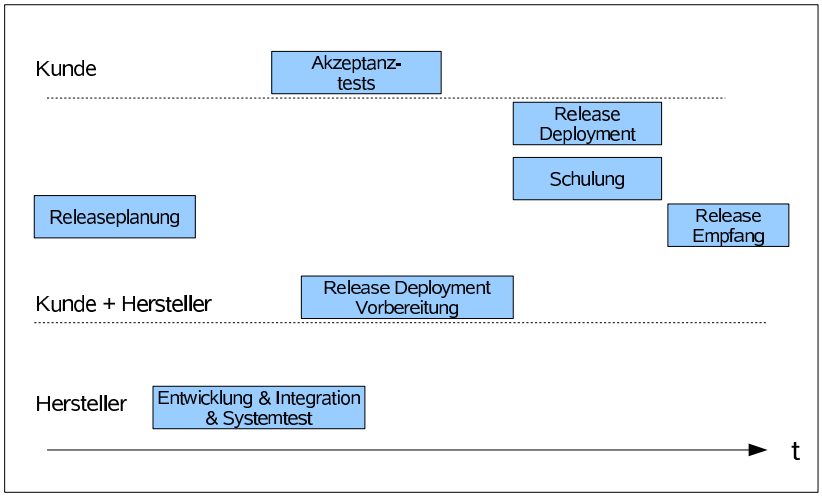


Abbildung 2.55: zeitlicher Prozessablauf des RM bei EM^3

Release Planung: Durch die Release-Planung wird der Umfang und Inhalt des Release festgelegt. Auf der Herstellerseite wird vom Release-Projektleiter die *Baseline* entworfen. Er analysiert den *Change Request* und schätzt daraufhin den Ressourcenbedarf. Auf dieser Grundlage wird der Release-Inhalt und Zeitplan mit dem Release-Manager der Kundenseite abgestimmt. Aus dieser Abstimmung resultiert ein vorläufiger Release-Plan. Dieser muss vom *Release Advisory Board* (RAB) genehmigt werden. Dieses

Gremium vergibt auch eine eindeutige Release-Bezeichnung. Der Release-Manager und der Release-Projektleiter stellen dann jeder für sich ein eigenes Release-Team zusammen.

Entwicklung & Build-Integration & Systemtest: In dieser Phase wird beim Hersteller der Change-Request realisiert, integriert und getestet.

Benuzterakzeptanztest: Das Abnahmetestteam erstellt den Abnahmetest und das Konfigurationsteam stellt die Abnahmeumgebung zur Verfügung. Das Abnahmetestteam überprüft, ob die Infrastruktur der Abnahmeumgebung für die Tests geeignet ist. Die Abnahme wird vom Abnahmetestteam zusammen mit dem Supportteam und Endbenutzern durchgefürt. Der Release-Manager und das Abnahemtestteam evaluieren anschließend die Testergebnisse und entscheiden aufgrund der Abnahmekriterien das weitere Vorgehen. Der Release-Projektleiter wird dann vom Release-Manager über die Annahme informiert. Der Release-Projektleiter initiiert daraufhin die Deployment-Vorbereitungsphase. Die Phasen Entwicklung & *Build-Integration* & Systemtest und Benuzterakzeptanztest sind hier nicht als eigentliche RM Phasen eingeordnet. Sie werden jedoch als Vorbedingung für eine Durchgängigkeit des Release-Prozesses mit betrachtet.

Deployment-Vorbereitung: Der Hersteller trifft in dieser Phase die Vorbereitungen für die Installation des Systems beim Kunden. Der Release-Projektleiter und das Wartungsteam entwickeln einen Back-out-Plan, falls der Roll-out scheitert. Dieser wird vom Systemtestteam getestet. Nach dem erfolgreichen Abnahmetest erstellt der Dokumentationsverantwortliche die Release-Notes und die Benutzerhandbücher. Wenn all diese Dokumente für die Release-Baseline fertiggestellt sind, erzeugt der Packaging-Verantwortliche das Release-Package. Dieses wird dem Release-Manager übergeben. Die Systemadministratoren müssen die Eignung der Umgebung für das neue Release bestätigen. Wenn das Release für die Installation bereit ist, fordert der Release-Manager vom RAB die Freigabe.

Release-Verteilung: In dieser Phase wird das Release von den Systemadministratoren installiert. Wenn hier Probleme auftreten, kommt ggf. der Back-out-Plan zum Einsatz.

Schulung: Diese Phase wird nur bei Bedarf durchlaufen. Die Vorbereitung hierfür können jederzeit innerhalb des Release-Zyklus beginnen. Die Schulungsverantwortlichen erstellen dann einen Schulungsplan und -unterlagen. Geschult werden u.a. die Systemadministratoren in der Installation des Systems oder die Benutzer, wenn neue Funktionalitäten eingeführt werden.

Release-Empfang: Beim Release-Empfang wird bestätigt, dass der *Change Request* erfolgreich implementiert worden ist. Dazu führt der Release-Manager zusammen mit dem Release-Projektleiter einen Review durch.

2.3.3.1.2 Einordnung von EM^3 in das Lebenszyklusmodell

Abbildung 2.56 zeigt die Einordnung von EM^3 in das Lebenszyklusmodell. Der Schwerpunkt liegt hier auf der Endphase der Softwareentwicklung mit Integration und Systemtest und der Einführung eines Systems beim Kunden. Die Softwareentwicklung an sich wird in diesem Modell nicht beschrieben. Ebenso wird die eigentliche Betriebsphase nicht adressiert. Innerhalb der Betriebsphase jedoch beschreibt EM^3 den Prozess der Release-Steuerung.

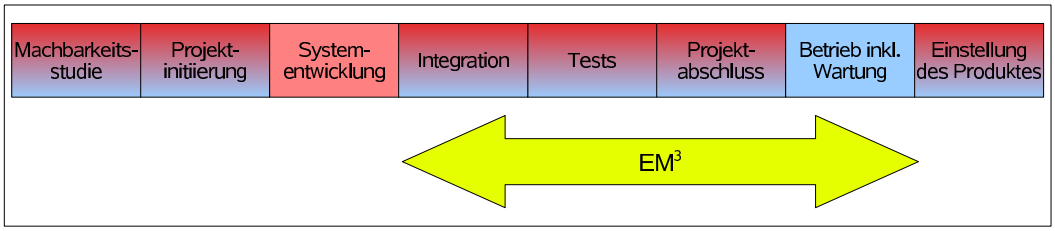


Abbildung 2.56: Einordnung von EM^3 in das Lebenszyklusmodell

2.3.4 IT Infrastructure Library

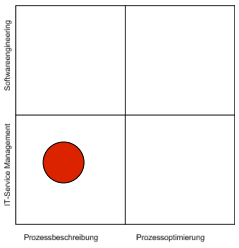


Abbildung 2.57: Einordnung von ITIL

Die IT Infrastructure Library (ITIL) [OGC04, vKP 02, Olbr 04, Kö 05] besteht aus einer Sammlung von Büchern über praxisbewährte Prozesse des ITSM. ITIL wurde im Auftrag der britischen Regierung mit dem Ziel der Normierung von IT Prozessen entwickelt. Das ITSM in ITIL ist in eine strategische Ebene, eine taktische Ebene und in eine operationelle Ebene unterteilt. In der strategischen Ebene werden IT-Dienstleistungen gemanagt. Die taktische Ebene plant und steuert IT-Dienstleistungen. Diese Ebene ist im *Service Delivery* Buch abgebildet. Die operationelle Ebene wird durch den *Service Support* beschrieben [ITIL03, ViG 04]. In der Praxis haben die Bereich Service Support und Service Delivery die meiste Verbreitung. Service Delivery beinhaltet die Bereitstellung von IT Services, so dass diese den Kundenerwartungen entsprechen. Sichergestellt werden soll dies durch die Prozesse *Service Level Management, Capacity Management, Availability Management, IT Continuity Management und Financial Management*. Das Service Level Management ist für das Entwickeln, Überwachen und Pflegen von *Service Level Agreements* (SLA) zuständig. SLA definieren und beschreiben das Produktspektrum der IT-Organisation [ViG 04]. Sie haben damit einen wesentlichen Einfluss auf die Gestaltung der anderen Prozesse. Service Support hingegen ist zuständig für die effiziente Bereitstellung der vereinbarten Leistungen [ITIL03]. Diese beiden Servicebereiche beinhalten die Beschreibung von notwendigen Schlüsselvorgängen für die Geschäftsprozesse einer Firma.

2.3.4.1 Release Management nach ITIL

In ITIL ist das RM ein Bestandtteil des Service Supports. Weitere Prozesse im Service Support sind *Incident-Management, Problemmanagement, Konfigurationsmanagement und Change-Management*.

Die Aufgabe des RM ist der Schutz der Produktivumgebung und ihrer Services durch Anwendung formaler Abläufe und Prüfungen bei der Implementierung neuer Versionen. Das RM arbeitet eng mit dem *Konfigurationsmanagement* und dem *Change-Management* zusammen (siehe Abbildung 2.58). Das Konfigurationsmanagement interagiert mit allen Prozessen des Service Support und Service Delivery [Olbr 04]. Es sorgt durch Bereitstellung von Informationen über alle Komponenten der IT-Infrastruktur für ein effizienteres und effektiveres Management. Besonders bei der Durchführung von Änderungen wird die Analyse der Auswirkungen von Änderungen mithilfe der Einträge (*Configuration Items (CI)*) in der *Configuration Management Database (CMDB)* durchgeführt. Die CMDB beinhaltet hierfür die Informationen zu Beziehungen zwischen der zu ändernden Einheit und den restlichen Komponenten der IT-Infrastruktur. In der CMDB stehen außerdem Informationen über Hardwarespezifikationen, Installationsanleitungen und Netzwerkkonfigurationen. Das RM wird durch die CMDB unterstützt durch Speicherung von Informationen:

- zum Inhalt von geplanten Releases einschließlich Hard- und Software CI und dem Bezug zum ursprünglichen *Request for Change (RfC)*
- zu Hard- und Software CI, auf die sich das Release auswirkt
- zu Ortsinformationen von Hardware, die vom Release abgedeckt wird
- zu *Configuration Baselines*, welche einen Schnappschuss einer definierten Gruppe von CI zu einem bestimmten Zeitpunkt entsprechen; die Baseline dient dann als Grundlage zur Entwicklung oder zum Testen neuer Konfigurationen oder als Back-out, falls es mit der neuen Konfiguration Probleme gibt; sie kann auch eine Standardkonfiguration z.B. einen Standardarbeitsplatz definieren der an die Benutzer verteilt wird; die *Baseline* ist auch die Ausgangslage bei der Verteilung neuer Software

Das Change-Management stellt standardisierte Methoden und Prozeduren zur Behandlung von Änderungen bereit. Durch das Abwägen des Bedarfs einer Änderung gegen die möglichen Auswirkungen einer Änderung soll die Servicequalität verbessert werden. Durch diese Zusammenarbeit der Prozesse soll die Aktualität der gemeinsam verwendeten CMDB sichergestellt werden.

Durch die Anwendung des RM zusammen mit dem Change- und Konfigurationsmanagement ergeben sich u. a. folgende Vorteile ([OGC04, Olbr 04, vKP 02]):

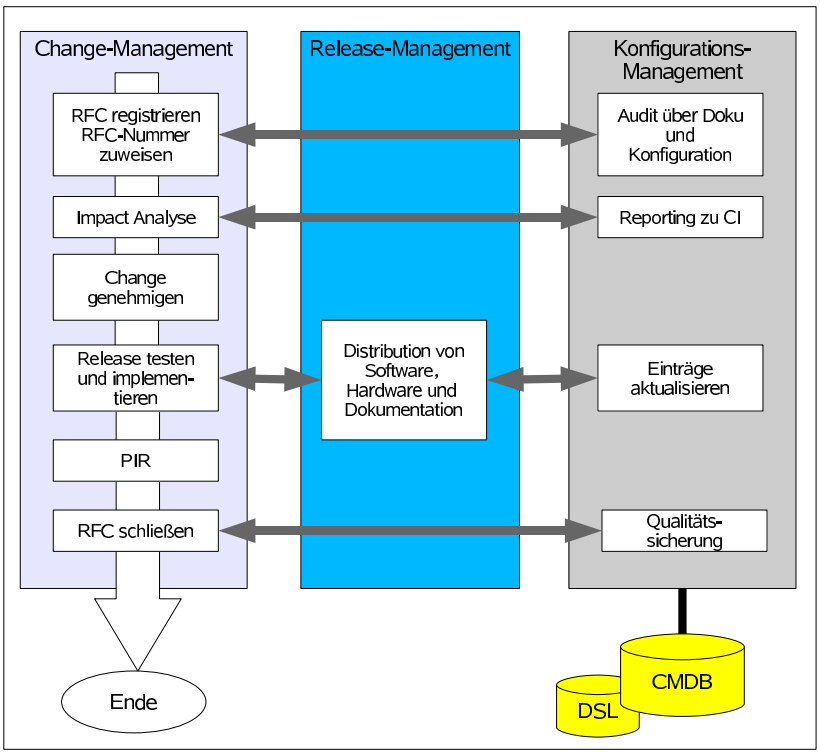


Abbildung 2.58: ITIL Prozesse Release- , Change- und Konfigurations-Management nach [Olbr 04]

- Soft- und Hardware in der Produktivumgebung sind von hoher Qualität, da sie unter der Qualitätskontrolle entwickelt und getestet wurden, ehe sie eingeführt wurden
- das Risiko von Fehlern in Soft-/Hardwarekombinationen und der Einführung von inkorrekten Versionen ist minimiert
- das Risiko des Auftritts von *Incidents* und *Known Errors* wird durch das Testen und durch die kontrollierte Einführung minimiert
- die Benutzer werden mehr in das Testen einbezogen
- es gibt einen reibungsloseren Übergang der Releases von der Entwicklung in die Kundenumgebung
- die Benutzererwartung ist konform mit den Releases, da die Einführungstermine vorab bekannt gemacht werden
- eine Standardisierung von Soft- und Hardwareversionen zwischen unterschiedlichen Einsatzorten ist möglich; dadurch wird der Support einfacher
- nicht genehmigte Kopien und falsche Versionen werden einfacher gefunden

Im folgenden Abschnitt erfolgt zunächst ein Überblick über die wichtigsten Begriffe des ITIL Release-Managements. Anschließend wird der Release-Prozess von ITIL beschrieben.

2.3.4.2 Wichtige Begriffe des ITIL Release-Management

Nachfolgend werden die wichtigen ITIL Begriffe und ihre Definition aufgeführt.

2.3.4.2.1 Release Das *Release* bündelt eine Menge von genehmigten Änderungen an IT-Services. Es wird durch den *Request for Change (RfC)* definiert, den es umsetzt. Das Release besteht häufig aus diversen Pro-

blembehebungen und Serviceerweiterungen. Um die Änderung zu implementieren, sind somit im Release neue und geänderte Software oder Hardware enthalten. Releases werden oft weiter unterteilt in:

Major-Release Das *Major-Release* ist ein umfassender Roll-out von neuer Hard- und Software. Ein Major-Release erfolgt oft in Zusammenhang mit einer signifikanten Anzahl neuer Funktionalität. Solche Releases beheben oft mehrere *Known Errors* mit Umgehungslösungen (*Work-Arounds*) und *Emergency Fixes*.

Minor Software Releases und Hardware Upgrades *Minor Software Releases und Hardware Upgrades* beinhalten meist mehrere geringfügige Verbesserungen und Behebungen von Known Errors. Einige wurden möglicherweise bereits früher als *Emergency Fix* implementiert und nun werden sie durch dieses Release umfassend behandelt. Sowohl durch Major- als auch durch Minor-Releases werden vorangegangene Emergency Fixes und Umgehungslösungen abgelöst. Damit wird der *Previous Trusted State* [vKP 02], d.h. die *Baseline* als neuer Ausgangspunkt aktualisiert.

Emergency Fix *Emergency Fixes* werden üblicherweise zur schnellen Behebung von Problemen oder Known Errors durchgeführt.

2.3.4.2.2 Release-Einheit Die *Release-Einheit (Release Unit)* bestimmt die Menge der Bestandteile der IT Infrastruktur, welche normalerweise gemeinsam bearbeitet werden [OGC04].

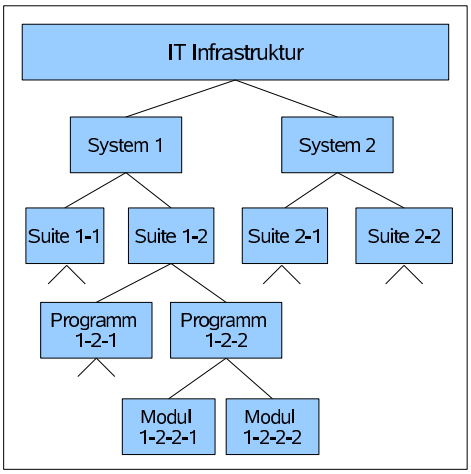


Abbildung 2.59: Software-Release-Einheiten nach [OGC04]

Bei Hardware muss etwa festgelegt werden, ob ein kompletter PC an sich oder einzelne Hardwarebauteile für sich ausgetauscht werden. Abbildung 2.59 zeigt die verschiedenen Möglichkeiten zur Festlegung der Release-Einheit bei Software. Bei Software können Änderungen auf System-, Suite-, Programm- oder Modulebene stattfinden.

Das RM legt hierbei Regeln fest, auf Basis welcher Release-Einheit das Release durchgeführt wird. Entscheidungskriterien dabei sind u.a. der nötige Ressourcenverbrauch für den Build-Prozess, das Testen, Verteilen und Implementieren oder der verfügbare Speicherplatz in der jeweiligen Umgebung.

2.3.4.2.3 Release-Typ Des Weiteren unterscheidet ITIL die folgenden Release-Typen Voll-Release, Delta-Release und Package-Release.

Voll-Release Beim *Voll-Release (Full Release)* wird eine Release-Einheit in seiner Gesamtheit, d.h. einschließlich unveränderter Module, verteilt. Alle Komponenten der Release-Einheit durchlaufen zusammen den Build-Prozess, werden gemeinsam getestet, verteilt und implementiert [OGC04]. Abbildung 2.60 zeigt ein Voll-Release auf Suitelevel. Hier bilden die geänderten Elemente (rot in der Grafik) zusammen mit den unveränderten das Voll-Release. Zu diesem Ansatz wird geraten, wenn nicht alle Änderungen ganzheitlich bekannt sind. Sowohl die Soft- als auch die Hardware wird ausführlicher getestet und nach der Einführung sind weniger *Incidents* zu erwarten. Hier kann bereits in der Vorbereitungsphase überprüft werden, ob Performanzkriterien eingehalten werden. Der Vorteil ist hier, dass mehrere Änderungen simultan implementiert werden können. Allerdings ist der Vorbereitungs- und Ressourcenaufwand höher, als beim Delta-Release [vKP 02].

Delta-Release Das *Delta-Release (Delta Release)* beinhaltet ausschließlich die seit dem letzten Release geänderten oder neuen CI der Release-Einheit[OGC04].

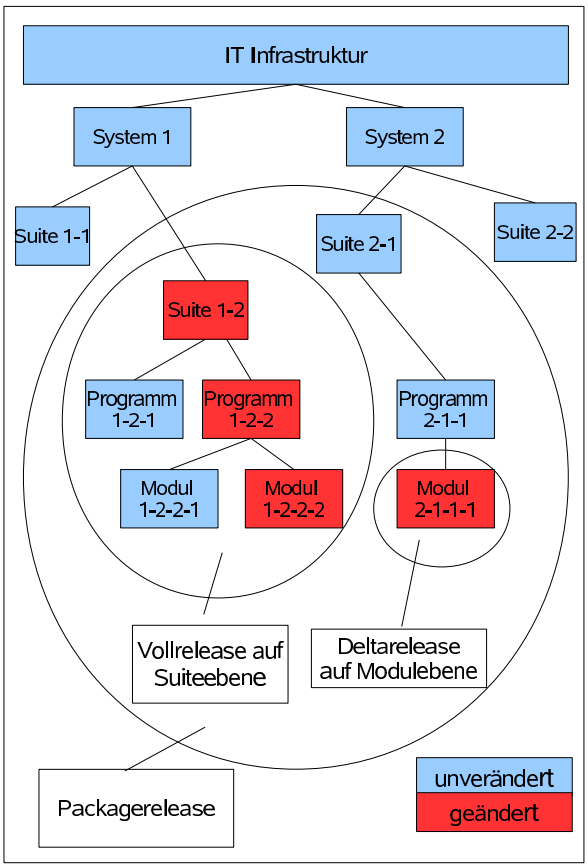


Abbildung 2.60: Beispiel: verschiedene Release-Typen in ITIL

In Abbildung 2.60 ist ein Delta-Release auf Modulebene dargestellt. Auch wenn als Release-Einheit die Suiteebene festgelegt wurde, wird beim Delta-Release nur das geänderte Modul betrachtet. Das Delta-Release ist geeignet, wenn die Software vom Rest der IT-Umgebung isoliert werden kann.

Der Vorteil hier ist, dass der Aufbau der Testumgebung weniger Arbeit benötigt. Als Nachteil wird genannt, dass es oft nicht möglich ist, alle Verbindungen mit der restlichen Umgebung zu testen und Module, die evtl. nicht mehr benötigt werden, nicht gelöscht werden [vKP 02].

Package-Release: Das *Package-Release* oder Release-Bündel kann sowohl Delta- als auch Voll-Releases beinhalten. Mit ihm sollen durch reduzierte Release-Zyklen längere stabilere Betriebsphasen für den Benutzer erreicht werden. Abbildung 2.60 zeigt eine Bündelung von Suite1-2 und Suite2-1 zu einem Release. Durch die Bündelung von Releases ist z.B. das Testen des Zusammenspiels von Systemen möglich. Es sollte hier lediglich darauf geachtet werden, dass die Menge der Änderungen in einem überschaubaren Rahmen bleiben [OGC04].

2.3.4.2.4 Release-Bezeichung Releases werden aufgrund der Festlegungen in der Releasepoliy bezeichnet. Aus der Release-Bezeichnung muss erkennbar sein, welche CI in diesem Release enthalten sind. Die CI sind durch Name und Versionsnummer identifizierbar.

2.3.4.2.5 Definitive Software Library Die *Definitive Software Library* (DSL) ist ein sicherer Aufbewahrungsort für die endgültig authorisierten Versionen aller Software CI. Sie kann physisch aufgeteilt sein und z.B. mehrere Magazine und feuerfeste Tresore umfassen. Die Releases selbst werden aus Software zusammengestellt, die sich in der DSL befindet. In der DSL können mehrere Versionen der gleichen Software inklusive dazugehöriger Dokumentation und Quelldateien aufbewahrt werden. Aus diesem Grund sollte auch die DSL regelmässig gesichert werden, da sie nicht nur die im Einsatz befindliche Software, sondern auch die Back-out-Versionen beinhaltet. Falls es mehrere Einsatzorte mit lokaler Administration gibt, hat jeder dieser Orte für den Roll-out eine Kopie der DSL.

2.3.4.2.6 Definitive Hardware Store Das *Definitive Hardware Store (DHS)* ist ein Lagerort für geprüfte und freigegebene Hardware (Ersatzteile)[ITIL03]. Die Ersatzteile im DHS werden immer dann gewartet, wenn auch in der Produktivumgebung eine Wartung erfolgt. Die eingelagerte Hardware kommt zum Einsatz, wenn in der IT Infrastruktur ein Teil ersetzt oder repariert werden muss. Die Konfigurationsdetails dieser Hardware sollten auch in der CMDB stehen. Da die physische Lagerung aller Ersatzteile sehr kostenintensiv sein kann, sollten dort nur kritische Ersatzteile gelagert werden [Olbr 04]. Für Standardhardware kann das DHS einer Datenbank mit den relevantenen Informationen, z.B. Lieferaten- oder Bestelldaten, entsprechen.

2.3.4.3 Release-Prozess in ITIL

Das RM ist zuständig für:

- Planung, Koordination und Implementierung bzw. Steuerung der Implementierung von Soft- und Hardware
- Entwurf und Realisierung von effizienten Abläufen der Verteilung und Installation von Änderungen an der IT-Infrastruktur
- Sicherstellung, dass von Änderung betroffene Hard- und Software rückverfolgbar und gesichert ist und dass nur korrekte, authorisierte und getestete Versionen installiert werden [vKP 02]
- Kommunikation mit den Benutzern bzgl. deren Erwartungen zur Planung und Roll-out von neuen Releases
- Bestimmung der Zusammensetzung und Planung von Roll-outs, zusammen mit dem Change-Management
- Implementierung neuer Software-Releases und Hardware in der Betriebsumgebung mit Unterstützung des Konfigurationsmanagements und unter Steuerung des Change-Managements. Das Release kann neben Soft- und Hardware auch Dokumentationen, wie etwa Handbücher oder Berichte enthalten
- Sicherstellung der sicheren Aufbewahrung der Orginalsoftware in der DSL und entsprechende Aktualisierung der CMDB. Das gilt analog für Hardware in der DHS [vKP 02]

Eine Übersicht der wichtigsten Aktivitäten des Release-Prozesses zusammen mit den relevanten Umgebungen ist in Abbildung 2.61 dargestellt. Im Vergleich zur Originaldarstellung in [OGC04] ist diese Abbildung komprimierter. In der *Entwicklungsumgebung* werden neue Versionen auf der Basis von früheren Versionen

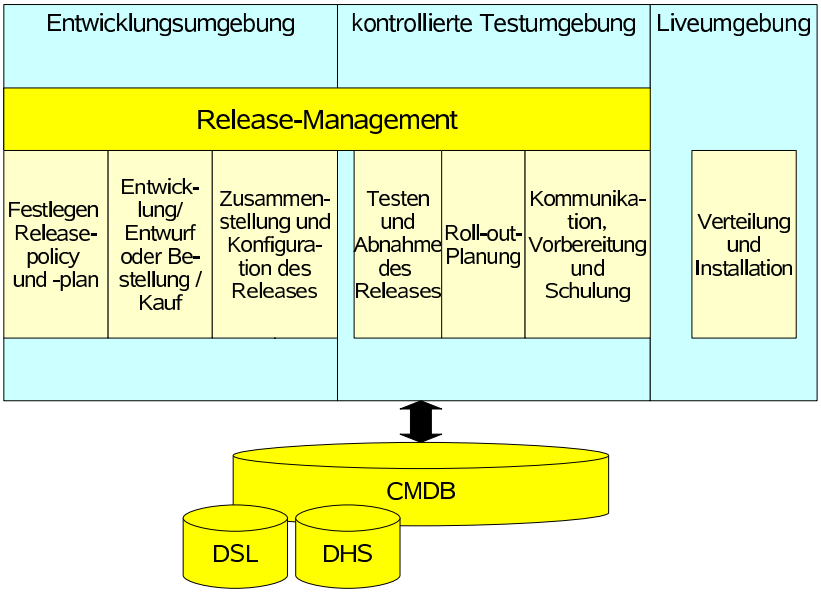


Abbildung 2.61: Aktivitäten im Release-Prozess und betroffene Umgebungen nach [vKP 02]

entwickelt. Die Vorgängerversion wird aus der DSL entnommen. Die Versionsnummer wird nach jeder neuen Version erhöht. Die Software darf nur in der Entwicklungsumgebung geändert werden. In der *Testumgebung* werden die Versionen getestet. Oft wird nochmals unterteilt in technische Tests durch den Entwickler, funktionale Tests durch die Benutzer, Installationstest durch die Release-Entwickler und möglicherweise endgültige Abnahmetests durch die Benutzer und das Management. Die *Produktionsumgebung* ist die Lifeumgebung für die Benutzer. Im folgenden Abschnitt werden die Aktivitäten des RM detaillierter beschrieben.

2.3.4.3.1 Festlegen der Releasepolicy und des Release-Plans Der Release-Manager legt eine Releasepolicy fest. Sie beinhaltet auch die Festlegung der Verantwortlichkeiten im Prozess. Durch diese Policy wird auch bestimmt, wie oft und wann Releases konfiguriert werden. Ein Major-Release kann vorab zusammen mit der Release-Bezeichung oder Versionsnummer geplant werden. Weiterhin wird festgelegt, auf welchem Level CI unabhänging voneinander verteilt werden können. Das hängt wiederum ab von der möglichen Auswirkung auf andere Komponenten, vom zeitlichen und personellen Aufwand des Testens und der Build-Prozedur, dem Installationsaufwand und der Komplexität der Auswirkungen auf andere Soft- und Hardware.

Zur zeitlichen Planung des Releases müssen Informationen über Produktlebenszyklen, über die übergebenen Produkte, über Beschreibung der relevanten IT-Dienste und Dienstleistungsvereinbarungen, über Genehmigungen der relevanten RfC usw. gesammelt werden.

Die Ergebnisse dieser Planungsaufgaben sind Bestandteil des Änderungsplans und beinhalten Pläne für die Releases, Testpläne und Akzeptanzkriterien.

2.3.4.3.2 Entwurf, Release-Zusammenstellung und -Konfiguration Für den Entwurf, Release-Zusammenstellung und -Konfiguration sollten Standardverfahren entwickelt werden. Ein Release kann aus einer Menge von CI bestehen, die sowohl intern entwickelt als auch von Dritten zugekauft worden sind. Die Installationsanweisungen und Konfigurationsanweisungen sollten auch als Bestandteil des Releases behandelt werden und demnach als CI unter die Kontrolle des Change- und Konfigurationsmanagements gestellt werden.

Die Hard- und Software sollten in einer eigenen Umgebung vorab getestet werden, ehe sie in die Liveumgebung gebracht werden. Die Konfiguration der Hard- und Software sollte jederzeit nachvollziehbar sein und muss deswegen dokumentiert werden. Für das Kompilieren und Erzeugen des Images wird häufig standardisierte Hardware reserviert.

Mit dem *Back-out-Plan* werden die Schritte im Fehlerfall des Releases festgelegt. Die Aufgabe des Change-Managements ist die Erstellung dieses Plans. Das RM stellt sicher, dass er praktikabel ist [vKP 02]. Vor allem beim Package-Release kann es nötig werden, auch die einzelnen Back-out Pläne zu kombinieren. Beim Scheitern eines Voll- oder Delta-Releases muss evtl. das ganze Release komplett zurückgerollt werden. Fall der komplette Roll-back nicht möglich ist, sollte ein *Disaster Recovery Plan* sicherstellen, dass der Service wiederhergestellt werden kann.

Das tatsächliche Zusammenstellen des Releases kann das Kompilieren und Verbinden von Softwaremodulen beinhalten, wie auch das Befüllen von Datenbanken mit (Test-)Daten. Das wird oft durch automatische Installationsskripte erledigt. Diese werden zusammen mit den Back-out-Plänen in der DSL abgelegt. Testpläne existieren für Tests, Abnahme der Software, Hardware, Prozeduren, und Roll-out-Skripte vor dem Release-Termin und dem Evaluationstest nach dem Release-Termin. Auch die Installationsskripte werden getestet.

2.3.4.3.3 Testen und Abnahme des Releases Meist liegt die Ursache von unbefriedigenden Änderungen und Releases beim ungenügenden Testen. Zur Abhilfe soll das Release vor der Implementierung funktional durch die Benutzer und operational durch das IT Personal getestet werden. Die operationellen Tests beinhalten technische Operationen, Performanz und Integration mit der restlichen Infrastruktur. Diese Tests sollen auch die Installationsskripte, Back-out-Prozeduren und Änderungen der Managementprozesse abdecken. Eine formelle Abnahme sollte nach jedem Schritt an das Change-Management weitergeleitet werden. Zum Abschluss wird das Release für die Implementierung freigegeben.

Das Release soll in einer kontrollierten Testumgebung mit entsprechender Grundkonfiguration abgenommen werden. Diese Konfiguration wird in der Release-Definition beschrieben und in der CMDB gespeichert. Wenn das Release nicht freigegeben wird, geht es zurück an das Change-Management.

Ergebnis dieser Aktivität sind getestete Installationsprozeduren, getestete Release-Komponenten, Known Errors, Testergebnisse, Management- und Supportdokumentation, getestete Back-out-Pläne, Schulungsprogramm für das Personal, Manager und Benutzer und unterschriebene Abnahmedokumente.

2.3.4.3.4 Planung der Implementierung Der Release-Plan aus den vorherigen Schritten wird nun um Informationen über die Implementierungsschritte ergänzt. Der Roll-out-Plan beinhaltet u.a. einen Ressour-

cenplan (Zeit, Personal, Aufgaben), eine Liste der zu installierenden CI, Benachrichtigung der betroffenen Parteien. Die Implementierung kann als Big Bang erfolgen, d.h. das Release wird komplett ausgerollt oder schrittweise nach funktionalen (alle Benutzer bekommen gleichzeitig neue Funktionalitäten), örtlichen oder evolutionären (die Änderung wird schrittweise ausgerollt) Gesichtspunkten.

2.3.4.3.5 Kommunikation, Vorbereitung und Schulung Dem Personenkreis, der mit den Kunden kommuniziert (*Service Desk, Customer Releation Management*, Betriebspersonal und Vertreter des Benutzerkreises) werden die Pläne und ihre Auswirkungen auf den laufenden Betrieb bekannt gemacht. Die Verantwortlichen werden benannt. Falls das Release schrittweise ausgerollt wird, werden die Benutzer informiert, wann sie die neuen Funktionen erwarten können. Änderungen in *Service Level Agreements (SLA)* , *Operational Level Agreements (OLA)* und *Underpinning Contracts (UC)* werden vorab allen relevanten Personen bekanntgemacht.

2.3.4.3.6 Release-Verteilung und Installation Das RM überwacht die logistischen Prozesse des Einkaufs, Speicherung, Transports, Lieferung und Übergabe der Soft- und Hardware. Dieser Prozess wird durch Abläufe, Dateien und Begleitdokumente unterstützt. Die Lagerorte der Hard- und Software sollten abgesichert sein und nur authorisierten Personen zugänglich sein. Die Benutzung von Tools für die automatische Verteilung und Installation wird empfohlen. Vor der Installation soll die Zielumgebung auf die Release-Bedingungen, wie Speichergrösse, Sicherheit etc. überprüft werden. Nach der Installation wird die CMDB aktualisiert, um Lizenzvereinbarungen verifizieren zu können.

2.3.4.4 Einordung von ITIL in das Lebenszyklusmodell

Abbildung 2.62 zeigt die Einordnung von ITIL in das Lebenszyklusmodell. Prinzipiell deckt ITIL den kompletten Lebenszyklus von Systemen ab. Der Bereich des SWE wird im ITIL-Buch Application Management behandelt. Der Schwerpunkt von ITIL liegt jedoch im operativen Betrieb und auf den Phasen Betrieb inkl. Wartung, Integration und Test. Dieser Bereich wird in den Büchern ITIL Service Support und Service Delivery betrachtet.

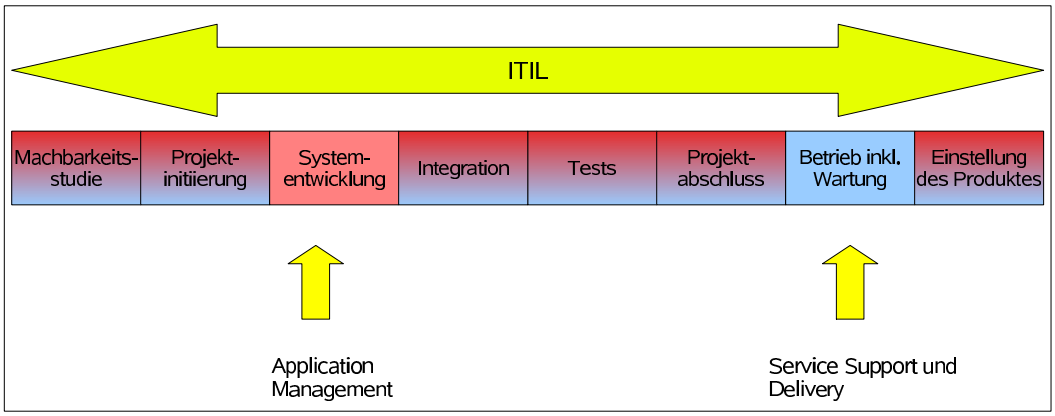


Abbildung 2.62: Einordnung von ITIL in das Lebenszyklusmodell

2.3.5 Control Objectives for Information and Related Technology

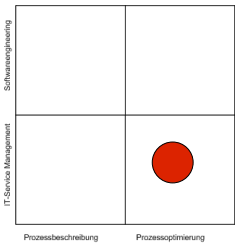


Abbildung 2.63: Einordnung CobIT

Control Objectives for Information and Related Technology (CobIT) [Cob05, Cob01, Cob00] ist ein Referenzmodell für IT-Governance. CobIT definiert eine Menge von Kontrollzielen für IT-Prozesse. Im Gegensatz zu ITIL konzentriert sich CobIT auf die Kontrollaspekte der Prozesse in der gesamten IT-Organisation. Die Organisation Information Systems Audit and Control Association (ISACA) entwickelt CobIT weiter und achtet auf die Konformität zu ITIL [HoHu 03]. Die durch interne und externe Revisionen zu prüfenden Objekte sind im Sinne einer lückenlosen Nachvollziehbarkeit aller Änderungen an der IT-Infrastruktur das Herzstück dieser Systematik.

In seiner aktuellen Version identifiziert das Modell 34 IT-Prozesse, die anhand von 318 Kontroll- und Überwachungsrichtlinien bewertet werden. Über kritische Erfolgsfaktoren (*critical success factor* (CSF)), *Key Performance*-Indikatoren (KPI) und andere Kennzahlen wird dem Bedarf des Managements nach Kontrolle und Messbarkeit der IT Rechnung getragen. Hierdurch kann die IT- Umgebung den von CobIT identifizierten IT-Prozessen gegenübergestellt und beurteilt werden. CobIT ist in die vier Bereiche Planung & Organisation, Beschaffung & Einführung, Deployment & Support und Monitoring eingeteilt. Diese sind wiederum unterteilt in bestimmte Steuerungseinheiten.

2.3.5.1 Release-Management bei CobIT

Das RM ist ein Bestandteil der Domäne Beschaffung & Einführung. Hier wiederum ist das Release-Management ein Bestandteil des Änderungsprozesses. Weitere Einheiten von Beschaffung & Einführung sind z.B. die Installation und Akkreditierung von Systemen. Aufgaben, die bei der Installation und Akkreditierung von Systemen anfallen sind etwa das Testen von Änderungen in einer getrennten Testumgebung, das Erstellen von Back-out-Plänen, das Definieren von formellen Verfahren zur Übergabe von Systemen von der Entwicklungs- in die Test- und schließlich in die Produktivumgebung.

Durch das Änderungswesen werden die Auswirkungen von Änderungen auf die produktiven Systeme beurteilt, ein Rahmen für Notfalländerungen festgelegt oder die Softwareverteilung gesteuert. Mithilfe von messbaren KPI kann dann überprüft werden, ob ein IT-Prozess den Anforderungen genügt. Für das Änderungsmanagement sind z.B. folgende KPI vorgesehen: die Anzahl von Software-Release- und Softwareverteilungsprozessen je Plattform, die Anzahl von *Emergency Fixes*, die in der Nachbetrachtung nicht über den normalen Prozess abgewickelt wurden oder das Verhältnis von akzeptierten zu abgelehnten Änderungen.

CSF unterstützen das Management bei der Einführung der IT-Prozesslenkung. CSF wären hier u.a. Änderungsregelungen, die verständlich und bekannt sind und systematisch eingeführt sind, ein stark integriertes Änderungsmanagement innerhalb des Release-Managements oder die Definition eines formalen Übergabeprozesses von der Entwicklung in den Betrieb.

2.3.5.2 Einordnung von CobIT in das Lebenszyklusmodell

Die vier Domänen von CobIT Planung & Organisation, Beschaffung & Einführung, Deployment & Support und Monitoring wurden so entwickelt, dass sie einen geschlossenen Lebenskreis von IT-Ressourcen und damit auch von Software abdecken. CobIT kann somit den gesamten Bereich des Lebenszyklusmodells abdecken. Auf eine eigene Darstellung der Abdeckumg mit den Lebenszyklusmodell wird deshalb an dieser Stelle verzichtet. Die Entscheidung allerdings, welche und wieviele Prozesse mit CobIT kontrolliert werden, kann individuell erfolgen.

2.3.6 Maturity Model for IT Operations

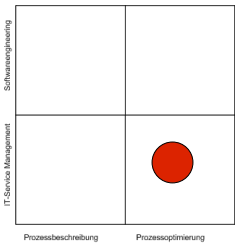


Abbildung 2.64: Einordnung MITO

Das Maturity Model for IT Operations (MITO) [SFS 00, fQFI 03] ist ein Modell zur Reifegradbestimmung für den IT Betrieb. MITO wird von der Swiss Association for Quality (SAQ) entwickelt.

Es definiert auf der Basis der Prozesse von ITIL und den *Code of Practice for IT Service Management* des British Standard Institute erfolgskritische Aufgaben. Diese so genannten *Tasks* bilden als Prüfobjekte die Grundlage für Assessments. Durch die Reifegradbestimmung als Ergebnis des Assessments sollen Möglichkeiten zur Verbesserung deutlich werden. MITO baut auf dem *CMM Modell* von SEI (vgl. Abschnitt 2.2.5) und der Assessmentmethode der European Foundation for Quality Management (EFQM) auf. Analog dem CMM Modell gibt es bei MITO fünf Reifegradstufen (stochastische Produktion, wiederholbare Produktion, nachvollziehbare Produktion, messbare Produktion und Servicestrategie, optimierte Produktion und Strategieverbesserung).

Die Betreiberprozesse werden nach Abbildung 2.65 eingeordnet. Der Fokus liegt hier ausschließlich auf

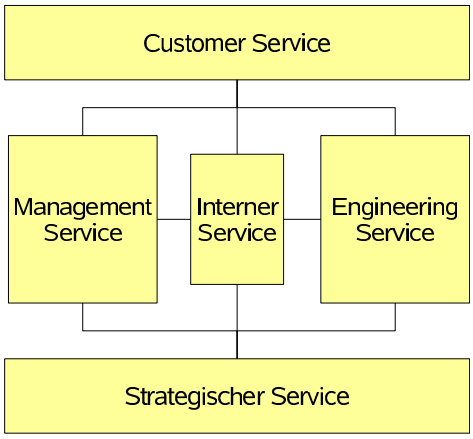


Abbildung 2.65: Prozessgruppen von MITO

Prozessen von Betreiberorganisationen. Die Prozesse des Bereiches *Engineering Services* entsprechen hier den Prozessen des ITIL Service Supports. Sie stellen Entwicklung, Weiterentwicklung und Management des Infrastruktur- und Dienstleistungslebenszyklusses sicher. Die *Engineering Services* sind in Mito weiter aufgeteilt in:

1. Incident Processing
2. Service Change Management
3. Service Design & Update
4. Solution Change & Configuration Management
5. Solution Rollout & Service Deployment

Für diese Prozesse liefert MITO keine konstruktiven Beschreibungen. Das wird als die Aufgabe von ITIL gesehen. Zu jedem dieser Prozesse werden bei MITO erfolgskritische Aufgaben identifiziert. Diese sind dann die Grundlage für Assessments.

2.3.6.1 Release-Management bei MITO

ITIL bildet zwar die Grundlage der Prozessbeschreibung, dennoch wird das RM nicht eins zu eins bei MITO abgebildet. Das RM bei MITO ist aufgeteilt auf das *Service Change Management* und das *Solution Roll-out & Service Deployment*.

Das Ziel des *Service Change Management* ist die geplante und koordinierte Entwicklung und Änderung von Services. Dadurch soll die Stabilität des Betriebes gewährleistet werden. Ein Service wird hier als vermarktbare Dienstleistung definiert. Auslöser dieses Prozesses ist ein Auftrag oder ein *Change Request*. Ein Auftrag ist eine einmalige ungeplante Nachfrage nach einer Dienstleistung. Ein *Change Request* ist der Wunsch nach der Änderung von bestehenden Konfigurationen sowohl von Arbeitsplätzen, als auch von Applikationen. Ergebnis des *Service Change Managements* ist ein Release-Plan, ein *Change Request* an das *Service Design and Update* und statistische Auswertungen über die aufgetretenen Anliegen (Störung, Änderung, Auftrag). Die kritischen Erfolgsfaktoren sind die wirksame Kundenkommunikation, das Erkennen aller vom geplanten Release betroffenen Personen, der Testplan und die Testumgebungen, das Verfahren der Freigabe von Releases für die Produktion und die Mittel für die statistische Auswertung der Anliegen. *Tasks* des *Service Change Management* sind:

- Release-Umfang definieren
- Testvorgehen und -umgebung definieren
- Release-Wechsel planen
- Release-Bereitstellung koordinieren und überwachen
- Daten der Anliegen auswerten
- Maßnahmen anhand dieser Auswertungen einleiten

Das Ziel des *Solution Rollout & Service Deployment* ist die Verteilung, Installation und Inbetriebnahme der geänderten oder neuen *Solution*. Unter *Solutions* werden Mittel (HW, SW, Abläufe) zur Erbringung der Services verstanden. Sowohl *Solutions* als auch Services können geändert werden. Das Release einer neuen *Solution* hat i.d.R. keinen Einfluss auf den Service, während bei einem neuen Service-Release auch neue *Solutions* erstellt werden. Auslöser für den *Solution Rollout & Service Deployment* Prozess ist die Freigabe der Solution. Ergebnis des Prozesses ist die Inbetriebnahme und Installation der *Solution*, die Umstellung oder Neueinführung von Prozessen und der Nachweis der vorgenommenen Installation.

Als kritische Erfolgsfaktoren werden von MITO realistische Inbetriebnahmepläne, Benutzerinformationen über Roll-out- und Deployment-Aktivitäten und die rasche, korrekte und vollständige Verteilung und Installation definiert. *Tasks* des *Solution Rollout & Service Deployment* sind:

- Änderungen an HW Konfiguration durchführen
- SW Konfiguration verteilen und installieren
- in Betriebnahme geänderter oder neuer *Solutions*
- umstellen oder implementieren und schulen von Prozessen im *Customer Service*
- informieren und schulen der Benutzer über Änderungen

2.3.6.2 Einordnung von MITO in das Lebenszyklusmodell

Abbildung 2.66 zeigt die Einordnung von MITO in das Lebenszyklusmodell. MITO deckt die Bereiche von ITIL Service Support und Service Delivery ab. Der Fokus liegt auf den eigentlichen Betreiberprozessen. Der Bereich der Systementwicklung wird von MITO nicht adressiert. Hierfür wird auf das CMM Modell von SEI verwiesen, welches diesen abdeckt.

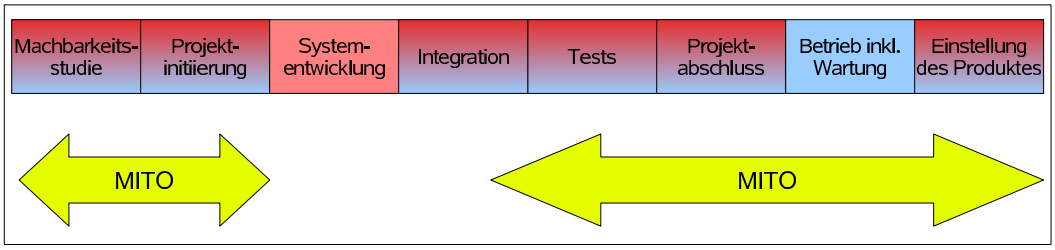


Abbildung 2.66: Einordnung von MITO in das Lebenszyklusmodell

2.3.7 Software Maintenance Capability Maturity Model

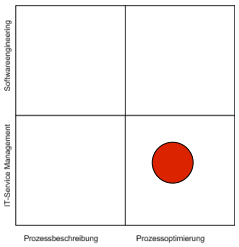


Abbildung 2.67: Einordnung SMCMM

Bei Zitouni et. al. [AbZi 96, AAD 04b, AAD 04a, AADHH 05] beginnen die Softwarewartungsfunktionen (*Software Maintenance*) nach der Fertigstellung der Software durch die Softwareentwicklungsfunktionen. Die Autoren definieren im Rahmen eines Forschungsprojektes an der Ecole de Technologie Superiere de Montreal in Verbindung mit der Otto von Guericke Universität in Magdeburg das *Software Maintenance Capability Maturity Model* (SM^{CMM}) zur Evaluation der Qualität von Softwarewartungsprozessen auf der Basis des CMM des SEI.

Hierfür definieren sie *Key Process Areas* (KPA) , die im CMM Modell fehlen, aber für Wartungsprozesse relevant sind. Abbildung 2.68 zeigt die Kontexteinbettung der *Software Maintenance*.

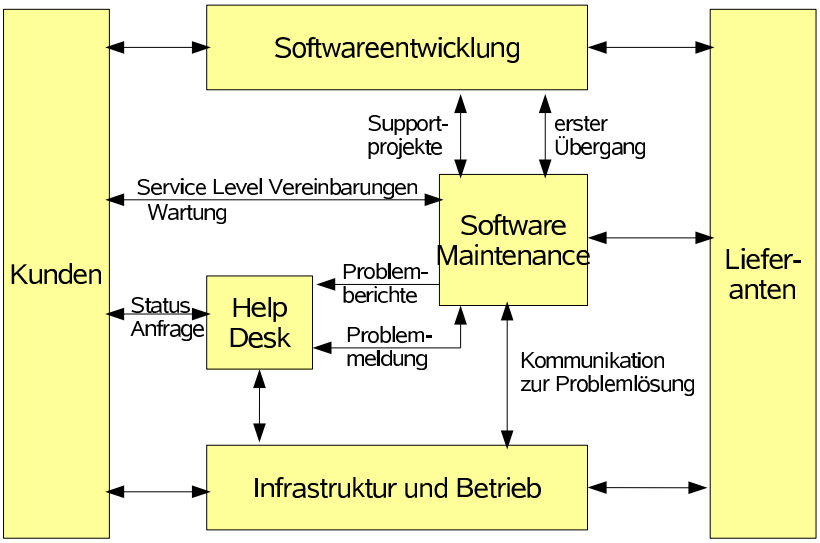


Abbildung 2.68: Kontexteinbettung von Software Maintenance laut [AAD 04b]

2.3.7.1 Release-Management beim Software Maintenance Capability Maturity Model

Der Begriff Release-Prozess wird hier nicht explizit benannt. Statt dessen werden *Transition*- und Versionsmanagement als KPA definiert. Der *Transition*-Prozess steuert und koordiniert die Abläufe der Tätigkeiten der Systemübergabe von den Entwicklern zur Wartung. Durch den Versionsmanagement- bzw. Migrationsprozess [AADHH 05] werden die Systeme in die Produktion gebracht und mit dem *Retirement*-Prozess aus der Produktion genommen. Weitere KPA sind z.B. das Problemmanagement, die Akzeptanz der Software oder das Schulungsprogramm. Um eine Einstufung des Wartungsprozesses auf Stufe Zwei (*Repeatable*) analog des

CMM-Modells zu erhalten, ist hier u.a. die Implementierung der KPA *Transition* nötig. Innerhalb der jeweiligen KPA sind wiederum verschiedene *Key Practices* definiert. Bei der KPA *Transition* ist eine *Key Practice* z.B. ein formaler Abnahmeprozess der Software.

Abran et. al. [AAD 04b] haben die von Zitouni et. al. [AbZi 96] vorgeschlagene Version mit Inhalten aus verschiedenen *Software Maintenance* relevanten Gebieten ergänzt. Input hat hier u.a. ITIL Service Support, ITIL Service Delivery oder CobIT geliefert.

2.3.7.2 Einordnung von SM^{CMM} in das Lebenszyklusmodell

Abbildung 2.69 zeigt die Abdeckung des Lebenszyklusmodells. Adressiert werden von SM^{CMM} die Testphase, Betriebsphase inkl. Wartung und die Einstellung von Produkten.

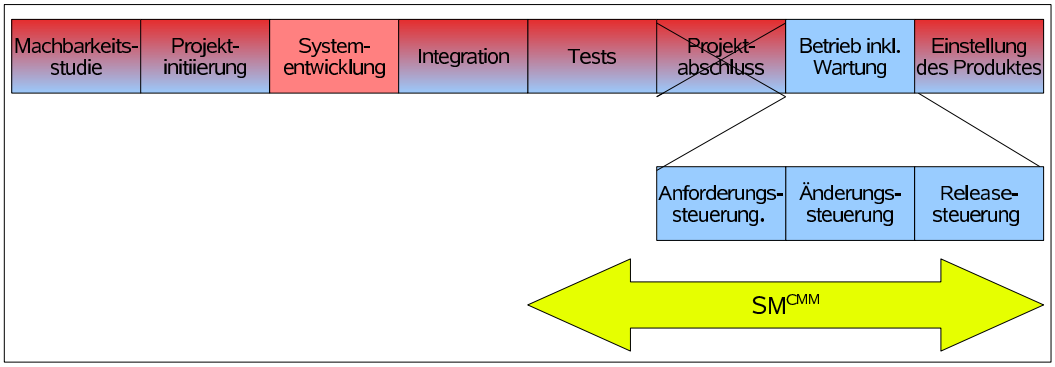


Abbildung 2.69: Einordnung von SM^{CMM} in das Lebenszyklusmodell

2.3.8 IT Service Capability Maturity Model

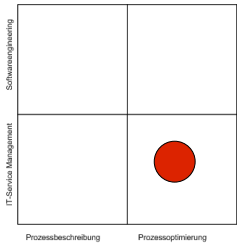


Abbildung 2.70: Einordnung IT-SCMM

Das *IT Service Capability Maturity Model* [NCv 05] ist ein Projekt des niederländischen Wirtschaftsministerium und wird von der Technischen Universität Delft und Einhoven und der Vrije Universität in Amsterdam zusammen mit Partnern aus der Wirtschaft entwickelt. Mithilfe des IT Service CMM kann der Reifegrad von IT-Serviceunternehmen bestimmt werden. Es dient nicht zur Bestimmung der Reife einzelner Services, Projekte oder Organisationseinheiten.

Grundlage für dieses Modell sind u.a. auch Forschungsarbeiten von Niessink et. al. [Niva 98, Niva 00]. Sie unterscheiden zwischen IT Serviceunternehmen, die Services bereitstellen und Softwareentwicklern, die keine Services sondern Produkte bereitstellen.

Dieser Unterschied wirkt sich auf die Beurteilung der Qualität durch den Kunden aus. Bei Services spielt die technische Qualität (das Ergebnis eines Services) und die funktionale Qualität (wie der Service bereitgestellt wurde) bei der Bewertung eine Rolle. Unterschiede zwischen der wahrgenommenen und gelieferten Serviceleistung lassen sich durch das in Abbildung 2.71 gezeigte Modell darstellen.

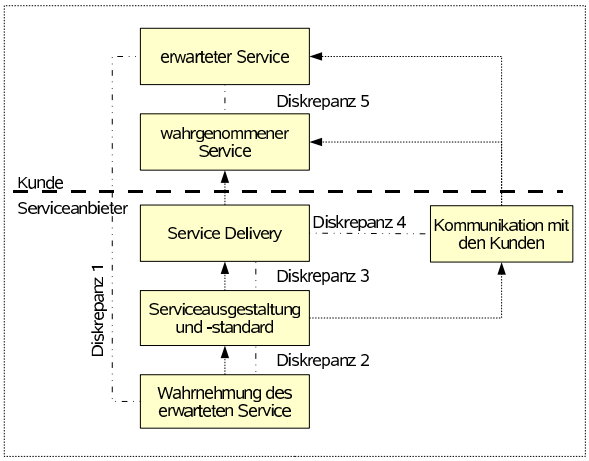


Abbildung 2.71: Diskrepanz zwischen wahrgenommener und gelieferter Servicequalität nach [Niva 00]

Diese Abbildung stellt die Diskrepanz 5 als Summe der anderen Diskrepanzen dar. Die Diskrepanz 1 resultiert aufgrund von Mängeln bei der Marktforschung oder der Kommunikation zwischen Kunde und Management. Die zweite Lücke entsteht aufgrund von nichtkundenorientierten Standards, mangelnden Prozessmanagement, fehlender formaler Prozesse zur Bestimmung von Servicequalitätszielen, schlechten Servicedesign oder schlechter Serviceleistung. Wenn die Dienstbereitstellung nicht nach den spezifizierten Standards erfolgt, entsteht die dritte Diskrepanz. Ursache hiervon können z.B. Schwachstellen in der Personalpolitik oder Schwierigkeiten bei der Abstimmung von Nachfrage und Angebot sein. Die vierte Lücke entsteht durch mangelhaftes Managen der Kundenerwartungen, falsche Versprechungen oder ungenügender horizontaler Kommunikation.

Um diese Lücken zu schließen sind folgende Maßnahmen beim Dienstleister nötig:

Diskrepanz 1: Erfassen der Kundenerwartungen an den Service im Rahmen klarer Servicevereinbarungen

Diskrepanz 2: Diese Servicevereinbarungen werden als Grundlage zur Planung und Implementation der Dienstleistung verwendet

Diskrepanz 3: Sicherstellen, dass die Serviceleistung entsprechend der definierten Pläne und Prozesse erbracht wird

Diskrepanz 4: Managen der Kommunikation über die erbrachten Dienstleistungen

Bei der Wartung von Software werden deshalb im Vergleich zur Softwareentwicklung andere und zusätzliche Prozesse gebraucht, um eine hohe Servicequalität zu liefern. Zur Unterscheidung einer erfolgreichen Wartungsorganisation von einer weniger erfolgreichen führen Niessink et. al. das Reifegradmodell ein.

2.3.8.1 Release-Management beim IT Service Capability Maturity Model

Services, die hier im Rahmen der Wartung anfallen sind die Wartung von Hardware, die Softwareverteilung und der Betrieb eines Computerzentrums. Innerhalb der Wartungsplanung fällt auch die Release-Planung an. Beim Release-Prozess werden hier in nicht kritischen Situationen Änderungen zu einem Release gebündelt und terminiert. Die Prozesse für eine service-orientierte Wartungsorganisation werden auf der Grundlage von ITIL gesehen. Allerdings schlagen die Autoren zusätzlich das *IT Service Capability Maturity Model* vor. Hierdurch soll analog dem CMM Modell der Reifegrad der Serviceprozesse in fünf Kategorien eingeordnet werden können. Dadurch soll ein Ansatz zur Verbesserung der Serviceprozesse gegeben werden. Für z.B. eine Einstufung auf Level Zwei müssen Unternehmen folgende drei Supportprozesse implementieren. Da fast alle IT Services das Management, den Betrieb oder Wartung von Hard- und Software beinhalten, werden die Komponenten unter die Steuerung des Konfigurationsmanagement gebracht. Dadurch ist jederzeit der Status aller Komponenten bekannt. Als zweiter Prozess ist das Event-Management vorgesehen. Es sorgt dafür, dass Ereignisse, die während der Dienstbereitstellung auftreten, behandelt werden. Ereignisse können hier einfache *Requests for Service* oder schwerwiegende *Incidents* sein. Diese Ereignisse (*Events*) müssen identifiziert, überwacht, gelöst und an den Kunden berichtet werden. Daraus resultierende *Change Request* werden durch das *Configuration Control Board* bezüglich des *Service Level Agreements* und des Risikos bewertet. Falls die Änderung genehmigt wird, wird die Konfiguration geändert. Als dritten Prozess auf Level Zwei ist die Qualitätssicherung vorgesehen. Dadurch werden Qualitätssicherungstechniken wie Reviews oder Audits vorgesehen.

2.3.8.2 Einordnung des IT Service Capability Maturity Model in das Lebenszyklusmodell

In den Arbeiten von Niessink et. al. steht der eigentliche Betrieb im Vordergrund. Aus diesem Grund lässt sich dieses Modell wie in Abbildung 2.72 in das Lebenszyklusmodell einordnen.

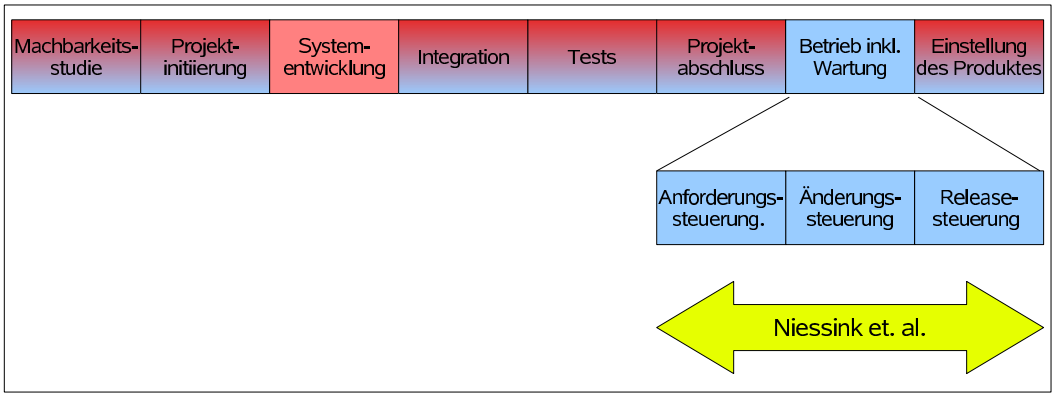


Abbildung 2.72: Einordnung des Modells von Niessink et. al. in das Lebenszyklusmodell

2.3.9 Weitere Beispiele für Release-Management innerhalb von IT-Servicemanagement

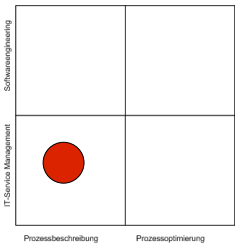


Abbildung 2.73: Einordnung der Prozesse

In diesem Abschnitt erfolgen weitere Beispiele von Prozessen, die das RM als Bestandteil innerhalb des ITSMs sehen. Alle aufgeführten Beispiele sind Bestandteil beschreibender Prozesse und somit wie in Abbildung 2.73 zu klassifizieren.

2.3.9.1 Release-Prozess bei der Object Management Group

Bei nachfolgender Beschreibung ist der Release-Prozess ebenfalls ein Bestandteil des so genannten Deployment-Prozesses. Im Vergleich zu Abschnitt 2.2.6.6 ist es hier jedoch dem ITSM zugeordnet.

Die Object Management Group (OMG) beschreibt eine Spezifikation zum Deployment und zur Konfiguration von komponentenbasieren verteilten Applikationen [Obje 03]. Die Beschreibung umfasst Spezifikationen für verschiedene Modelle (*component model, target model, execution model*). Diese sollen als Grundlage für die Herstellung von Werkzeugen für den Deployment-Prozess dienen.

Es wird hier sowohl ein plattformunabhängiges, als auch ein plattformspezifisches Modell für *Common Object Request Broker Architecture* (CORBA) -Komponenten beschrieben. Der Deployment-Prozess ist hier der Prozess zwischen dem Erwerb und der Ausführung von Software. Der Release-Begriff wird hier nicht explizit verwendet. Zu den Aufgaben des Deployment-Prozesses gehören u.a. die Entgegennahme der Software und ihre Konfiguration für die Zielumgebung, die Beschreibung der Einrichtung der Zielumgebung, die Planung, wie die Software in die Zielumgebung eingeführt wird, das Vorbereiten der Software für die tatsächliche Ausführung, wie etwa Verbringen der Software an den eigentlichen Ausführungsort und das Starten, Überwachung und Abschalten von Anwendungen. Bis zur eigentlichen Einsatzentscheidung wird die Software in einem definierten *Repository* abgelegt. Das kann auch eine Staging-Umgebung sein. In dieser können vorab z.B. Sicherheitsprüfungen durchgeführt werden.

2.3.9.1.1 Erweiterung von Bulej et. al. Bulej et. al. [BuBu 05] erweitern diese Spezifikation um Softwarekonnektoren. Diese sollen das Deployment von heterogenen Komponenten unterstützen, indem sie als Brücke zwischen den heterogenen Bestandteilen der Applikation fungieren.

2.3.9.1.2 Erweiterung von Hnetynka Hnetynka [Hnet 04] erweitert die OMG Spezifikation mit *Features*, die für eine vereinheitlichte Deployment-Umgebung für komponentenbasierte Applikationen relevant sind. Diese *Features* beinhalten Versionierung, Verhalten der Komponenten und dynamische Änderung der Anwendungen zur Laufzeit.

2.3.9.1.3 Einordnung des Modells von OMG in das Lebenszyklusmodell Die Aufgaben des Deployment-Prozesses werden hier aus der Sichtweise des Softwarebenutzers beschrieben. Sie starten mit der Entgegennahme und enden mit der Abschaltung dieser Software. Aus diesem Grund wird der Prozess von OMG wie in Abbildung 2.74 eingeordnet.

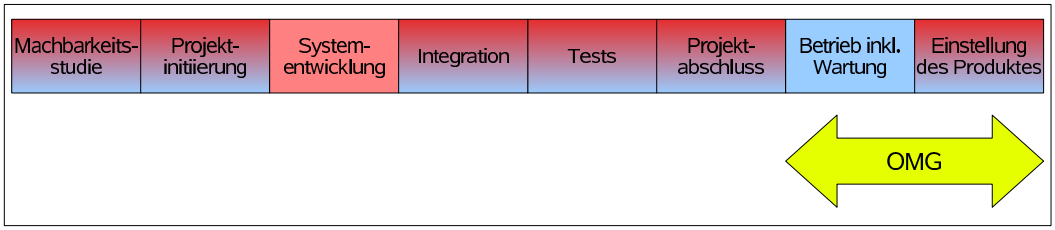


Abbildung 2.74: Einordnung des OMG Modells in das Lebenszyklusmodell

2.3.9.2 Release-Management innerhalb des Wartungsmanagement bei Curth et. al.

Curth et. al. [CuGi 89] definieren in ihrem Wartungsmodell die Wartungsphase als Bestandteil der Nutzungsphase. Sie schlagen die Einrichtung einer eigenständigen und unabhängigen Wartungsabteilung vor. Diese soll organisatorisch von der Entwicklungsabteilung getrennt sein, damit nicht Neuentwicklungen höher priorisiert werden als die Wartung bestehender Systeme. Der Wartungsprozess soll genau wie die Entwicklungsaktivitäten geplant, koordiniert und auftragsbezogen abgewickelt werden. Als Maßnahme schlagen sie vor, dass Änderungsanträge durch den Wartungsmanager klassifiziert und anhand der Klassifikation priorisiert werden.

Die Einteilung der Änderungsanträge erfolgt in vier Kategorien. Die erste Kategorie sind schwerwiegende Fehler, die sofort korrigiert werden müssen („emergency repair“). In die zweite Stufe werden kurzfristig durchzuführende Änderungen eingeordnet („corrective coding “). Die dritte Kategorie beinhaltet mittelfristig durchzuführende Systemerweiterungen („upgrades “) und die vierte Restrukturierungen („system redesign “). Bei Änderungsanträgen der vierten Kategorie wird die Wartungsaufgabe zunächst vom Wartungsteam analysiert. Aufgrund dieser Analyse entscheidet der Wartungsmanager, ob das System neu entwickelt oder geändert wird. Der Wartungsmanager bildet bei diesem Prozess die Schnittstelle zwischen der Entwicklung und den Fachabteilungen. Er plant den Wartungsablauf, ist zuständig für die Realisierung und die Kontrolle des Prozess.

2.3.9.2.1 Release-Management bei Curth et. al. Curth et. al. benutzen nicht Release als Begriff. Sie haben die Phasen und Aktivitäten der Wartung von Software definiert. Dazu gehören laut Abbildung 2.75 die Phasen der Integration und der Einführung.

Abbildung 2.75: Phasen der Softwarewartung aus [CuGi 89]

In ihrem Modell ist das Wartungsmanagement für organisatorische Maßnahmen zuständig. Die Qualitätssicherung liefert Qualitätskriterien für den Wartungsprozess und für die Abnahme der geänderten Software. Durch die Dokumentation werden alle Wartungsvorgänge dokumentiert. In der Integration werden getestete Programme in ein bestehendes Programmpaket integriert. Anschließend erfolgt ein Integrationstest aller Programme, auch der nicht geänderten. Dann werden die Programme eingeführt. Die Autoren schlagen eine schrittweise Einführung vor, so dass das bestehende System notfalls als Fall-Back-Lösung verwendet werden kann. Der Endbenutzer enthält auch die geänderten Benutzerdokumentation. Besondere Problempunkte werden in der Wartungsdokumentation beschrieben. Bei Bedarf werden die Anwender neu geschult. In der ersten Zeit nach der Einführung wird dann noch vom Wartungsteam ein Hotline-Service angeboten.

Abbildung 2.76 zeigt, welchen Bereich des Lebenszyklusmodells die Wartung nach Curth et. al. abdeckt.

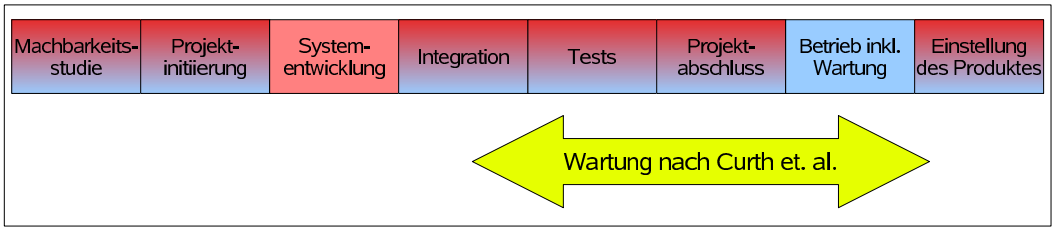


Abbildung 2.76: Einordnung der Wartung nach Curth et. al. in das Lebenszyklusmodell

2.4 Zusammenfassung

Obige Definitionen und Prozessbeschreibungen zeigen, dass kein einheitliches Verständnis des Release-Begriffs und des Release-Prozesses vorhanden ist. Es gibt prinzipiell zwei Sichtweisen. Die SWE- und die ITSM-Sicht. RM hat u.a. einen Fokus auf die Schnittstelle zwischen einer Entwicklungsorganisation und dem Kunden. Beim SWE endet meist mit dem RM das Projekt Systementwicklung während beim ITSM der Systembetrieb mit dem RM startet. Abbildung 2.77 zeigt zusammenfassend die Einordnung der vorgestellten Modelle.

Mehrheitlich wird das RM aus Sicht des Softwareherstellers betrachtet. Die vorgestellten Modelle stellen allerdings nur eine Auswahl aus dem Bereich des SWE dar. Bei einer Schlagwortsuche nach dem Begriff SWE wurden im Bayrischen Bibliothekenverbund über eintausend Literaturhinweise geliefert. Dem stehen gerade 17 Bücher zum Schlagwort Dienstleistung-Management-Informationstechnik, unter dem z.B. auch die ITIL-Literatur gepflegt ist, gegenüber.

Abbildung 2.78 zeigt die Übersicht der Einordnung einer Auswahl der vorgestellten Referenzmodelle in das Lebenszyklusmodell. Bei den dargestellten Modellen CMMI, RUP, V-Modell, V-Modell XT ist das RM innerhalb des SWE angesiedelt. Bei MSF bildet das RM die Schnittstelle zum Betrieb. In MOF, dem Referenzmodell für den Betrieb, gibt es dafür dann als Gegenstück das RM aus ITSM-Sicht. Das RM bei MOF entspricht im Wesentlichen dem RM in ITIL. Ebenso ist beim IT Service CMM das RM an ITIL orientiert. Sowohl bei MOF, als auch beim IT Service CMM und ITIL ist das RM ein Bestandteil innerhalb des ITSM.

Im Sinne des SWE wird das Release gleichgesetzt mit einer neuen Version eines Produktes. Das RM ist hier eine Aufgabe innerhalb des Projektmanagement zur Bereitstellung dieses Produktes innerhalb eines definier-

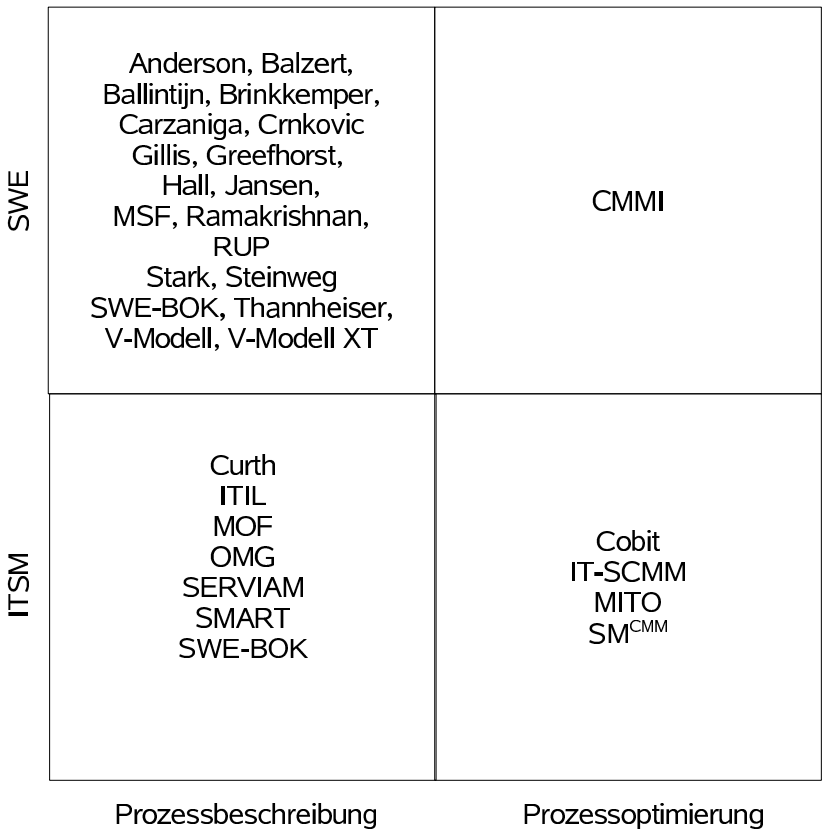


Abbildung 2.77: Übersicht Einordnung aller beschriebenen Modelle

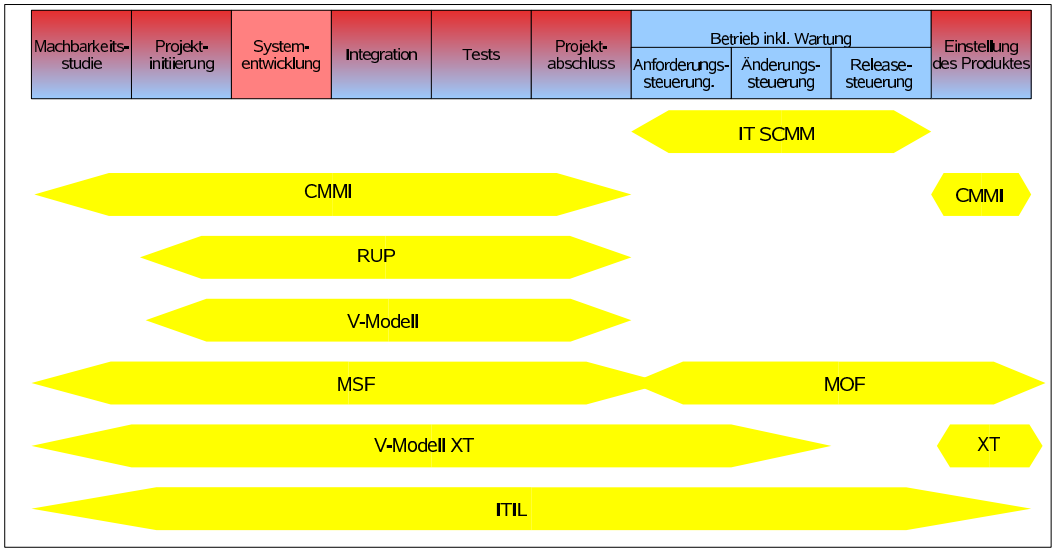


Abbildung 2.78: Übersicht Einordnung einer Auswahl der beschriebenen Referenzmodelle in das Lebenszyklusmodell

ten Zeitraumes und innerhalb eines vorherigen definierten Projektabschlusstermin. Das Aufgabengebiet des Release-Prozesses bezieht sich somit auf Aktivitäten, die mit der Verfügbarmachung dieses Releases für den Kunden im Zusammenhang stehen. Nicht einheitlich definiert sind jedoch die Grenzen dieses Prozesses. Bei verschiedenen Modellen erfolgt durch das RM noch die Einführung in den Produktivbetrieb beim Softwarebenutzer (vgl. z.B. V-Modell in Abschnitt 2.2.1). Technische Aspekte des Release-Prozesses, wie etwa die automatische Installation in einer Kundenumgebung wird auch häufig als Serviceleistung des Herstellers betrach-

tet. Mögliche Szenarien sind hier beispielsweise Push- oder Pull-Modelle, welche automatisch Neuerungen installieren (vgl. z.B. Abschnitt 2.2.6.6.2). Das Managen der Kommunikation zwischen Hersteller und Kunde wird bei dieser Serviceleistung adressiert (vgl. z.B. Abschnitt 2.2.6.2). Ein Vorschlag zur Verbesserung des Informationsaustausches sieht die automatische Rückmeldungen von Fehlermeldungen bei den verschiedenen Kunden vor, damit die Entwicklerorganisation die Software korrigieren kann (siehe z.B. Abschnitt 2.2.6.1).

RM-Prozesse des SWE können erfolgreich eingesetzt werden, wie das Beispiel des Change-Managements bei der Hausbank München zeigt [ZeMa 03]. Sie stellt für ihre Kunden einen Komplettservice für die Hausverwaltung zur Verfügung. Bei einer Überarbeitung ihres Entwicklungsprozesses hat sie das Change- und Release-Management auf der Grundlage von RUP implementiert. Dieser Ansatz ist vor allem dann erfolgversprechend, wenn die Softwareentwicklung vom eigenen Unternehmen gesteuert wird. Des Weiteren ist der entwicklungszentrierte RM-Ansatz einfacher, wenn wie im Beispiel der Hausbank München lediglich ein einziges (komponentenbasiertes) System entwickelt wird.

Nur wenige Beschreibungen verfolgen den Ansatz, das RM als eigenständigen Prozess innerhalb einer oft als Wartungsorganisation bezeichneten Einheit zu sehen. Hier steht das RM als Service für die Kunden zur Verfügung. Der Release-Prozess beginnt dann nach Fertigstellung der Software beim Hersteller und endet mit der Inbetriebnahme von Software beim Kunden. Dieser Ansatz ist im Kontext der Diplomarbeit relevant. Bei dieser Perspektive ist ein Release eine Bündelung von Änderungen, die gemeinsam in eine Produktivumgebung eingeführt werden (vgl. z.B. ITIL 2.3.4). Der Release-Prozess beinhaltet dann u.a. so genannte Logistik- oder Deployment-Aktivitäten, wie Softwareverteilung, -installation oder -aktivierung und Wartungsaktivitäten. Im Gegensatz zum SWE, wo das RM einen Projektcharakter hat, ist RM hier ein Prozess.

In Tabelle 2.3 werden Unterschiede zwischen der Anwendungsentwicklung und der Betreiberorganisation dargestellt, wie sie Thompson [Thom 05] identifiziert hat. Diese Unterschiede verdeutlichen, dass die Modelle des SWE nicht einfach bei IT-Serviceunternehmen eingesetzt werden können. Hier sind eigene Modelle nötig. Dennoch fehlt auch eine integrierte Sichtweise. Das wird vor allem beim RM sichtbar, welches sowohl aus SWE-Sicht als auch aus ITSM-Sicht die Schnittstelle zwischen Entwicklern, Betreibern und Anwendern bildet. Die genaue Prozessgrenze hängt dann vom jeweils betrachteten Referenzmodell bzw. zitierter Quelle ab. Das hat folglich auch Auswirkungen auf das Zusammenspiel bzw. das Verständnis der am Prozess beteiligten Rollen.

	Anwendungsentwicklung	**Anwendungssupport und -wartung**
Lebenszyklus	Linear	Zyklisch
Fokus	Anwendungen entsprechend Design, innerhalb Zeit und innerhalb Budget	reibungslos laufende Anwendungen
Benchmarks	*Deadlines*, Kosten	Antwortzeit, Zeit bis zur Lösung
Rolle in der Organisation	Zukunft	Gegenwart
Kultur	Visionär	Problemlöser
Führungskultur	*Aristocracy (Diktatorship)*	Demokratie
Managementparadigma	Projekte, Technologien	Anwendungen, Prozesse

Tabelle 2.3: Gegenüberstellung Anwendungsentwicklung vs. *Support & Maintenance* nach [Thom 05]

Brenner et. al. [BGN 06] fordern deshalb ein übergreifendes Modell. Sie beschreiben die Notwendigkeit hierfür anhand des *Requirements Engineering* (RE) . Die Schnittstellen des RE haben je nach Entwicklungsvorhaben unterschiedliche Ausprägungen. Bei Neuentwicklungen ist das RE eher vom SWE-Prozess gesteuert, bei Weiterentwicklungsprojekten eher vom ITSM-Prozess. Durch eine integrierte Sicht und ein formalisiertes übergreifendes Modell lassen sich so z.B. Synergien bei den zentralen Dokumenten Lastenheft und SLA erzielen, da zwischen beiden wechselseitige Abhängigkeiten bestehen.

Der Entwicklungstrend im ITSM ist gekennzeichnet durch Dienstleistungs-, Prozess- und Architekturorientierung und IT-Performance-Management [BöKr 05, HSW 04, Havo 04]. Durch Dienstleistungsorientierung wird die Bereitstellung von Informationssystemen als Leistung und Dienst am Kunden verstanden. Die Prozessorientierung basiert auf der Dienstorientierung, indem die Prozesse im Hinblick auf die Dienstleistungsvereinbarungen optimiert werden. Durch die Architekturorientierung wird diese Optimierung mithilfe von

gezielten Servicearchitekturen unterstützt. IT Performance Management oder IT-Governance stellt u.a. ein optimiertes Kostenmanagement, gezielte Investitionen und Transparenz bei der Verrechnung sicher. Auch ist es für das Überprüfen der IT-Architekturkonformität bzw. Erarbeiten von IT-Architekturvorgaben zuständig. Die Umstellung von einer technikorientierten zu einer serviceorientierten Organisation erfolgt dann mithilfe von ausgewählten Prozessreferenzmodellen, welche u.U. die Entwicklungstrends der Dienstleistungs-, Prozess- und Architekturorientierung und des IT Performance Managements bereits integrieren.

2.4.1 Release-Managementbegriffe mit unterschiedlicher Bedeutung

Aufgrund der beiden Sichtweisen auf das RM resultieren Begriffe, die je im Kontext SWE oder ITSM unterschiedliche Bedeutung haben. Nur der SWE-BOK Ansatz hat es sich zur Aufgabe gemacht, einen Konsens der Begriffe und Aufgaben über eine einheitliche Wissenbasis zu erreichen. Nachfolgende werden Begriffe gegenübergestellt, die im SWE- oder ITSM-Bereich verschieden verwendet werden.

2.4.1.1 Konfigurationsmanagement

Das Konfigurationsmanagement hat im Wesentlichen die Aufgabe der Verwaltung und Steuerung von Konfigurationseinheiten und ihren Beziehungen zueinander. Im Bereich des SWE sind die verwalteten Einheiten die Komponenten aus denen ein einzelnes Softwaresystem besteht.

Der Ansatz des ITSM ist dagegen umfassender. Das Konfigurationsmanagement verwaltet hier alle für die Bereitstellung von Diensten nötigen Einheiten. Das sind neben den einzelnen Softwaresystemen auch z.B. Hardware oder Netze. Unterstützt wird das Konfigurationsmanagement durch den Einsatz von Datenbanken (z.B. ISKB oder CMDB). Diese Datenbank hat meist eine logische Sicht auf das Modell der Infrastruktur und Services. Die entsprechenden Informationen bezieht sie aus anderen vorhandenen Datenquellen. Frühauf et. al. [FrLi 04] bezeichnen das Konfigurationsmanagement als Drehscheibe zwischen den vorhandenen Prozessen wie z.B. Change- oder Release-Management. Sie führen in ihrem Artikel Schwierigkeiten bei der Einführung des Konfigurationsmanagements vor allem darauf zurück, dass die Nutznießer daran immer nur andere sind. Deshalb vergleichen sie es mit Hygienevorschriften in einem Krankenhaus, die zwar nicht zur Gesundheit beitragen, aber dafür sorgen, dass ein Patient während der Behandlung nicht kranker wird. Analog sorgt das Konfigurationsmanagement für eine saubere Entwicklung oder einen sauberen Betrieb, indem jeweils mit den richtigen Einheiten gearbeitet bzw. die richtigen Konfigurationen manipuliert werden und eine Übersicht über alle Änderungen besteht.

2.4.1.2 Release-Planung

Die Release-Planung im SWE hat die Aufgabe, den Funktionalitätsumfang eines einzelnen Produktes festzulegen (Vergleich z.B. MSF Abschnitt 2.2.3). Im ITSM jedoch beinhaltet die Planung die Festlegung der Änderungen an verschiedenen Produkten, die nötig sind, um ein Release in der Produktivumgebung zu implementieren. Das kann wie in Abschnitt 2.3.3 bei SERVIAM zu einer unternehmensübergreifenden Planung führen, falls Produkte von Zulieferfirmen in das Release integriert werden.

2.4.1.3 Änderungsmanagement

Bei SWE ist die Aufgabe des Änderungsmanagement die Kontrolle und Steuerung von Änderungen an der Spezifikation eines einzeln zu erstellenden Produktes. Beim ITSM liefert das Änderungsmanagement standardisierte Methoden und Prozeduren zur Behandlung von Änderungen an allen Konfigurationseinheiten, die unter der Verwaltung des ITSM-Konfigurationsmanagements sind.

2.4.2 Release-Management im Kontext der Diplomarbeit

Im Kontext der Diplomarbeit ist die ITSM-Sicht besonders relevant. Die HVBInfo ist ein Dienstleister für die HVBGroup und betreibt deren IT-Infrastruktur. Bei der HVBInfo findet selbst keine nennenswerte Softwareentwicklung statt. Die Software wird von anderen Firmen geliefert. Ein wichtiger Zulieferer in diesem Zusammenhang ist die HVBSystems. Sie ist ebenfalls eine Tochterfirma der HVBGroup und arbeitet eng zusammen mit der HVBInfo.

Die HVBSystems setzt das selbst definierte SWE-Modell *Application Development Information Navigator* (ALADIN) ein. ALADIN ist ebenfalls der Name des Dokumentationssystems mit dem Entwicklungsprojekte dokumentiert werden. Das Aladinvorgehensmodell der HVBSystems wurde hauptsächlich auf der Basis von RUP entwickelt. Weitere Einflüsse hat das V-Modell geliefert. Die HVBSystems bewertet die eigenen Prozesse auf der Basis von *Software Process Improvement Capability Determination* (SPICE) . SPICE ist die Norm ISO/IEC TR 15504. Das ist ein Modell zur Bewertung des Reifegrades von Prozessen in IT-Unternehmen mit der Einstufung in fünf Reifegradstufen ähnlich dem CMM Modell (vgl. CMMI in Abschnitt 2.2.5) [Stie 99] . Die HVBSystems plant für das Jahr 2006 die Erhöhung des SPICE Levels von zwei auf drei.

Obwohl das RM im Kontext von ITSM steht, ist die Betrachtung der SWE-Perspektive dennoch von Bedeutung. Zum einen wird die Einordnung der zu analysierenden Prozesse dadurch einfacher. Zum anderen hilft es bei der Abgrenzung sowohl von Prozessen, als auch von Begriffen.

Von den vorgestellten Modellen im ITSM Bereich kommt ITIL die größte Bedeutung zu. Das SERVIAM Framework ist noch im Status der Entwicklung und hat den Fokus ausschließlich auf *Webservices*. SMART hat im Praxisumfeld keine Relevanz. Die Verwendung von MOF ist eher sinnvoll, wenn hauptsächlich Microsoftprodukte eingesetzt werden und wenn auch die Entwicklerorganisationen MSF einsetzten. Dadurch wäre ein lückenloser Prozessübergang möglich.

Die HVBInfo selbst hat seit 1999 ihre Prozesse an ITIL ausgerichtet. ITIL hat weiterhin den Vorteil, dass es sich mittlerweile zu einem internationalen De-facto-Standard entwickelt hat und öffentlich zugänglich ist [HoHu 03, ViG 04, Somm 04, PuQu 03]. Im Rahmen des OGC ITIL Refresh Projektes wird ITIL laufend weiterentwickelt. Eine Teilnehmergruppe an dieser Weiterentwicklung ist das IT Service Management Forum (itSMF) . Aufgrund dessen Mitgliederanzahl von über 1000 Mitgliedsfirmen ist eine breite Wissensbasis vorhanden [ITIL04, ITIL05]. MOF hingegen ist ein propriäteres Modell von Microsoft. Es basiert wiederum auf ITIL. Sommer empfiehlt den Einsatz von MOF, wenn Microsoft-Technologien als Schlüsselelemente der IT-Infrastruktur im Einsatz sind [Somm 04, S.45]. Das ist bei der HVBInfo allerdings nicht der Fall. Neben Microsoft-Technologien werden hier auch andere Technologien (UNIX, Linux, IBM) verwendet. Des Weiteren hat ITIL in der Praxis einen hohen Bekanntheits- und Nutzungsgrad. Laut einer in der Computer Zeitung veröffentlichten Statistik ist ITIL mit 34% das am meisten eingesetzte Referenzmodell bei Firmen im deutschsprachigen Raum [Schn 06]. In einer von Kemper et. al. [KHP 04] durchgeführten Studie haben zwei Drittel der Befragten angegeben, dass ihnen ITIL bekannt wäre und von diesem Personenkreis haben 57% bereits ITIL eingeführt. Von den ITIL-Implementierungen wurde die Mehrzahl als Erfolg gesehen, vor allem in Unternehmen mit mehr als 50 Mitarbeitern. Durch Einführung von ITIL können laut den Umfrageergebnissen Verbesserungen im Bereich der Transparenz der IT-Prozesse, der Qualität und den Reaktionszeiten erzielt werden. Diese Vorteile wurden auch in der Umfrage von Hochstein et. al. [HTB 05] aufgeführt. Köhler [Kö 05] und Victor et. al. [ViG 04] haben noch folgende weitere Vorteile identifiziert, die durch den Einsatz von ITIL zum Tragen kommen:

- ITIL beschreibt die für erfolgreiches ITSM notwendigen Prozesse
- ITIL beschreibt Prozesse, lässt aber Raum für die eigene Implementation dieser Prozesse
- ITIL basiert auf langjähriger Erfahrung von Nutzern aus verschiedenen Branchen
- ITIL liefert eine gemeinsame Sprache zur Unterstützung des gemeinsamen Verständnisses
- durch ITIL ist eine effizientere und genauere Ausrichtung der IT auf die Unternehmensziele möglich; das erhöht ihr *Standing*
- durch die Kommunikation über definierte Ansprechpartner verbessern sich Arbeitsabläufe und das Vertrauensverhältnis der IT-Kunden

- ITIL erhöht die Reaktionsgeschwindigkeit auf die Wünsche der Kunden und damit die Wettbewerbsfähigkeit

Das Bundesamt für Sicherheit in der Informatik (BSI) [BSI05] führt positive Synergieeffekte in ökonomischer und sicherheitstechnischer Hinsicht bei einer Integration von ITSM und Sicherheitsmanagement auf. Durch ITIL können sicherheitstechnische Anforderungen transparent gemacht werden und das Sicherheitsmanagement kann produktiv in Entscheidungsprozesse mit eingebunden werden. Die Maßnahmen des IT-Grundschutzhandbuches sind im ITIL-sinne Vorschläge für einen Sicherheitsservice (*Service Level*).

Allerdings löst ITIL nicht alle Probleme. Als Nachteil von ITIL gilt, dass nicht konkret beschrieben wird, wie die Prozesse eingeführt werden sollen. Es wird auch nicht dargestellt, wie die IT-Prozesse implementiert werden können [Kö 05]. Des Weiteren können Probleme durch den Widerstand der Mitarbeiter gegen neue Prozesse, durch nicht ausreichendes Wissen des Release-Inhaltes oder durch fehlende Bereitschaft gescheiterte Releases zurückzuziehen [Macf 02] auftreten. Hochstein et. al. [HoHu 03] kritisieren, dass ITIL aufgrund von Schwächen bei formalen Anforderungen kein Referenzprozessmodell im wissenschaftlichen Sinne darstellt.

Trotzdem kann mit Einführung der ITIL Prozesse eine gute Grundlage für das gesamte IT-Management gelegt werden. Hochstein et. al. [HHB 05, ZBP 05] haben die Überdeckung der ITIL Prozesse mit ihrem serviceorientierten IT-Management (SOITM) Modell untersucht. Ihr Modell basiert auf dem *Supply-Chain Operation Reference* (SCOR) -Modell des Supply-Chain-Council. Abbildung 2.79 zeigt, dass die ITIL Service Support- und Delivery-Prozesse einen Teil der planerischen und steuernden Entwicklungs-, Produktions- und Kundenmanagementbereiche abdecken. Lücken sehen sie hauptsächlich im Lieferanten- und Portfoliomanagement und den strategischen Bereichen des Entwicklungs-, Produktions- und Kundenmanagement.

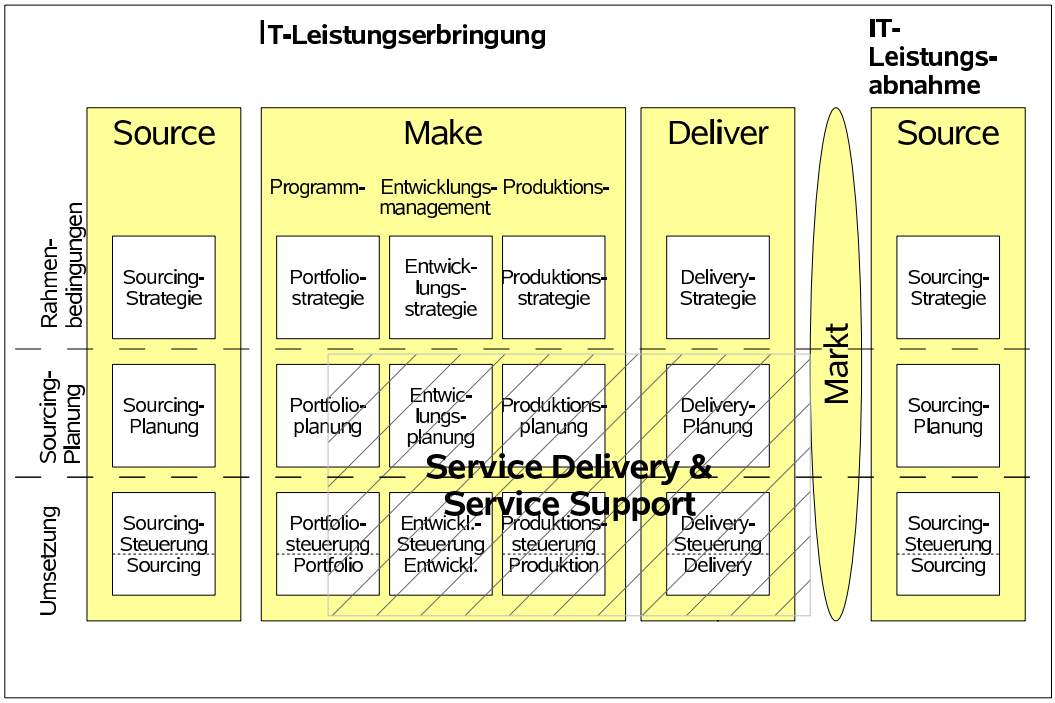


Abbildung 2.79: ITIL Service Support und Delivery im Vergleich zu SOITM nach [HHB 05]

Dennoch ist durch die Anwendung der *Best Practices* aus ITIL auch die Grundlage für eine spätere Zertifizierung nach BS15000 bzw. dem kommenden Standard ISO 20000 gelegt [BS15000]. Des Weiteren setzten die vorgestellten Reifegradmodelle MITO, SM^{CMM} und IT-SCMM ITIL als Basis der Bewertung voraus. Bei diesen Reifegradmodellen ist im Moment noch nicht absehbar, welches sich in Zukunft als Standard durchsetzten wird. MITO ist ein Modell aus der Schweiz. Das IT-SCMM hat die offizielle Erlaubnis von SEI erhalten, den Begriff „CMM“ für ihr Modell zu benutzen. Langfristig ist hier die Integration mit dem CMMI vorgesehen. Das SM^{CMM} wird ebenfalls als Ergänzung zum CMMI entwickelt. Begrüßenswert wäre hier ein einziges (gemeinsames) Reifegradmodell.

Der Ansatz von CobIT kann ebenfalls als komplementär zu ITIL gesehen werden. CobIT ist ein internationaler De-facto-Standard im Bereich der IT-Governance. Die Aufgaben von IT-Governance umfassen laut Grohmann [Groh 03] u.a. folgende Bereiche:

- Aussehen von IT-Architekturen
- Umfang und Art von IT-Prinzipien und IT-Policies
- Gestaltung der IT-Infrastruktur
- Festlegung der Entscheidungsträger zu IT-Investitionen und Prioritäten
- Verrechnung der IT-Kosten
- Festlegungen zur Messung der Effektivität der *IT-Governance*

ITIL bietet dann die Basis für die Umsetzung der Referenzprozesse und CobIT wird als Kontroll- und Prüfmethode verwendet [HoHu 03, Alc05, HoCo 02, ITGI-OGC, ZBP 05]. So kann die gemeinsame Verwendung von ITIL und CobIT die operativen und strategischen Bereiche des *Service Support* abdecken.

3 Beschreibung des derzeitigen Release-Managements bei der HVBInfo

Inhaltsverzeichnis

3.1 Die HVB Informations-Verarbeitungs-GmbH und ihr Umfeld 66
3.2 Release-Management bei der HVBInfo 67
3.2.1 Plattformen bei der HVBInfo 68
3.2.2 Begriffe im Release-Management bei der HVBInfo 69
3.2.2.1 Anwendungssysteme und Teilsysteme 69
3.2.2.2 Zuordnung der Teilsysteme zu den Plattformen 70
3.2.2.3 Release-Begriffe bei der HVBInfo 71
3.2.2.4 Kundenset 73
3.2.2.5 Umgebungen 73
3.3 Motivation für die Integration der vorhandenen Release-Prozesse 75
3.4 Release-Management im dezentralen Bereich 77
3.4.1 Beteiligte Rollen und ihre Aufgaben 77
3.4.2 Release-Aktivitäten im dezentralen Bereich 79
3.4.2.1 Festlegen der Releasepolicy und des Release-Plans 79
3.4.2.2 Entwurf, Releasezusammenstellung und -Konfiguration 85
3.4.2.3 Testen und Abnahme des Releases 86
3.4.2.4 Planung des Einsatzes 87
3.4.2.5 Kommunikation, Vorbereitung und Schulung 87
3.4.2.6 Release-Verteilung und Installation 88
3.4.3 Beschreibung des Release-Steuerungsprozesses 88
3.4.3.1 Release eröffnen 90
3.4.3.2 Produkt- und Release-Status überwachen 91
3.4.3.3 Change eröffnen 96
3.4.4 Bewertung des Release-Durchlaufes 97
3.4.5 Einordnung in das Lebenszyklusmodell 98
3.5 Release-Management bei zentralen Servern 99
3.5.1 Beteiligte Rollen und ihre Aufgaben 99
3.5.2 Release-Aktivitäten im Bereich zentraler Server 102
3.5.2.1 Festlegen der Releasepolicy und des Release-Plans 102
3.5.2.2 Entwurf, Release-Zusammenstellung und -Konfiguration 106
3.5.2.3 Testen und Abnahme des Releases 106
3.5.2.4 Planung des Einsatzes 108
3.5.2.5 Kommunikation, Vorbereitung und Schulung 109
3.5.2.6 Release-Verteilung und Installation 109
3.5.3 Beschreibung des Release-Steuerungsprozesses 109
3.5.4 Bewertung des Release-Durchlaufes 111
3.5.5 Einordnung in das Lebenszyklusmodell 112
3.6 Release-Management im Host-Bereich 113
3.6.1 Beteiligte Rollen und ihre Aufgaben 113
3.6.2 Release-Aktivitäten im Host-Bereich 114
3.6.2.1 Festlegen der Releasepolicy und des Release-Plans 114

3.6.2.2 Entwurf, Release-Zusammenstellung und -Konfiguration 119
3.6.2.3 Testen und Abnahme des Releases 119
3.6.2.4 Planung des Einsatzes 120
3.6.2.5 Kommunikation, Vorbereitung und Schulung 120
3.6.2.6 Release-Verteilung und Installation 120
3.6.3 Beschreibung des Release-Steuerungsprozesses 121
3.6.3.1 Release eröffnen 122
3.6.3.2 Produkt- und Release-Status überwachen 123
3.6.3.3 Change eröffnen 124
3.6.4 Bewertung des Release-Durchlaufes 125
3.6.5 Einordnung in das Lebenszyklusmodell 125
3.7 Zusammenfassung 126

In diesem Kapitel wird ab Abschnitt 3.1 die HVBInfo und ihr Umfeld beschrieben. Der Fokus des Release-Managements bei der HVBInfo ist in Abschnitt 3.2 dargestellt. Für das leichtere Verständnis des ab Abschnitt 3.4 beschriebenen plattformspezifischen Release-Managements werden in Abschnitt 3.2.2 die releaserelevanten Begriffe der HVBInfo erklärt. In Abschnitt 3.3 werden die Motive für die Integration der bestehenden Verfahren erläutert.

Die jeweilige Beschreibung des Release-Managements im dezentralen Bereich (Abschnitt 3.4), Bereich zentraler Server (Abschnitt 3.5) und im Host-Bereich (Abschnitt 3.6) enthält jeweils die am Prozess beteiligten Rollen mit ihren Aufgaben und die Release-Aktivitäten. Die Gliederung dieser Beschreibungen orientiert sich an ITIL, auch wenn es aufgrund der SWE-Einordnung hier oft keine entsprechenden Punkte gibt. Hierdurch lassen sich Abweichungen von ITIL und damit Anforderungen an den neuen Prozess leichter feststellen. Da speziell im Bereich zentraler Server der Prozess aus der Softwareentwicklung gesteuert wird, wird er hier Produktionsübergabeprozess anstatt Release-Prozess genannt.

3.1 Die HVB Informations-Verarbeitungs-GmbH und ihr Umfeld

Die HVB Informations-Verarbeitungs-GmbH (HVBInfo) wurde 1991 gegründet und ist eine hundertprozentige Tochter der HypoVereinsbank AG. Die HVBInfo betreibt mit ca. 700 Mitarbeitern an den Standorten München, Hamburg und London die IT-Infrastruktur ihrer Kunden. Die Kunden sind Unternehmen der HVB Group wie etwa HVBSystems, HVBPayments&Service GmbH, HVBImmobilien AG oder die HypoVereinsbank AG.

Betrieben werden ca. 3400 Server und 30000 Clients. Der größte Kunde ist die HypoVereinsbank AG. Mit ihr werden täglich ca. 13 Millionen Online-Transaktionen (IMS/DB2) abgewickelt. Im Einsatz befinden sich hierfür über 2000 Server.

Für die Erstellung des Dienstleistungsangebots der HVBInfo ist der Kundenservice bzw. das Service Level Management zuständig. Die angebotenen Services werden im Servicekatalog veröffentlicht und enthalten folgende Dienste:

HYPer-Full Service, Arbeitsplatz und Arbeitsplatz-Services: Der so genannte *HYPerPC* ist ein standardisierter IT-Arbeitsplatz. Der Service umfasst die Hardware, d.h. PC mit Monitor, Tastatur und Maus und dezentrale und zentrale Infrastruktur-Server. Dienstleistungen im Bereich der Arbeitsplatz-Services beinhalten Bereitstellung, Beschaffung, Installationsmanagement, Betrieb und Überwachung von IT-Arbeitsplätzen inklusive Peripheriegeräten wie z.B. Drucker. Für Anfragen steht den Kunden eine Supportunterstützung durch die *Serviceline* zur Verfügung. Die Serviceline nimmt Störungen und Anfragen der Anwender entgegen. Sie bearbeitet Aufträge zu technischen Störungen und fachlichen Anfragen zu Applikationen. Unterstützt wird sie hierbei durch den *2nd Level Support*. Dieser ist zweigeteilt in den Applikationssuport für inhaltliche Fragen zu Anwendungen, die nicht direkt von der Serviceline beantwortet werden können und in einen technischen Support zur Bearbeitung von technischen Störungen. Die Serviceline ist die zentrale Anlaufstelle für die Anwender. Das *Release-Management* stellt

für diese IT-Arbeitsplätze die Software für die Endgeräte und für die Standort-Server bereit. Die Arbeitsplatzservices beinhalten auch Internet-Services. Im Rahmen von Internet-Services verwaltet das Internet-Domain-Management Domain-Namen, betreibt die entsprechenden Systeme und überwacht interne und externe DNS-Server. Für Internetanwendungen stellt es Sicherheitssysteme bereit. Weitere Internet-Services beinhalten den Betrieb von Standarddiensten wie http, Mailservices oder Webhosting-Lösungen, die Überwachung, Messung und das Reporting von der End-to-End Verfügbarkeit von Web-Anwendungen oder das User-Management für verschiedene Systeme (NT, XP, Unix, Host, SAP).

Infrastrukturservices: Infrastrukturservices sind verantwortlich für den Host-Betrieb inkl. Administration, 1^{st} und 2^{nd}-Level Support, Überwachung, Störungsbearbeitung, Server-Betrieb, Speicherlösungen und für Netzanbindungen.

Lizenzen: Das Lizenzmanagement verwaltet und verrechnet Lizenzen für fremdentwickelte Software.

Output-Services: Im Rahmen des Output-Managements werden Output auf Papier oder CDs erstellt und Formulare online bereitgestellt.

Selbstbedienungs-Services: Die Selbstbedienungs-Services (SB-Services) betreiben und überwachen Selbstbedienungsgeräte der Kunden wie etwa Geldautomaten, Kontoauszugsdrucker oder Selbstbedienungsterminals. Diese Dienste werden ebenfalls durch die Serviceline und den Applikationssupport unterstützt. Das *Release-Management* stellt die entsprechende Software für die entsprechenden SB-Geräte und Standort-Server in Form von Releases bereit.

Dienstleistungen: Im Rahmen der allgemeinen Dienstleistungen ist der Support, sprich Serviceline und Applikationssupport eingeordnet. Zu den weiteren Dienstleistungen gehört das Beschaffungs- und Installations-Management, das *Release-Management* und das System Management für das Monitoring, die Softwareverteilung und die Inventarisierung.

Consulting: Ein weiterer Service der HVBInfo ist das Consulting. Es bietet den Kunden die Dokumentation und Beratung hinsichtlich Optimierungsmöglichkeiten von Geschäftsprozessen an.

3.2 Release-Management bei der HVBInfo

Die HVBInfo ist ein reiner IT-Betreiber ohne nennenswerte eigene Softwareeigenentwicklung. Abbildung 3.1 zeigt die Schnittstellen zu den Kunden und zu zuliefernden Softwareentwicklern.

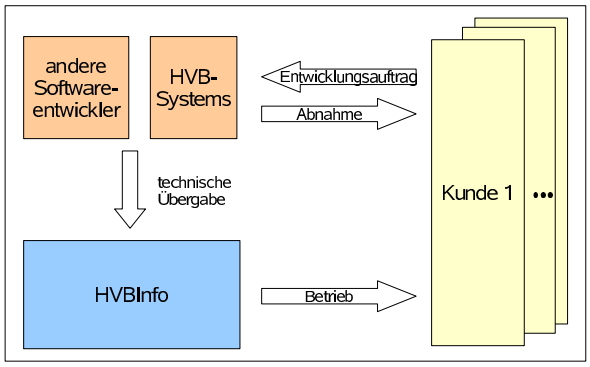


Abbildung 3.1: Schnittstellen zwischen HVBInfo, Entwicklungsorganisationen und Kunden

Die jeweiligen Kunden beauftragen hier die Softwareentwickler mit der Erstellung neuer oder Änderung bestehender Software. Die HVBInfo als Betreiber erhält über die technische Übergabe diese Software, um sie in die bestehende IT-Infrastruktur einzuführen und zu betreiben. Die jeweilige Abnahme der Software erfolgt am Ende der Softwareentwicklung durch den Kunden.

Ein wichtiger Softwareentwickler in diesem Zusammenhang ist die HVBSystems. Sie ist ebenfalls eine Tochterfirma der HVBGroup. Die Zusammenarbeit bei der Einführung von Softwareänderungen bei der HVBInfo ist sehr eng. Bei der Einführung von Softwareänderungen anderer Firmen werden die für den jeweiligen Prozess benötigten Rollen daher von Mitarbeitern der HVBSystems oder der HVBInfo wahrgenommen. Es wird im Folgenden daher nicht mehr auf diese anderen Firmen eingegangen.

Zarnekow et. al. [ZBP 05] beschreiben diese Art der Organisationsform als traditionellen Ansatz. Die Kunden schließen hierbei Verträge mit der Entwicklungs- und der Betreiberorganisation ab. Ein zitierter Manager bei Zarnekow sagt dazu: „Die Software wird entwickelt und dann über die Mauer in die Produktion geworfen„ [ZBP 05, S. 38]. Anstelle von „Mauer“ hat sich bei der HVBInfo der Begriff *Theke* etabliert. Die Theke stellt

eine virtuelle Schnittstelle zwischen den zuliefernden Softwareentwicklern und der HVBInfo dar. Die an der Theke übernommene Software wird durch den Release-Prozess in den Produktionsbetrieb gebracht. Diese Art des Ablaufes der Beauftragung und Einführung von Softwareänderungen entspricht der SWE-Sicht. Analog wie ab Abschnitt 2.2 beschrieben, begleiten die jeweiligen Softwareentwickler die Einführung ihrer Software beim Kunden. Im Falle der Organisation bei der HVBGroup wird die Software anstatt direkt beim Kunden bei der HVBInfo als Betreiber eingeführt.

Obwohl die HVBInfo eine reine Betreiberorganisation ist und die internen Prozesse an ITIL orientiert sind, ist das derzeitige RM bei der HVBInfo in die vorher vorgestellte SWE-Sicht einzuordnen. Unter RM wird bei der HVBInfo der Prozess der gemeinsamen Abnahme- und Einführung und Verteilung von Softwareprodukten in eine Kundenumgebung verstanden. Der Prozess beinhaltet daher Komponenten des Abnahme- und Einführungsprozesses, wie er bei Balzert oder Steinweg (siehe Abschnitt 2.2.6.8) beschrieben wurde und Komponenten des Deployment-Prozesses, wie in Abschnitt 2.2.6.6 beschrieben.

Aus diesem Grund treten folgende Aktivitäten bei der HVBInfo während des Release-Prozesses auf:

- Übernahme der von anderen Organisationen entwickelten Software
- Bereitstellen der entsprechenden Testinfrastruktur
- Organisation und Koordination von Testaktivitäten
- Anpassungen der Software an die lokalen Gegebenheiten, d.h. analog lokales *Customizing* (vgl. Einführungsphase Steinweg Abschnitt 2.2.6.8.2)
- Schulung des Personals und der Benutzer
- Koordination und Organisation des Pilotbetriebes
- Koordination der Softwareverteilung
- Koordination des Roll-outs

Die Verantwortung dieser Tätigkeiten obliegt nicht immer dem Release-Management. Teilweise koordiniert und organisiert das RM lediglich diese Tätigkeiten. Die Ursache für die Einordnung des RM der HVBInfo in die SWE-Sicht liegt in der historischen Organisation der Prozesse. Die HVBInfo hat im Jahr 1999 ihre Prozesse nach ITIL umgestellt. Allerdings wurde zu diesem Zeitpunkt das gesamte RM noch von der HVBSystems betrieben. Das RM im dezentralen Bereich wurde im August 2003 von der HVBSystems zur HVBInfo verlagert. Die dort bestehenden Prozesse wurden dann in die bestehende Prozesslandschaft integriert. Das RM im Bereich zS erfolgt weiterhin als begleitete Abnahme- und Einführungsphase durch die Softwareentwickler der HVBSystems. Lediglich für die Änderungen am Betriebssystem und betriebssystemnaher Software ist momentan ein durch die HVBInfo gesteuertes RM in Einführung. Ebenso werden im Host-Bereich die meisten Einführungen von Softwareänderungen durch die Softwareentwickler gesteuert. Durch ein Projekt werden derzeit vier Anwendungssysteme pilothaft über einen durch das Release-Management gesteuerten Prozess in die Produktivumgebung eingeführt.

Derzeit momentanen Release-Verfahren unterscheiden sich durch die Plattform, auf der sie angewandt werden, den jeweiligen Ablauf und ob sie vom Release-Manager oder vom jeweiligen Softwareentwicklungsverantwortlichen in die Produktivumgebung gehandhabt werden. In den folgenden Abschnitten werden die betrachteten Plattformen, verschiedenen releaserelevanten Begriffe der HVBInfo und die Motive für die Integration dieser verschiedenen Release-Prozesse zu einem einzigen Prozess vorgestellt. Ab Abschnitt 3.4 werden diese plattformspezifischen Release-Verfahren erläutert.

3.2.1 Plattformen bei der HVBInfo

Die Aufgabe dieser Diplomarbeit ist die Integration des bei der HVBInfo bestehenden Release-Managements. Es gibt allerdings derzeit bei der HVBInfo drei verschiedene Bereiche im RM (Beschreibung des bestehenden RM siehe ab Abschnitt 3.4). Diese Bereiche sind technologiespezifisch aufgeteilt. Diese Aufteilung betrifft die drei Technologiebereiche bzw Plattformen Client/Server (C/S) , zentrale Server (zS) und Host.Die Klassifikation dieser Plattformen wird nachfolgend dargestellt. Nicht betrachtet wird der Release-Prozess für Hardware,

Netz, Hintergrundspeicher/Storage und Telekommunikation. Die für die Aufgabe der Diplomarbeit berücksichtigten Plattformen klassifizieren sich wie folgt:

Host: Mainframes von IBM der zSeries.

zentrale Server (zS): Zentrale Server werden direkt von der HVBInfo betrieben und verwaltet. Sie stehen auch räumlich in den Gebäuden der HVBInfo. Es sind derzeit ca. 557 Windows Server mit den Betriebssystemen Windows 2000, Windows 2003 und WinNT und ca. 1.194 Nodes im UNIX Bereich (Solaris, AIX) und Linux Server im Einsatz.

Client/Server bzw. dezentrale Server (C/S): Der Bereich Client/Server bezieht sich auf Server und Clients, die bei den Kunden der HVBInfo im Einsatz sind. Hier werden ca. 1.290 Server und 30.000 Clients betreut.

Für das jeweilige RM dieser drei verschiedenen Plattformen haben sich verschiedene releaserelevante Begriffe etabliert. Diese werden für ein leichteres Verständnis des ab Abschnitt 3.4 beschriebenen Status quo im RM nachfolgend eingeführt.

3.2.2 Begriffe im Release-Management bei der HVBInfo

In diesem Abschnitt werden die bei der HVBInfo im RM gebräuchlichen Begriffe dargestellt. Diese Begriffe werden zum besseren Verständnis der eigentlichen Beschreibung des bestehenden RM vorweggenommen. Die Release-spezifischen Begriffe betreffen die Benennung und Aufteilung von Softwareeinheiten (siehe Abschnitt 3.2.2.1), die Klassifikation verschiedener Release-Arten (siehe Abschnitt 3.2.2.3) und Bezeichnungen für spezielle (Test-)Umgebungen (siehe Abschnitt 3.2.2.5).

3.2.2.1 Anwendungssysteme und Teilsysteme

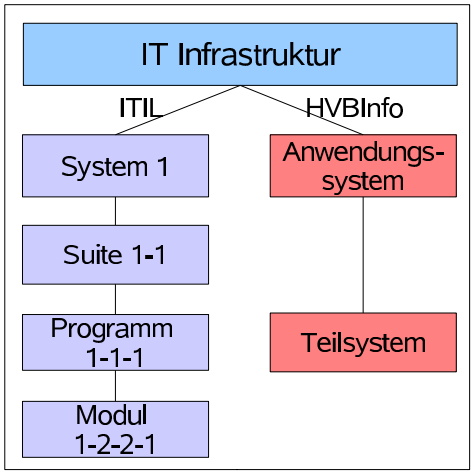


Abbildung 3.2: Gegenüberstellung Release-Einheit ITIL und HVBInfo

Abbildung 3.2 zeigt die Gegenüberstellung der ITIL Begriffe zu den Software-Release-Einheiten (vgl. Abschnitt 2.3.4.2.2) und den entsprechenden Begriffen bei der HVBInfo. Anstelle von System in ITIL wird bei der HVBInfo von Anwendungssystem (AWS) gesprochen. Laut der internen Definition ist ein AWS eine Software oder auch ein Softwarepaket für ein geschlossenes, logisch und technisch abgrenzbares Arbeitsgebiet. Für den Suite-Begriff von ITIL gibt es bei der HVBInfo keine Entsprechung. Programme in ITIL werden bei der HVBInfo Teilsysteme oder ebenfalls Programme genannt. Für die weitere Unterteilung der Programme sind die Begriffe Modul, Objekt oder Komponente in Gebrauch. Sie sind in der Graphik aus Übersichtsgründen nicht dargestellt.

3.2.2.1.1 Repository Die AWS werden im *Repository* der HVBSystems verwaltet. Das *Repository* ist ein Informationssystem mit Metadaten zu jedem Anwendungssystem und Teilsystem. Die weitere Unterteilung der Teilsysteme ist im *Repository* nicht abgebildet. Die HVBSystems stellt der HVBInfo und damit auch dem RM einen Zugriff auf die Einträge im *Repository* zur Verfügung. Neben Eigenentwicklungen werden dort auch die Informationen von zugekaufter Fremdsoftware eingetragen. Die HVBSystems selbst betreut derzeit ca. 30.000 HOST-Programme mit insgesamt ca. 43 Mio. Lines of Code, 690 Anwendungssysteme mit ca. 2.000 Teilsystemen (850 HOST, 175 Java, 850 WinNT, 150 UNIX, 10 SAP). Zu jedem AWS werden u.a. Name des AWS, System-Owner (=Geschäftsfeld/Konzerneinheit), verantwortlicher Mitarbeiter, Risikoklasse (siehe 3.11 auf Seite116) und zugehörige Teilsysteme gepflegt.

Der Eintrag eines Systems bzw. seiner Untereinheiten im *Repository* der HVBSystems ist seine Voraussetzung, um im dezentralen Release-Prozess und im Host-Release-Prozess der HVBInfo gehandhabt werden zu können.

Hier werden relevante Einträge vom dort eingesetzten (Release-)Tool übernommen. Für Software, die im zS Release-Prozess gehandhabt wird, gilt diese Voraussetzung nicht.

3.2.2.1.2 Teilsysteme Ein AWS kann sich wiederum aus mehreren Teilsystemen zusammensetzten. Die Teilsysteme unterscheiden sich nach Plattform und nach Eigen- oder Fremdentwicklung. Folgende Teilsystemtypen werden im *Repository* unterschieden:

Teilsystem Komponentenverwaltungsystem: Komponentenverwaltungsystem (KVS) -Teilsysteme sind Host-Systeme; sie werden eigentlich im KVS verwaltet und gepflegt; relevante Metadaten werden per *Batchjob* in das *Repository* übertragen; KVS selbst verwaltet die AWS und beinhaltet auch automatische Deployment-Aktivitäten zur Übertragung der Host-Softwarekomponenten in die Produktion; anstelle des Begriffs KVS-Teilsystem wird im Host-Umfeld auch der Begriff Komponente oder Objekt verwendet

Teilsystem HOST und Teilsystem HOST-FSW: Teilsysteme Host sind Eigenentwicklungen, welche nicht im KVS abgelegt werden; ebenso sind Fremdsoftware-Teilsysteme (FSW) der Host-Plattform, welche nicht im KVS abgelegt sind; Diese beiden werden nicht für die Aufgabe in der Diplomarbeit betrachtet

Teilsystem NT/XP: Teilsysteme mit Windows NT bzw. XP als Plattform

Teilsystem NT/XP-FSW: Fremdentwickelte Windows NT bzw. XP Teilsysteme

Teilsystem UNIX: eigenentwickelte Teilsysteme der UNIX-Plattform

Teilsystem UNIX-FSW: fremdentwickelte Teilsysteme der UNIX-Plattform

Teilsystem Java/WEB und Java/WEB-FSW: plattformunabhängige Javakomponenten und fremdentwickelte Javakomponenten

Teilsysteme im SAP Bereich (SAP-Add On, SAP-Add On-FSW, SAP-Modul-FSW): Diese Teilsysteme sind für die Aufgabe der Diplomarbeit nicht relevant.

In dieser Diplomarbeit soll das derzeit bei der HVBInfo bestehende RM integriert werden. Die in obiger Beschreibung als nicht betrachtet gekennzeichneten Teilsysteme werden vom momentan bestehenden RM nicht gehandhabt und sind deshalb für diese Aufgabenstellung auch nicht relevant.

3.2.2.2 Zuordnung der Teilsysteme zu den Plattformen

Die obigen Teilsysteme werden über die verschiedenen plattformspezifischen Release-Prozesse gehandhabt. Abbildung 3.3 zeigt die Zuordnung der Teilsysteme zu den Plattformen.

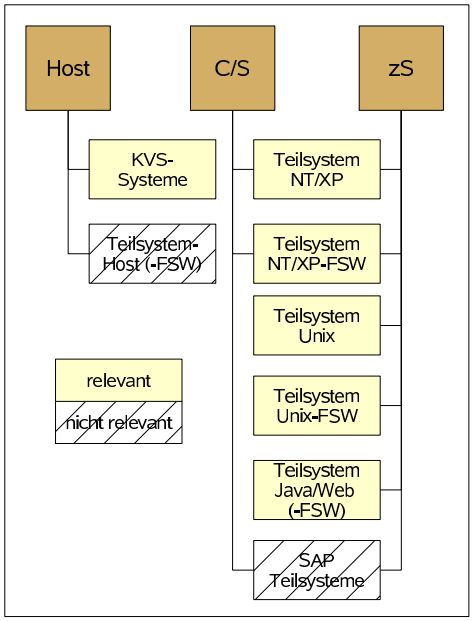


Abbildung 3.3: Zuordnung der Teilsysteme zu den Plattformen

Das ist auch nachfolgend beschrieben.

3.2.2.2.1 Dezentraler Bereich Der dezentrale Release-Prozess ist zuständig für manche Teilsysteme NT/XP und NT/XP-FSW. Dieser Eintrag im *Repository* ist für die werkzeugunterstützte Zuordnung eines Teilsystems zu einem Release nötig. Hierfür ist mindestens ein Eintrag der so genannten Rolle *Stargate-PV* in den Attributen des Teilsystems nötig. Stargate ist ein Werkzeug zur Unterstützung des dezentralen Release-Prozesses (siehe Abschnitt 3.4.2.1.1). Der dezentrale Release-Prozess behandelt nur diejenigen Teilsysteme, in denen die Rolle Stargate-PV ausgefüllt ist. Anstelle des Begriffs Teilsystem wird im dezentralen Bereich auch von Produkten gesprochen.

3.2.2.2.2 Bereich zentrale Server Die Teilsysteme UNIX, NT/XP und Java/Web sind mögliche Kandidaten für den Bereich zS. Allerdings erfolgt die Pflege der Teilsysteme nach der eingesetzten Technologie und nicht nach der Plattform, auf der sie zum Schluß laufen. So kann z.B. ein Teilsystem Java/Web ein System sein, welches einen Host als Server verwendet. Eine allgemeingültige Zuordnung der Teilsysteme zu der Plattform ist hier nicht möglich. Aus diesem Grund wird im Bereich zS das *Repository* lediglich für das Beziehen weniger Informationen verwendet, z.B. zur Bestimmung der verantwortlichen Mitarbeiter als Ansprechpartner,

3.2.2.2.3 Bereich Host Die Teilsysteme HOST, KVS und Teilsystem HOST-FSW sind Teilsysteme der Plattform Host. Für die Aufgabe in der Diplomarbeit sind allerdings lediglich KVS-Teilsysteme relevant.

3.2.2.3 Release-Begriffe bei der HVBInfo

Generell ist ein *Release* bei der HVBInfo im ITIL-Sinne ein Bündel von Release-Einheiten, das gleichzeitig abgenommen und ausgerollt wird. Tabelle 3.1 zeigt eine Übersicht der in der HVBInfo verwendeten Release-Begriffe. Es wird der jeweils bei der HVBInfo gebräuchliche Release-Begriff mit den wichtigsten Merkmalen dargestellt. Ein Merkmale ist die Plattform in der der Release-Begriff Verwendung findet. Eine weitere Unterscheidung ist das Testen in der IT bzw. in der QS (IT und QS siehe Abschnitt 3.2.2.5). Die Release-Begriffe implizieren auch, ob ein Release geplant ist oder ob es ad hoc auftritt.

Weitere relevante Bezeichnungen im dezentralen Bereich sind das *Voll-Release* und das *Updaterelease* (Beschreibung siehe Abschnitt 3.2.2.3.1). Folgende weitere Aussagen werden mit den in Tabelle 3.1 dargestellten Release-Begriffen assoziiert:

Durchlauf definierter Prozessabläufe: Bei manchen Release-Begriffen ist implizit der Workflow des Release-Durchlaufes enthalten. Bei Standard-Releases, Sonderversorgungen, Hotreleases z.B. legt der Release-Begriff den Ablauf der Abnahme- und Einführungsphase fest. Dieser Ablauf wird bei der HVB-Info Release-Durchlauf genannt. Ein Standard-Release im dezentralen Bereich folgt einem standardisierten Ablauf (siehe Abschnitt 3.4.3). Beim Securityfix im dezentralen Bereich z.B. kann der Ablauf je nach Dringlichkeit variieren. Wesentliche Merkmale der Release-Durchläufe sind die Umgebungen bzw. Phasen, in denen die Tests erfolgen. Hier wird unterteilt in die Integrationstestumgebung (IT-Umgebung) und die Qualitätssicherungsumgebung (QS-Umgebung) (siehe Abschnitt 3.2.2.5).

zeitlicher Faktor (Planung, Taktung): In den Release-Begriffen ist auch ein zeitlicher Faktor enthalten. In diesem Fall sagt der Release-Begriff aus, ob die Releases im Rahmen einer Jahres-/Monats- oder Wochenplanung vorab festgelegt sind oder ob sie ad hoc durchgeführt werden. Ein Standard-Release im dezentralen Bereich wird z.B. auf Jahresbasis eingeplant, während ein Hotrelease im Host-Bereich

Begriff	Plattform	Test in IT	Test in QS	geplant	Bemerkungen
Change	zS	ja	ja	ja	auf Applikationsebene
Standard-Release	Host, C/S	ja	ja	ja	
Sonderversorgung	C/S	ja	ja	meistens	
Sonder-Release	Host	ja	ja	meistens	entspricht Sonderversorgung C/S
Securityfix	C/S	ja, wenn frei	ja, wenn frei	ja/nein	mindestens ein Test nötig, Microsoft Patchday geplant, neuer Virus nicht geplant
Securityfix	zS	ja	ja	ja	falls Einstufung als Fatal, kann auf Test in IT und QS verzichtet werden
Mini-Release	C/S	ja	ja	ja	wird kundenspezifisch gehandhabt
Hotrelease	Host	nein	nein	ja	wöchentliche Softwareversorgungen
Hotfix	C/S	ja	ja	nein	
Hotfix	Host	nein	nein	nein	Incident muss vorliegen
Hotfix	zS	ja	ja	nein	
Corerelease	zS	ja	ja	ja	

Tabelle 3.1: Release-Begriffe in der HVBInfo

wöchentlich erfolgt. Außerdem implizieren die Release-Begriffe eine Taktung. Die Taktung der Hotreleases im Host-Bereich ist wöchentlich. Bei Standard-Releases im dezentralen Bereich liegt die Anzahl der durchgeführten Releases bei durchschnittlich vier pro Jahr. Hotfixes sind dagegen ungeplant und nicht getaktet.

Im Rahmen der SLA Vereinbarungen zwischen der HVBInfo mit ihren Kunden wird die Häufigkeit der Standard-Releases pro Jahr, evtl. die Anzahl der enthaltenen Produkte je Release-Bündel und die Durchlaufzeit je Release-Art mit den Kunden vereinbart.

Plattform: Ein Teil der Release-Begriffe wird plattformspezifisch verwendet. Der Begriff Hotrelease wird nur im Host-Bereich benutzt, während ein Corerelease spezifisch für den Bereich zS ist.

Release-Inhalt bzw. Umfang: Einige Release-Begriffe legen den Inhalt bzw. den Umfang eines Releases fest. Ein Corerelease bei zS beinhaltet lediglich Core-Software (siehe Kundenset Abschnitt 3.2.2.4), während ein Standard-Release bei zS nur Applikationen auf Common- oder Customize-Ebene beinhaltet. Beim Mini-Release wurde etwa mit dem Kunden HVB Corporates & Markets (MCB) vereinbart, dass es lediglich wenige Produkte beinhalten darf. Ebenso beinhaltet ein Hotfix bei diesem Kunden nur ein Produkt, währen bei anderen Kunden hier oft keine Einschränkung erfolgt.

3.2.2.3.1 Release-Begriffe im Vergleich zu ITIL Tabelle 3.2 zeigt die Zuordnung der oben beschriebenen Release-Begriffe zu den entsprechenden ITIL Begriffen. Sowohl das Voll- als auch das Updaterelease im dezentralen Bereich entsprechen dem ITIL-Begriff des Package-Releases. Das Voll-Release ist eine Bündelung aller beim Kunden im Einsatz befindlicher AWS. Es beinhaltet sowohl geänderte als auch unveränderte Software. Dieses Voll-Release wird für Notfälle zur Versorgung von Neugeräten aufgrund von Reparaturen oder für Lokationserweiterungen erstellt. Das Voll-Release wird im Release-Steuerungsprozess durch die IT- und QS-Phase gesteuert, ohne jedoch beim Kunden verteilt zu werden. Für einen kompletten Roll-out ist es in der Regel zu groß. Beim Kundenset KVZ etwa liegt die komprimierte Größe des Voll-Releases derzeit bei ca. sechs GByte. Beim Updaterelease dagegen wird nur geänderte Software gehandhabt. Es entspricht im ITIL-

Begriff HVBInfo	Plattform	entsprechende ITIL Begriffe
Change	zS	RfC
Standard-Release	Host, C/S	Package-Release, Major
Sonderversorgung	C/S	Package-Release, Major
Sonder-Release	Host	Package-Release, Major
Securityfix	C/S	Package-Release, Emergencyrelease, Minor
Securityfix	zS	Emergencyrelease
Mini-Release	C/S	Package-Release, Minor
Hotrelease	Host	Package-Release, Major und Minor
Hotfix	C/S	Package-Release, Emergencyrelease, eher Minor
Hotfix	Host	Emergencyrelease
Hotfix	zS	Emergencyrelease

Tabelle 3.2: Release-Begriffe in der HVBInfo im Vergleich zu ITIL

Sinne einer Bündelung von Delta-Releases und damit ebenfalls einem Package-Release. Anstelle des Begriffs Updaterelease wird bei der HVBInfo auch der Begriff *Roll-out-Release* verwendet.

Im Host-Bereich wird weder der Voll- noch Updaterelease-Begriff verwendet. Hier werden prinzipiell nur geänderte Release-Einheiten im Release-Prozess gehandhabt. Im ITIL-Sinne entspricht das KVS-Package einer Bündelung von Delta-Releases und somit einem Package-Release.

Im Bereich zS werden bei Corereleases die Änderungen zum Standard-Release gebündelt. Bei Changes werden Änderungen je einzeln AWS durchgeführt. Der jeweilige Verantwortliche entscheidet dann im Einzelfall, ob der komplette Server neu installiert wird, oder ob lediglich die Änderung eingespielt wird.

3.2.2.4 Kundenset

Die Zusammenstellung von AWS für ein Release-Bündel basiert auf der Grundlage von so genannten *Kundensets*. Das Kundenset definiert die Menge der AWS eines oder mehrerer Kunden bzw. von Kundenorganisationseinheiten.

Es wird aufgegliedert in:

Basispakete bzw. *Core*: Betriebssystem und betriebssystemnahe Software wie etwa Treiber

kundenneutrale Anwendungen bzw. *Common*: allgemein einsetzbare Software, z.B. Textverarbeitungssoftware oder Acrobat Reader

kundenspezifische Anwendungen bzw. *Custom*: z.B. Kreditbearbeitungsanwendung

Diese Einteilung hat den Vorteil, dass Änderungen von Software, die bei mehreren Kunden im Einsatz ist, nur einmal den Release-Prozess durchlaufen muss und nicht je Kunde extra. Software, die nur auf definierten Einzelarbeitsplätzen zum Einsatz kommt, wird nicht in das Kundenset integriert, sondern wird als so genanntes *AddOn* gehandhabt.

3.2.2.5 Umgebungen

Im Rahmen des Release-Prozesses werden die AWS bzw. Teilsysteme u.a. konfiguriert, getestet und verteilt. Dafür werden teilweise dedizierte Umgebungen bereitgestellt. Abbildung 3.4 zeigt die Phasen Entwicklung, Test und Betrieb wie sie im ITIL-Release-Prozess vorgesehen sind. Bei der HVBInfo ist der Testprozess hingegen in die zwei Phasen Integrationstestphase (IT-Phase) oder auch IT-Test-Phase, IT-Phase oder kurz IT und die Qualitätssicherungsphase (QS-Phase) oder auch QS-Phase bzw. kurz QS unterteilt. Die plattformspezifische Ausprägung dieser Phasen werden in den Abschnitten 3.4.3.2.3, 3.5.3 und 3.6.3.2.3 dargestellt. Diese Phasenbezeichnungen prägen auch die verwendeten Umgebungsbegriffe.

Im Sprachgebrauch wird dann für die IT, d.h. die IT-Test-Phase, die IT-Umgebung und für die QS-Phase die QS-Umgebung benötigt. Die Entwicklung findet in der Entwicklungsumgebung statt. Weitere Umgebungsbe-

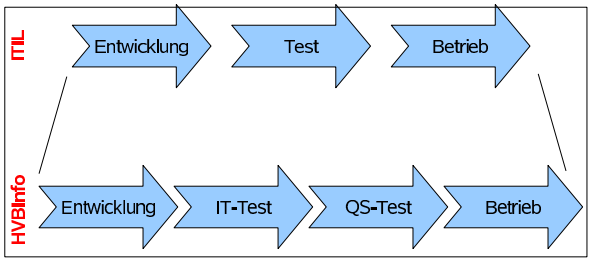


Abbildung 3.4: Phasenmodell des Testprozesses innerhalb des Release-Prozess bei der HVBInfo

griffe sind gemanagte Prelife-Umgebung sowie die Pilotumgebung. Die Umgebungsbegriffe werden nachfolgend kurz erläutert.

3.2.2.5.1 Entwicklungsumgebung Die Entwicklungsumgebung (EU) ist die entsprechende Hard- und Software mit der die Softwareentwickler arbeiten. Auf diese hat die HVBInfo im Bereich der Server oder Clients i.d.R. keinen Einfluss, außer die HVBInfo ist selbst wiederum der Betreiber dieser Umgebungen. Im Host-Bereich z.B. betreibt die HVBInfo für die Softwareentwickler der HVBSystems die so genannte gemanagte Entwicklungsumgebung (GET) .

3.2.2.5.2 Integrationstestumgebung Die Integrationstestumgebung (IT-U) ist die technische Infrastruktur für die IT-Phase. Die jeweiligen Verantwortlichen bauen im Rahmen des Release-Prozess für das Testen diese Infrastruktur auf. Im dezentralen Bereich steht hierfür extra Hardware zur Verfügung. Auf diese Hardware wird der Softwarestand der Produktivumgebung plus das neue Updaterelease aufgespielt. Beim Voll-Release wird auf diese Hardware das neue Voll-Release installiert. Beim Host-Release-Prozess erfolgt die IT-Phase in der GET. Für zS wird ebenfalls die entsprechende Hardware bereitgestellt. Teilweise wird diese dann jedoch auch in der nachfolgenden QS-Phase weiter verwendet.

3.2.2.5.3 Qualitätssicherungsumgebung Die Qualitätssicherungsumgebung (QS-U) bezeichnet die technische Infrastruktur für die QS-Phase. Im Vergleich zur IT-U, wo teilweise beliebige oder im dezentralen Bereich sogar virtualisierte Hardware verwendet wird, soll sie möglichst der Produktivumgebung nachgebildet werden. Das geschieht durch Verwendung von der gleichen Hardware, die auch produktiv im Einsatz ist. Die QS-U wird ebenfalls für das Testen im Release-Prozess aufgebaut. Für den dezentralen Bereich steht hier wiederum eigene Hardware zur Verfügung. Für die zS wird die Hardware zum Teil von der IT-Phase zur QS-Phase gewechselt. Im Host-Bereich erfolgt die QS-Phase bei Hotfixes und Hotreleases auf der GET. Die QS-Phase für Standard-Releases im Host-Bereich findet in der *gemanagte Prelife-Umgebung (GPL)* statt.

Die GPL wurde erstmalig im Projekt der Integration der Vereins- und Westbank eingesetzt. Der Begriff GPL wird auch verwendet, um eine produktionsähnliche Abnahmeumgebung aller Plattformen zu beschreiben. Hier besteht die GPL dann aus zS, C/S und Host.

Für die Abnahmen werden im dezentralen Umfeld in der QS-Phase je nach Kundenanforderung bis zu drei verschiedene Abnahmeumgebungen parallel aufgebaut. Zum einen die so genannte *Voll-Release-Umgebung* auf der ein PC mit dem Voll-Release versorgt wird. Zum anderen die *Updaterelease-Umgebung* und die *Hotfixumgebung*. Bei der Updaterelease-Umgebung werden auf die Hardware die produktiven Softwarestände und das Updaterelease aufgespielt. Die Hotfixumgebung entsteht durch das Aufspielen des Hotfixreleases auf einen PC mit produktiven Softwarestand.

3.2.2.5.4 Produktivumgebung Die Produktivumgebung (PROD) ist die Liveumgebung des jeweiligen Kunden. Im produktiven Umfeld gibt es weiterhin zwei Varianten für Pilotumgebungen.

Zum einen die so genannte *Filiale0* bzw. *Zentrale0*. Diese Umgebung ist in der produktiven Domäne des entsprechenden Kunden(sets). Sie wird ausschließlich für Installationstests durch den Roll-out-Verantwortlichen im dezentralen Bereich benutzt. Durch diese Tests soll die fehlerfreie und automatische Installation des Release-Bündels in der produktiven Umgebung sichergestellt werden. Ein synonymer Begriff dazu ist *Produktionsübergabeumgebung* (PÜ) . Diese Umgebung ist physisch im Gebäude der HVBInfo. Sie war früher die Schnittstelle der Softwareentwicklung der HVBSystems zum Betreiber und wurde nach der Verschiebung des dezentralen RM von der HVBSystems zur HVBInfo beibehalten. Sie kommt derzeit bei einer Beauftragung durch ein Projekt oder auf entsprechenden Kundenwunsch zum Einsatz. In diesen Pilotumgebungen werden nur Installationstests durchgeführt.

Ein Pilotbetrieb für die Kunden wird in den vom Kunden gewünschten Umgebungen durchgeführt. Im Fall der Hypovereinsbank kann das etwa eine definierte Filiale (Pilotfiliale) sein oder für andere Kunden spezielle Pilotarbeitsplätze oder -server.

Die begriffliche Verknüpfung von Phase und Umgebung führt in der Praxis oft dazu, dass z.B. von QS gesprochen wird und entweder die Umgebung QS-U oder die QS-Phase gemeint ist.

3.3 Motivation für die Integration der vorhandenen Release-Prozesse

In dieser Diplomarbeit wird ein Konzept zur plattformübergreifenden Integration des Release-Managements entworfen. Ausgangslage ist eine uneinheitliche, technologieabhängige Release-Prozesslandschaft bei der HVBInfo. Das bestehende RM wird ab Abschnitt 3.4 beschrieben und ab Kapitel 4 werden die durch das derzeitige RM resultierende Nachteile dargestellt. Durch das in der Arbeit ab Kapitel 5 entwickelte Konzept zum integrierten RM ergeben sich qualitative, organisatorische oder finanzielle Vorteile. Diese Vorteile, die durch das integrierte RM entstehen, werden nachfolgend beschrieben.

Abbildung 3.5 zeigt beispielhaft die derzeitige unterschiedliche Release-Taktung der jeweiligen Plattform.

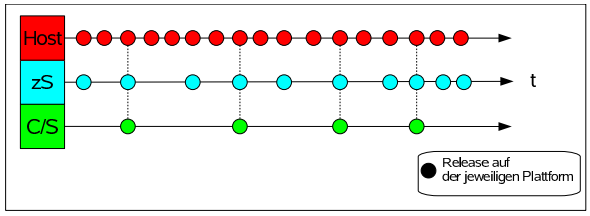


Abbildung 3.5: Schematische Darstellung der unterschiedlichen Release-Taktung je Plattform

Die gestrichelte Linie als Verbindung von Releases über alle Plattformen bedeutet, dass auch heute schon bei den Standard-Releases im dezentralen Bereich versucht wird, die anderen Plattformen terminlich zu synchronisieren. Es muss z.B. dafür gesorgt werden, dass die benötigten anderen Plattformen mit den entsprechenden Softwareständen zur Verfügung stehen und sich nicht etwa im Wartungszyklus befinden. Diese Abstimmung erfolgt weder durch die Unterstützung durch Werkzeuge, noch durch die Unterstützung durch Prozesse und ist deshalb mit einem hohen Aufwand verbunden. Das zukünftige plattformübergreifende Release-Management vereinfacht diese zeitliche Synchronisation. Das geschieht neben der Integration der Release-Prozesse auch durch die Unterstützung des RM mit einen neuen Release-Managementwerkeug (RM-Tool).

Aufgrund der verschiedenen Release-Zyklen und der damit verbundenen unterschiedlichen Änderungshäufigkeiten der AWS je Plattform entstehen hohe Testaufwände im Abnahmeprozess. Diese Aufwände resultieren aus dem Testen gegen verschiedene Entwicklungsstände. Das kann im Produktivbetrieb evtl. wieder zu Fehlern aufgrund von Inkompatibilitäten oder Schnittstellenproblemen führen. Abbildung 3.6 zeigt die Entstehung dieses Problems.

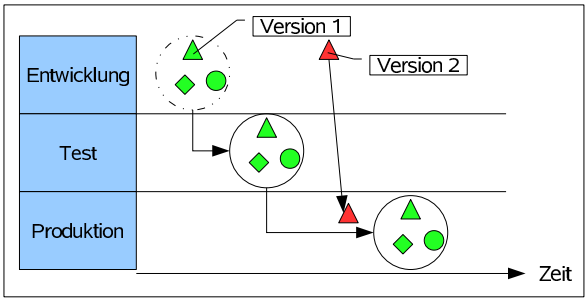


Abbildung 3.6: Problem durch überholendes Release

In der Grafik durchläuft ein Release-Bündel gemeinsam die Testphase und wird gemeinsam in die Produktivumgebung eingeführt. In der Zwischenzeit hat sich allerdings eine Version einer Release-Einheit selbst überholt. Das kann z.B. durch einen zwischenzeitlich erfolgten Hotfix (entspricht Emergencyrelease bei ITIL) passieren. In diesem Fall wurde das Release-Bündel in der Testphase mit einer veralteten Version (Version 1) getestet, was nicht mehr den Gegebenheiten in der Produktivumgebung (Version 2) entspricht.

Durch ein plattformübergreifendes Release-Management werden solche Probleme in Zukunft verhindert.

Die Integration der Release-Prozesse soll den gesamten Testaufwand reduzieren. Ein weiterer Vorteil ergibt sich durch die Möglichkeit zur Abnahmen von Geschäftsprozessen. Derzeit werden bei Abnahmen die Anwendungssysteme für sich alleine getestet. Eventuell erfolgen noch Tests von Schnittstellen zu abhängigen Systemen, wenn diese im Testprozess zur Verfügung stehen. Allerdings erfolgen keine Tests von Geschäftsprozessen, weil nicht sichergestellt ist, dass im Release-Prozess alle relevanten Plattformen zur Verfügung stehen. Der Test von Geschäftsprozessen wurde projekthaft im Rahmen der Integration der Vereins- und Westbank im Frühjahr 2005 durchgeführt. In der GPL, einer realen Bedingungen nachgebildeten Abnahme-Umgebung, konnten definierte Geschäftsprozesse abgenommen werden. Als Ergebnis wurde, wie Abbildung 3.7 zeigt, eine deutliche Reduzierung der Incidents vor der endgültigen Einführung des neuen Systems erreicht.

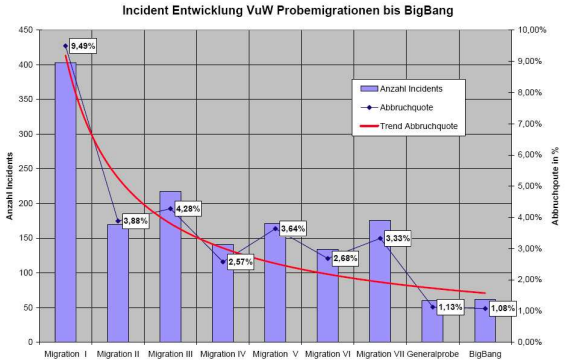


Abbildung 3.7: Reduzierung der Incidents bei der Integration der Vereins- und Westbank

Ein weiterer Vorteil wird durch die plattformübergreifende Einführung neuer Software in den Produktivbetrieb beim Kunden erhofft. Dadurch fallen evtl. weniger Umstellungs- oder Schulungsaufwände an. Ein plattformübergreifendes RM ermöglicht ebenfalls die gesamthafte Planung für alle Kundenanforderungen.

Vorteile organisatorischer Art entstehen durch einen einzigen stabilen Prozess und durch die einheitliche Dokumentation. Durch die Vereinheitlichung der Begriffe soll auch mehr Verständis der Prozessbeteiligten füreinander geschaffen werden. Durch Transparenz des Prozesses sollen rechtzeitige Eskalationen möglich werden. Für die Kunden gibt es dann einen Ansprechpartner zu Release-Managementthemen. Der Auftritt des RM nach Außen ist dadurch einheitlich.

Finanzielle Vorteile sollen sich durch die gesteigerte Qualität in einer Reduktion der *Incidents* auswirken. Auch soll durch die übergreifende Planung die Testumgebung besser ausgelastet werden. Einsparungspotenzial ergibt sich weiterhin durch die Vereinheitlichung der Tools.

Zur Strukurierung der Aufgabe der Diplomarbeit, der Integration der Release-Prozesse, wird nachfolgend zunächst jeder plattformspezifische Release-Prozess beschrieben. Diese Prozessbeschreibungen wurden inhaltlich durch einen HVBInfo-internen Review abgesichert. Die Gliederung der jeweiligen Prozessbeschreibung erfolgt anhand der Gliederung des ITIL-Release-Prozesses. Dieses Vorgehen wurde gewählt, obwohl die jeweiligen plattformspezifischen Prozesse eher einen SWE-Fokus, als einen ITSM-Fokus haben. Grund ist die

geplante Orientierung des integrierten Prozesses an ITIL. Dadurch soll der Zielprozess besser zu den bestehenden Prozessen der HVBInfo passen, die bereits nach ITIL ausgerichtet sind.

Die Informationen für die Beschreibung des nachfolgenden Release-Managements im dezentralen Bereich, Bereich zS und Host-Bereich wurden durch Studieren von vorhandenen Dokumenten, Interviews und eigenen Beobachtungen gesammelt.

3.4 Release-Management im dezentralen Bereich

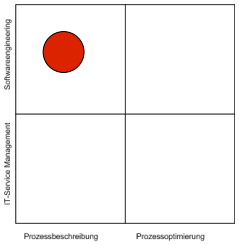


Abbildung 3.8: Einordnung RM dezentral

Ein Release im C/S-Bereich entspricht im ITIL-Sinne einer Bündelung von Produkten zum Zweck der gemeinsamen Testdurchführung und der gemeinsamen Einführung in die Produktivumgebung. Der Ablauf der Aktivitäten im Rahmen dieser Testdurchführung und Einführung in die Produktivumgebung wird bei der HVBInfo auch *Release-Durchlauf* genannt. Das Ziel des RM bei der HVBInfo ist die termingerechte und störungsfreie Bereitstellung von Software für den Kunden.

In diesem Abschnitt werden die am Release-Prozess beteiligten Rollen mit ihren Aufgaben beschrieben. Des Weiteren erfolgt anhand der Gliederung des ITIL-Release-Prozesses (vgl. Abschnitt 2.3.4.3) die Einordnung der bestehenden Aktivitäten im dezentralen Bereich. Dadurch sind Abweichungen vom ITIL-Prozess leichter darstellbar.

3.4.1 Beteiligte Rollen und ihre Aufgaben

Der Release-Manager Client/Server (RM-C/S) trägt die Verantwortung für den Release-Durchlauf. Abbildung 3.9 auf Seite 80 zeigt die Zusammenhänge zwischen RM, Change-Management und Softwareentwicklung. Im Vergleich zu ITIL (vgl. Abbildung 2.58) gibt es in diesem Bereich keine Berührungspunkte zwischen dem Change-Management und dem RM. Wichtige Ansprechpartner für das RM kommen deshalb aus dem Bereich der Softwareentwicklung.

Nachfolgend werden alle Rollen beschrieben, welche am Release-Prozess beteiligt sind. Der RM-C/S interagiert und kommuniziert mit Umgebungsverantwortlichen (UV) , Produktverantwortlichen (PV) oder Anwendungssystemverantwortlichen (AWSV) , Verschalern bzw. Paketierern, Systemintegratoren (SI) , Kunden, Roll-out-Verantwortlichen bzw. Einsatzverantwortlichen (EV) bzw. Softwareversorgern (SWV) , Einführungsmanagern (EM) , Security, Lizenzmanagement, und dem Change-Management. Zu den hier aufgeführten Rollen werden in Tabelle 3.3 die releaserelevanten Aufgaben beschrieben.

Rolle	Aufgaben	Bemerkungen
Change-Management	• genehmigen der Pilot- und Roll-out-Changes	
Einführungs-management (EM)	• fachliche Abstimmung des Release-Inhaltes • genehmigen von Sonderversorgungen oder Hotfixes • koordinieren und begleiten der Pilotierung • kommunizieren der Änderungen und Neuerungen an die Benutzer	diese Rolle wird meist von Mitarbeitern der HVBSystems wahrgenommen

Rolle	Aufgaben	Bemerkungen
Fachbereich	• offizielle Abnahme jedes Produktes	Testen und Abnahme siehe Abschnitt 3.4.2.3; jeder PV entscheidet selbst, wann der Fachbereich zur Abnahme erscheinen soll
Kunde		wird i.d.R. durch die jeweiligen Fachabteilungen bzw. Fachbereiche vertreten
Lizenzmanagement	• überprüfen der Lizenzen bei Neuinstallationen	
Produkt- oder Anwendungssystem-verantwortlicher (PV, AWSV)	• meldet Produkte für ein Release ein (siehe Abschnitt 3.4.3.2.1) • stellt die Release-Einheit für den Release-Prozess zur Verfügung • testet Produkt in IT- und QS-Phase	beide können auch gleichzeitig die Entwickler dieser Software sein; falls die Software bei der HVBSystems entwickelt wird, wird die Rolle von einem dortigen Mitarbeiter wahrgenommen; für fremdentwickelte Software gibt es bei der HVBInfo den so genannten *PV-light*; dieser nimmt im Release-Prozess die gleichen Aufgaben wie der eigentliche PV wahr; aus diesem Grund wird nachfolgend nicht mehr zwischen PV und PV-light bzw. Software der HVBSystems und fremdentwickelter Software unterschieden
Release-Manager (RM-C/S)	• Release planen • Release eröffnen • Change eröffnen • Produkt- und Release-Status überwachen und steuern • Release-Durchlauf bewerten	Release-Planung siehe Abschnitt 3.4.2.1; Aktivitäten im Release-Steuerungsprozess siehe Abschnitt 3.4.3
Security	• überprüfen der Produkte auf Einhaltung der Securityrichtlinien	
Softwareversorger (SWV)	• Verteilungsplan erstellen und veröffentlichen • verteilen und installieren des Releases in der Zielumgebung • betreiben der PÜ • dokumentieren und melden von Störungen und Installationsproblemen beim Roll-out	Roll-out-Planung siehe Abschnitt 3.4.2.4 und Roll-out in Abschnitt 3.4.2.6

Rolle	Aufgaben	Bemerkungen
Systemintegrator (SI)	• bündeln der Software zu einem einzigen Package-Release • sicherstellen der technischen Lauffähigkeit des Package-Releases • testen der Installationsprozedur	Testen siehe Abschnitt 3.4.2.3
Umgebungsverantwortlicher (UV)	• bereitstellen der entsprechenden Abnahmeumgebungen • betreiben dieser Umgebungen	Umgebungen siehe Abbildung 3.14
Verschaler oder auch Paketierer	• entgegennehmen der Software von der Theke • überprüfen der Software anhand definierter allgemeingültiger Eingangskriterien • erstellen der Installationshülle (= verschalen bzw. paketieren)	Release-Zusammenstellung und -Konfiguration siehe Abschnitt 3.4.2.2

Tabelle 3.3: Rollen und Aufgaben im dezentralen Bereich

3.4.2 Release-Aktivitäten im dezentralen Bereich

Abbildung 3.10 zeigt den Zusammenhang der wichtigsten Release-Aktivitäten im dezentralen Bereich. Der RM-C/S ist zusammen mit dem Service Level Manager für die Festlegung der Releasepolicy zuständig (siehe nächster Abschnitt 3.4.2.1). Weiterhin plant der RM-C/S Releases (siehe Abschnitt 3.4.2.1.2) und er steuert den Release-Steuerungsprozess. Die detaillierte Beschreibung der Aktivitäten im Release-Steuerungsprozess erfolgt in Abschnitt 3.4.3. Die Gliederung in diesem Abschnitt orientiert sich wieder an der Gliederung des ITIL-Release-Prozesses (siehe Abschnitt 2.3.4.3).

3.4.2.1 Festlegen der Releasepolicy und des Release-Plans

Nach ITIL gehören die Festlegung der Releasepolicy und des Release-Plans zu den Aufgaben des RM (vgl. Abschnitt 2.3.4.3.1). Beide Aktivitäten findet auch im dezentralen Bereich statt. Nachfolgend wird beschrieben, welche Aspekte der Releasepolicy festgelegt werden und wie die Release-Planung erfolgt.

3.4.2.1.1 Festlegen der Releasepolicy im dezentralen Bereich Das RM wird bei der HVBInfo als Dienstleistung im Service Katalog angeboten. Aus diesem Grund legt neben dem RM auch das Service Level-Management Inhalte der Releasepolicy fest. Im Rahmen von SLA-Vereinbarungen werden mit den jeweiligen Kunden die Anzahl der Releases pro Jahr und für die Einhaltung der SLA bestimmte Messkriterien festgelegt. Solche Kriterien sind z.B. die Einhaltung der im Release-Plan definierten Roll-out-Zeiten oder die Einhaltung der für die Releases vereinbarten Durchlaufzeiten. Folgende weitere Inhalte einer Releasepolicy analog ITIL (vgl. [OGC04]) sind festgelegt:

Festlegung der Release-Einheiten Im dezentralen Bereich erfolgt die organisatorische Handhabung der Release-Einheiten durch das RM auf Teilsystemebene.

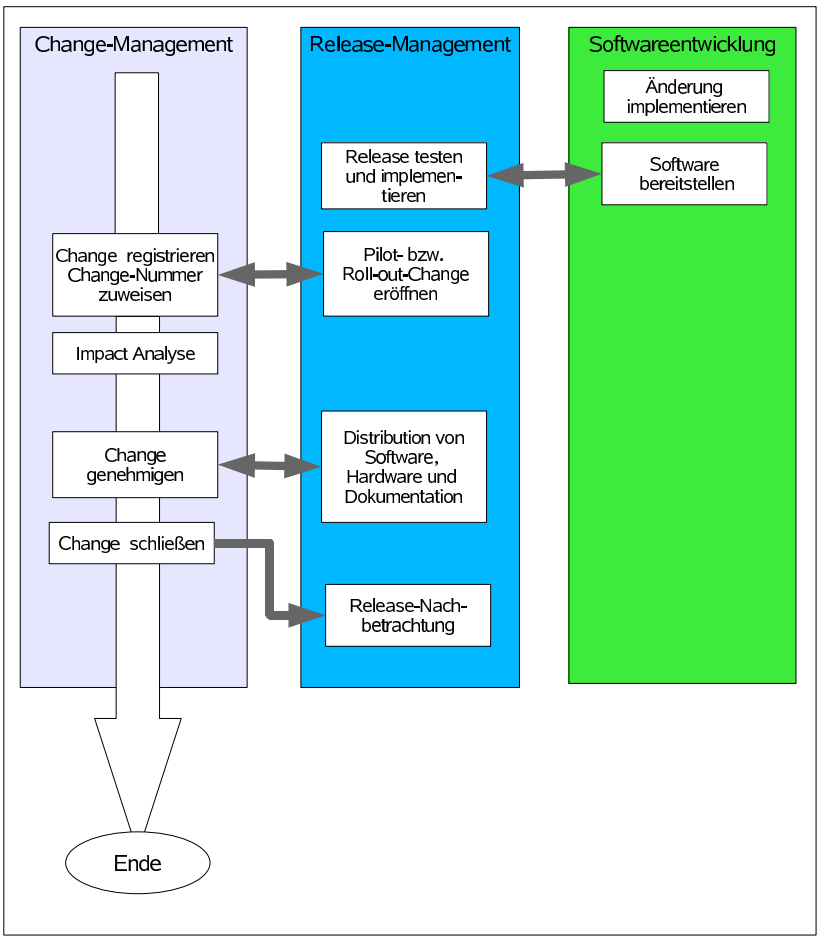


Abbildung 3.9: Übersicht Change- und Release-Management und Softwareentwicklung im Bereich C/S

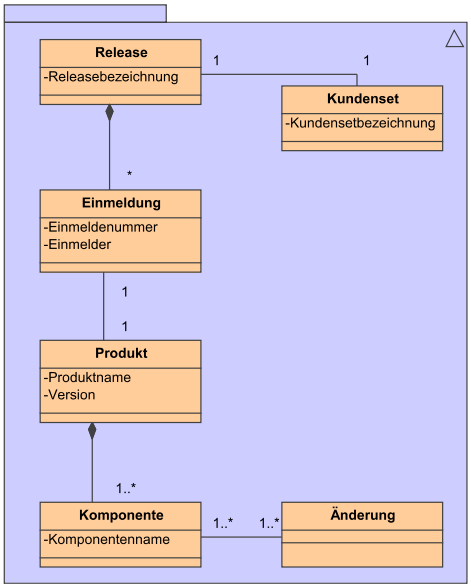


Abbildung 3.11: Zusammenhang zwischen Release, Einmeldung und Produkt

Hierfür wird synonym auch der Begriff Produkt verwendet. In der technischen Abwicklung, d.h. z.B. Bereitstellung durch den Softwareentwickler oder Installation, können allerdings lediglich Subkomponenten dieser Produkte auftreten.

Abbildung 3.11 zeigt den Zusammenhang u.a. zwischen Release, Einmeldung und Produkt. Der Release-Inhalt für ein bestimmtes Kundenset ergibt sich durch die Summe seiner Einmeldungen. Eine Einmeldung bezieht sich immer auf ein Teilsystem bzw. ein Produkt. Dieses Produkt kann wiederum aus mehreren Komponenten zusammengesetzt sein, von denen sich lediglich ein Teil geändert hat. Welche Komponenten sich hierbei konkret geändert haben ist dem RM nicht bekannt.

Release-Bezeichnung und -Nummerierung Die Release-Bezeichnung wird zusammengesetzt aus der Kundensetbezeichnung und einer fortlaufenden Nummerierung je Jahr. Weiterhin ist aus der Release-Bezeichnung die Release-Art ersichtlich. Das erste Standard-Release im Jahr 2006 für das Kundenset KVZ lautet dann KVZ 1-2006 Voll + Update, da hier für das erste Standard-Release sowohl ein Voll- als auch ein Updaterelease parallel durchgeführt werden.

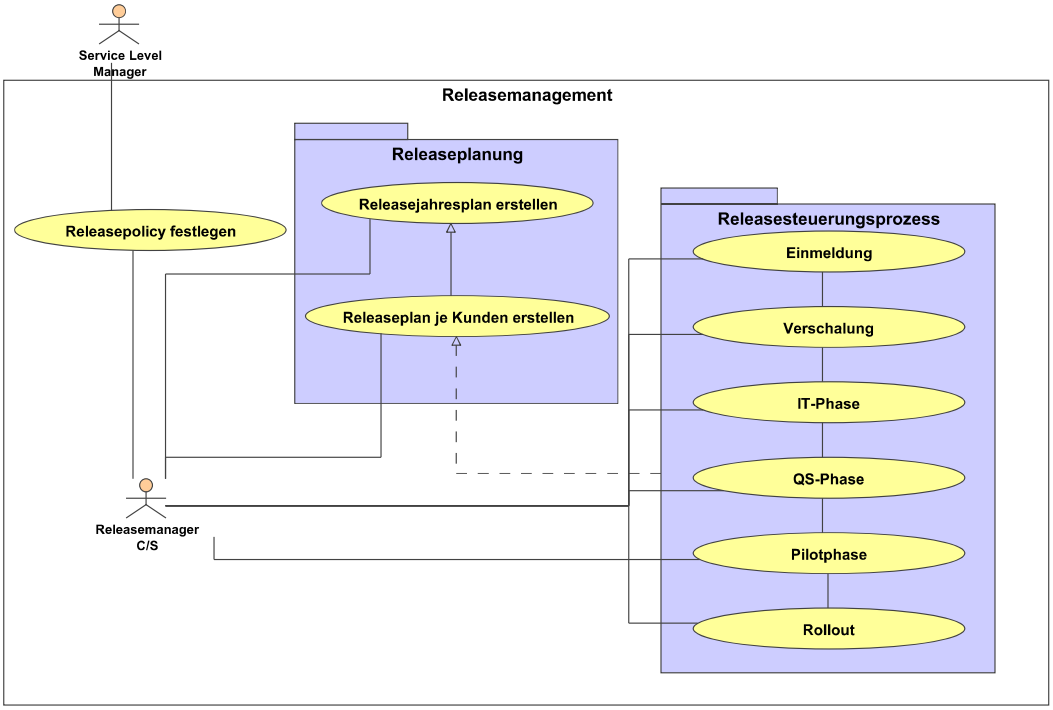


Abbildung 3.10: Darstellung der wichtigsten Release-Aktivitäten im dezentralen Bereich

Bei Hotfixes bezieht sich in der Bezeichnung auf das Standard-Release zu dem Fehler behoben werden müssen. Eine Fehlerbehebung zum Standard-Release 1-2005 wird dann als Hotfix1_1_2005 bezeichnet.

Definition von Major- und Minor-Releases und einer Regel für Emergencyfixes Die Zuordnung der bestehenden Releases zu Major- bzw. Minor-Releases ist in Tabelle 3.2 in Abschnitt 3.2.2.3.1 beschrieben. Für die Durchführung von Sonderversorgungen und Hotfixes sind Regeln definiert. Für die PV hat das RM hierfür Checklisten veröffentlicht. In diesen Checklisten stehen etwa notwendige Genehmigungen und Arbeitsschritte, welche der PV veranlassen muss.

Häufigkeit von Major- und Minor-Releases Die Häufigkeit von Standard-Releases wird in den SLA festgelegt. Die Häufigkeit aller anderen Releases (Sonderversorgungen, Hotfixes, Securityfixes) werden nicht vorab festgelegt. Diese werden jeweils nach Bedarf durchgeführt. Ihre Durchführung muss vom EM, Vorgesetzten des PV und RM genehmigt werden.

Identifikation von geschäftskritischen Zeiten, in denen Implementationen vermieden werden sollen Zeiten, in denen keine Veränderungen an der Liveumgebung der Kunden durchgeführt werden dürfen, werden *Frozen Zone* oder *Ultimo* genannt. Diese werden von den jeweiligen Kunden festgelegt. Häufig liegen diese Zeiten jeweils am Monats- und Jahresende und müssen bei der Release-Planung berücksichtigt werden.

Erwartete Begleitdokumentation für jeden Release-Typ Für jedes Produkt muss ein durch Security freigegebenes Securitykonzept und eine vom Kunden erfolgte und dokumentierte fachliche Freigabe vorliegen. Die Dokumentation des einzelnen Release-Durchlaufes und der im Release-Durchlauf gehandhabten Produkte erfolgt in Stargate. Hierfür wird für das Release ein Release-Begleitblatt und für die zum Release gebündelten Produkte je ein Produktbegleitblatt geführt (siehe Abschnitt 3.4.3.1). Andere Begleitdokumente wie etwa Installationsanleitungen oder Benutzerhandbücher sind zwar nötig, deren Vorliegen wird allerdings nicht vom RM vorgegeben bzw. überprüft.

Richtlinien, wie und wo Releases dokumentiert werden sollen Die Dokumentation der Releases erfolgt über *Stargate*. Stargate ist ein Tool zur Unterstützung des Release-Prozesses. Es basiert auf Version 8 von *PVCS Dimensions* [PVCS05]. PVCS Dimensions ist ursprünglich ein Tool zur Versionsüberwachung von Software. Es wird im Softwareentwicklungsbereich im Rahmen des SCM eingesetzt. Die Toolauswahl erfolgte, als das RM noch bei der HVBSystems durchgeführt wurde. Das Tool wurde intern mit einer Workflow-Unterstützung des Release-Prozesses erweitert und Stargate genannt.

Regel für Erstellen und Testen von Back-out Plänen Back-out Pläne werden von den SWV erstellt. Die Regeln für das Erstellen und Testen von Back-out Plänen werden vom ebenfalls von den SWV festgelegt.

Zuständigkeiten des RM Das RM ist verantwortlich für die qualitätsgesicherte und termingerechte Bereitstellung von Änderungen an der betroffenen IT-Struktur. Die hierfür anfallenden Aufgaben sind in Tabelle 3.3 aufgeführt.

Beschreibung des Release-Steuerungsprozesses Der Release-Steuerungsprozess wird vom RM selbst beschrieben und ist im so genannten *System Information Management-Portal (SIM-Portal)* dokumentiert. Im SIM-Portal werden intern freigegebene Dokumente veröffentlicht. Zum RM im dezentralen Bereich sind die Prozessbeschreibung, Verfahrensanweisungen, Arbeitsanweisungen, Formulare und Checklisten im SIM-Portal veröffentlicht. Die detaillierte Beschreibung des Release-Steuerungsprozesses steht in Abschnitt 3.4.3.

Dokumentation der exakten DSL-Konfiguration Eine DSL als solche ist bei der HVBInfo nicht definiert. Die Originalsoftware wird von den PV verwaltet und über die Theke für den Release-Prozess am so genannten *Übergabeserver* zur Verfügung gestellt. Von dort wird sie zwischen den verschiedenen Übergabe- und Codeservern kopiert (siehe Abbildung 3.15).

3.4.2.1.2 Festlegen des Release-Plans im dezentralen Bereich In diesem Abschnitt wird dokumentiert, wer die Release-Planung durchführt, was Gegenstand der Planung ist und wie die Planung abläuft.

Zuständigkeiten für die Planung Die Erstellung der Release-Pläne ist die Aufgabe des RM-C/S. Die Kunden, EM, UV, SI, SWV und Verschaler werden von ihm für Abstimmungen herangezogen.

Gegenstand der Planung Ausgehend von der in den SLAs vereinbarten Anzahl der Standard-Releases und vorab bekannten Sonderversorgungen wird für jedes Kundenset ein eigener Release-Jahresplan erstellt. Die Release-Pläne beinhalten die Termine von Meilensteinen der jeweiligen Release-Prozessphasen. Diese Meilensteine sind (Beginn bzw. Ende) Einmeldung, Verschalung, IT- und QS-, Pilotphase und der eigentliche Roll-out. Diese werden nachfolgend detaillierter beschrieben. Bestimmendes Element bei der Planung sind die Belegungszeiten der verschiedenen Testumgebungen.

Im Vergleich zu ITIL ist die Schnittstelle zum Change-Management hier nicht gegeben (siehe Abbildung 3.9). Der RfC, d.h. die zu implementierende genehmigte Änderung spielt bei dieser Planung keine Rolle. Somit sind die Release-Inhalte bei der Release-Planung vorher nicht bekannt. Der Release-Inhalt ergibt sich durch die Summe der Einmeldungen von Produkten durch die PVs.

Kajko-Mattsson et. al. prägen hierfür den Begriff der zeitplangetriebenen Release-Planung (vgl. Abschnitt 2.3.3.1). Zu den entsprechenden Roll-out-Terminen werden alle bis dahin fertiggestellten Änderungen eingeführt. In Abschnitt 2.1.1.7.2 wurde diese Art der Planung auch als „Überlebensmodus“beschrieben. In diesem Überlebensmodus ist der Zeitmaßstab zur Reduzierung von Arbeitsrückständen oder Arbeitsüberhang ausschlaggebend.

Ein Engpass und damit verbundener Arbeitsüberhang entsteht im dezentralen Bereich häufig bei den Verschalern und SI. Nachdem vorab die Release-Inhalte nicht bekannt sind, müssen diese oft Überstunden leisten,

weil die Anzahl der übergebenen Produkte zu hoch ist. Das RM hat in Absprache mit dem EM und SI mittlerweile auf diese Überbuchung reagiert. In Zukunft ist deshalb eine Begrenzung der Anzahl der Einmeldungen geplant. Dadurch soll ein Zeitpuffer gewonnen werden. Die Verschaler vergeben in Zukunft eine von vier verschiedenen Verschalungskategorien an das Produkt. Eine Einstufung in eine der Kategorien entspricht dem benötigten Ressourcenbedarf. Ein leicht verschalbares Produkt wird dann z.B. in die Kategorie A (geringer Aufwand), ein neues Produkt in die Kategorie S (Spezialaufwand) eingeordnet. Weitere Kategorien sind B (mittlerer Aufwand) und C (hoher Aufwand). Jeder dieser Kategorien ist eine Zeitdauer zugeordnet. In Zukunft soll sich diese Kategorie auf die Verschalungstermine auswirken. Ein Produkt der Kategorie S muss dann früher bereitgestellt werden, als ein Produkt der Kategorie A. So sollen in Zukunft Auslastungsspitzen des Verschalungspersonals vermieden werden.

Ablauf der Planung Der RM-C/S plant jährlich die Standard-Releases, Sonderversorgungen und Mini-Releases der jeweiligen Kunden(sets). Sonderversorgungen sind zusätzliche Releases, die oft aufgrund von gesetzlichen Rahmenbedinungen zu bestimmten Terminen ausgerollt werden müssen und daher nicht im Rahmen eines Standard-Releases gehandhabt werden können. In der Regel werden im Rahmen von Sonderversorgungen und Minireleases deutlich weniger Softwareprodukte an den Kunden ausgeliefert, als beim Standard-Release.

Üblicherweise werden zunächst die Releases des größten Kunden eingeplant. Für ihn werden zunächst die Abnahmezeiträume und damit die Belegung von Testumgebungen festgelegt. Bei diesen Zeitplänen müssen ggf. Frozen Zones oder Wartungszeiten der Testumgebungen berücksichtigt werden. Ausgehend von den Terminen der Abnahmezeiträume (IT-Phase und QS-Phase) werden die vor- und nachgelagerten Prozessphasen terminiert.

Standard-Releases, Sonderversorgungen und Mini-Releases kennzeichnen sich durch den gleichen vorab definierten Release-Prozess. Dieser betrifft im wesentlichen die Anzahl und Art der Abnahmen und Abnahmephasen und damit die benötigten Testumgebungen. Bei Standard-Releases wird weiter unterteilt in Voll- oder Updatereleases. Eine Sonderversorung bzw. ein Mini-Release ist in der Regel ein Updaterelease. Für das *Kundenset Vertrieb und Zentrale* (KVZ) sind z.B. für das Jahr 2005 fünf Standard-Releases, davon zwei Voll-Releases, vereinbart worden.

Bei der Planung können Securityfixes und Hotfixes in der Regel nicht betrachtet werden. Sie betreffen die Behebung von dringenden Sicherheitsproblemen oder Fehlern im laufenden Betrieb und treten daher ad hoc auf. Im ITIL-Sinne entsprechen sie den Emergencyreleases. Das RM muss darauf entsprechend reagieren. Manchmal werden dadurch Verschiebungen oder sogar Stornierungen der vorab geplanten Releases notwendig. Das gefährdet u.U. die Einhaltung der in den SLA vereinbarten Release-Anzahl.

Ausnahme in diesem Bereich sind die regelmäßigen Patchtage diverser Softwarehersteller (z.B. Microsoft). Diese Termine sind meist vorab bekannt und können zumeist auf monatlicher Basis eingeplant werden. Securityfixes und Hotfixes unterscheiden sich im Durchlauf zu Standard-Releases durch einen verkürtzen Testprozess. Die terminlich eingeplanten Prozessschritte bzw. -phasen werden nachfolgend beschrieben. Sie kennzeichnen Meilensteine die zur Steuerung und Kontrolle des Prozesses erreicht werden müssen (vgl. Release-Steuerungsprozess in Abbildung 3.10).

Einmeldung: Die Phase der *Einmeldung* ist der Zeitraum in dem die PVs ihre Release-Einheiten für die Erstinstallation, Änderung oder Deinstallation in ein Release-Bündel *einmelden*. Die Einmeldung kann mit der Reservierung einer Fahrkarte verglichen werden. Durch die alleinige Reservierung ist noch nicht die Mitfahrmöglichkeit garantiert, ein Bus kann z.B. zu überfüllt sein. In dieser Phase muss noch keine physische Softwareübergabe erfolgen.

Verschalung: Die Prozessphase *Verschalung* subsumiert die Aktivitäten der Verschalung. Hier wird für jedes Produkt eine Installationshülle erstellt. Diese verschalten Produkte werden anschließend in das Release-Bündel integriert. Anstelle von Verschalung wird auch der Begriff Paketierung verwendet.

IT: In der IT-Phase erfolgen Integrationstest. Für diese Phase plant das RM-C/S üblicherweise eine Iteration ein. Beim Auftreten eines Fehlers kann so eine korrigierte Version nachgeliefert werden. Anstelle von IT-Phasen wird auch von Abnahmen in der Systemintegrationsumgebung (SI) gesprochen.

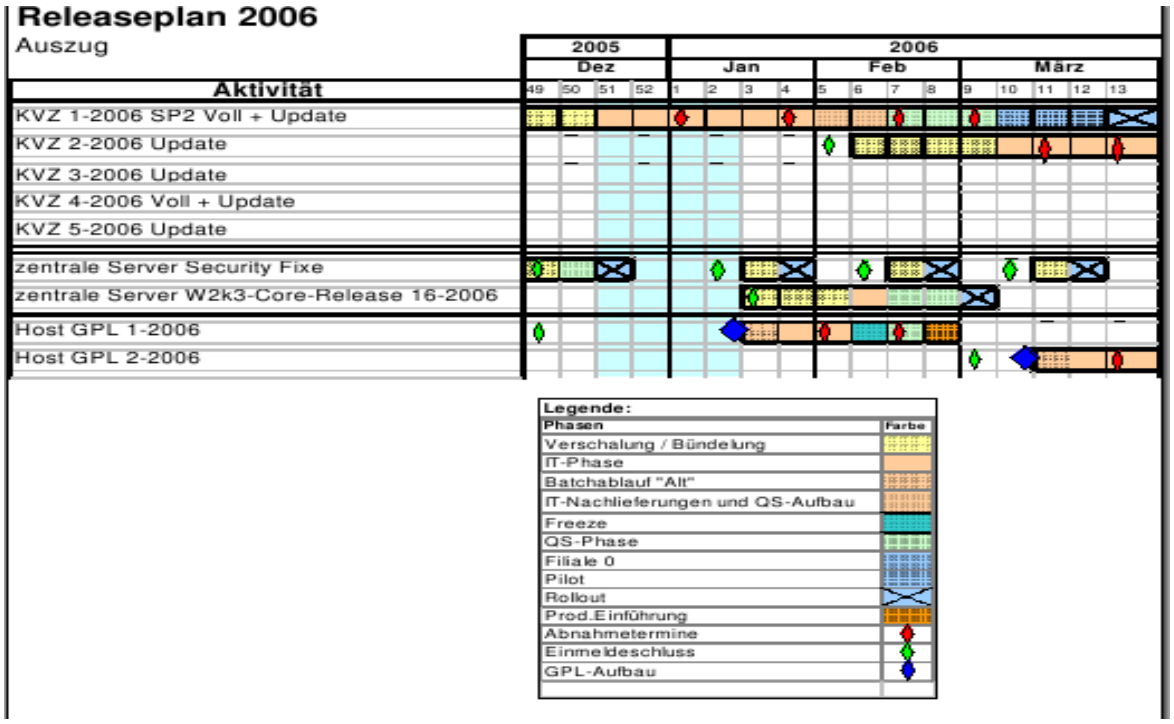


Abbildung 3.12: Auszug aus der Übersicht aller Release-Pläne der HVBInfo

QS: In der QS-Phase wird das komplette Release-Bündel in der QS-U installiert. In dieser Phase erfolgen Installations- und Akzeptanztests. (Anmerkung: Teilweise werden aufgrund der hohen Fehlerzahlen die Akzeptanztests in die zweite IT-Phase verlagert. Dadurch ist eine weitere Nachlieferung einer korrigierten Version für die QS-Phase möglich.)

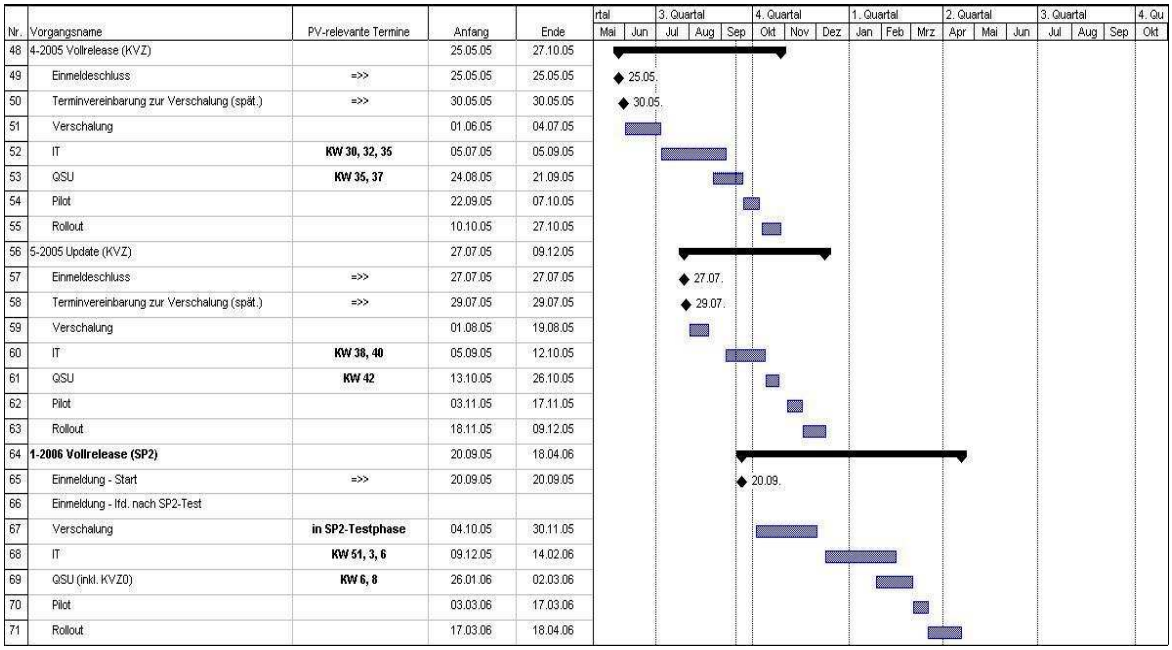
Pilot: Die Pilotphase kann zum einen als reiner Installationstestpilot in der PÜ erfolgen, zum anderen gibt es dedizierte Pilotumgebungen für Betatests der Benutzer.

Roll-out: Wenn das komplette Release freigegeben wird, erfolgt der Roll-out. Wenn der Roll-out an mehreren Kundenstandorten gleichzeitig erfolgt wird auch vom so genannten *Flächen-Roll-out* gesprochen.

Toolunterstützung bei der Release-Planung Die Planung der Releases erfolgt mithilfe von MS Excel und MS Project. Abbildung 3.12 zeigt einen Auszug aus der Gesamtübersicht aller Release-Termine.

Diese Excel-Liste stellt alle relevanten Release-Phasen aller Kundensets bezogen auf ein Jahr gegenüber. Die Release-Planung startet üblicherweise mit der Planung der Releases des grössten Kunden. Die Release-Planung aller anderen Kunden erfolgt dann je nach Verfügbarkeit der Abnahmeumgebungen. Aus dieser Excel-Liste ist auch ersichtlich, dass hier bereits Abnahmephasen im Bereich Host-GPL und bei Windows zS mit eingetragen werden. Dieser Plan wird im Intranet als Terminübersicht veröffentlicht. Die Eintragungen in dieser Datei dienen auch als Grundlage einer weiteren Excel-Liste, in der alle Termine auf Tagesbasis geführt werden. Diese dient den RM-C/S als tägliche Arbeitsgrundlage. Abbildung 3.13 zeigt den Ausschnitt aus einem Release-Plan der HVBInfo für ein bestimmtes Kundenset. Dieser wird aus den Terminen in der vorherigen Excel-Liste in MS Project generiert. Die Daten werden hierzu manuell von Excel in Project übertragen. Dieser Plan wird im Intranet der HVBInfo, HVBSystems und des jeweiligen Kunden veröffentlicht.

Von diesen Terminen wird lediglich der Einmeldeschlusstermin in Stargate manuell übertragen. Nach Ende der Einmeldephase können daher keine Einmeldungen von Produkten durch den PV mehr erfolgen.



Nr.	Vorgangsname	PV-relevante Termine	Anfang	Ende
48	4-2005 Vollrelease (KVZ)		25.05.05	27.10.05
49	Einmeldeschluss	=>>	25.05.05	25.05.05
50	Terminvereinbarung zur Verschalung (spät.)	=>>	30.05.05	30.05.05
51	Verschalung		01.06.05	04.07.05
52	IT	**KW 30, 32, 35**	05.07.05	05.09.05
53	QSU	**KW 35, 37**	24.08.05	21.09.05
54	Pilot		22.09.05	07.10.05
55	Rollout		10.10.05	27.10.05
56	5-2005 Update (KVZ)		27.07.05	09.12.05
57	Einmeldeschluss	=>>	27.07.05	27.07.05
58	Terminvereinbarung zur Verschalung (spät.)	=>>	29.07.05	29.07.05
59	Verschalung		01.08.05	19.08.05
60	IT	**KW 38, 40**	05.09.05	12.10.05
61	QSU	**KW 42**	13.10.05	26.10.05
62	Pilot		03.11.05	17.11.05
63	Rollout		18.11.05	09.12.05
64	**1-2006 Vollrelease (SP2)**		20.09.05	18.04.06
65	Einmeldung - Start	=>>	20.09.05	20.09.05
66	Einmeldung - lfd. nach SP2-Test			
67	Verschalung	**in SP2-Testphase**	04.10.05	30.11.05
68	IT	**KW 51, 3, 6**	09.12.05	14.02.06
69	QSU (inkl. KVZ0)	**KW 6, 8**	26.01.06	02.03.06
70	Pilot		03.03.06	17.03.06
71	Rollout		17.03.06	18.04.06

Abbildung 3.13: Auszug aus einem Release-Plan der HVBInfo

3.4.2.2 Entwurf, Releasezusammenstellung und -Konfiguration

Der Entwurf eines Release erfolgt nicht durch das RM. Der jeweilige Inhalt eines Releases ergibt sich aus der Summe der Einmeldungen (Einmeldung siehe Abschnitt 3.4.3.2.1). Die technische Voraussetzung für die Zusammenstellung eines Package-Releases ist die Paketierung jedes einzelnen übergebenen Softwareproduktes. Das ist die Aufgabe des *Verschalers* bzw. Paketierers. Er erstellt eine so genannte Installationshülle oder *Schale*. Durch diese Installationshülle installiert sich jedes Produkt automatisch und ohne weitere Eingriffe in der Zielumgebung. In der Installationshülle sind auch evtl. nötige Konfigurationseinstellungen wie etwa Registry-Einträge oder Installationspfade definiert.

Für die Verschalung muss der PV für jedes eingemeldete Produkt vorab Verschalungstermine vereinbart haben. Das RM erinnert die PV per Mail an diese Aufgabe.

Der Verschaler nimmt dann das jeweilige Produkt vom Übergabeserver (siehe Abbildung 3.15) und bearbeitet es auf seinem eigenen Arbeitsplatz. Zunächst überprüft er die Einhaltung der Eingangskriterien. Dazu gehören z.B.:

das Produkt ist für ein Release eingemeldet

das Produkt muss an einem definierten Übergabepunkt liegen

das so genannte *Produktblatt* muss beiliegen; das Produktblatt wird bei der Einmeldung vom PV in Stargate ausgefüllt, es enthält releaserelevante Attribute wie z.B.:

- Version
- Plattform / Kundenset
- Reboot (Ja/Nein) bei Installation nötig
- Abhängigkeiten (Software/Hardware), z.B. bestimmte Javaversion muss installiert sein
- Installationspfade
- Installationsabhängigkeiten (Reihenfolge)
- Konfigurationsparameter (User/Rechner)

- Sicherheitseinstellungen

Anschließend verschalt er das Produkt und bildet mit VB-Script die Installationshülle. Diese Installationshülle entspricht im Prinzip einer Konfigurationsdatei, in der die entsprechenden Parameter festgelegt werden. Die Paketierer dokumentieren ihre Änderungen in einer zugehörigen Log-Datei. Immer wieder entstehen dadurch Fehler durch Kopieren und Einfügen bzw. es wird die Dokumentation gänzlich vergessen. Im Falle von Fehlern bei der Verschalung ist für die überarbeitete Installationshülle auch eine eigene Versionierung nötig. Bis jetzt findet die Paketierung allerdings ohne Unterstützung eines Versionierungs- bzw. Konfigurationssystemes statt.

Im Rahmen des Projektes *(ReQuest)* soll in Zukunft als Installations- und Konfigurationsstandard Microsoft Windows Installer (MSI) [MSI] eingeführt werden. Komplett neue Produkte werden schon heute mit MSI konfiguriert.

Des Weiteren ist die Einführung einer Datenbank zur Konfigurationsunterstützung - Product and Release Integration Suite (PARIS) geplant. In ihr sollen die Versionsunterschiede der Konfigurationsdateien dokumentiert werden. Des Weiteren sollen in ihr Abhängigkeiten zwischen Produkten dokumentiert werden. PARIS nimmt so einen Teil der Funktionalitäten einer CMDB wahr, die es bis jetzt nicht gibt.

Das verschalte Produkt überspielt der Verschaler auf den Verschalungsserver bzw. den so genannten *Referenz-Code-Baum* oder *Ref-CID*. Die Zusammenstellung des Releases aus den übergebenen Softwareprodukten ist die Aufgabe des SI. Der SI erstellt ein weitere Konfigurationsdatei. Diese bündelt die jeweiligen Produktkonfigurationsdateien in einer bestimmten Reihenfolge. Somit entsteht die Installationsreihenfolge der Einzelprodukte durch Aneinanderreihung der jeweiligen Installationshüllen. Der Vorgang der Produktübergabe wird in Stargate durch den Status „Prod übergeben an IMPL “ des Produktbegleitblattes dokumentiert.

3.4.2.3 Testen und Abnahme des Releases

Die Testphase im dezentralen Bereich ist aufgeteilt in die IT-, QS-Phase und in eine PÜ- bzw. Pilot-Phase.

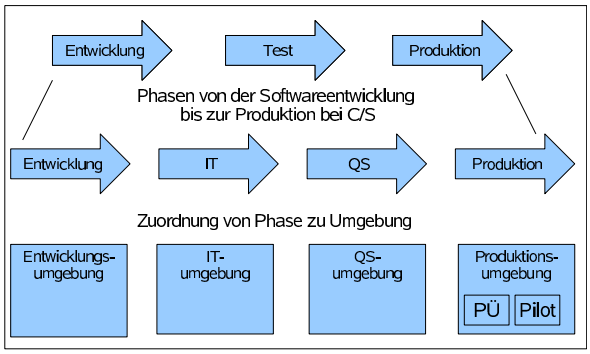


Abbildung 3.14: Zuordnung von Prozessphasen zu Umgebungen im dezentralen Bereich

Für die IT- und QS-Phase werden vom UV die Abnahmeumgebungen aufgebaut. Der Aufbau der PÜ erfolgt durch den SWV, eine echte Pilotumgebung ist ein Teilbereich der produktiven Umgebung. Abbildung 3.14 zeigt die Zuordnung von Prozessphasen zu den benötigten Umgebungen.

Die Termine für die Abnahmezeiten werden vom RM-C/S im Intranet im Rahmen der Release-Pläne veröffentlicht. Zu Beginn der jeweiligen Testphase versendet der RM-C/S zusätzlich eine Einladungsmail. Beim Update- und Voll-Release werden alle PV informiert. Im späteren Abnahmeprozess müssen dann auch für alle Produkte eines Kundensets Abnahmestatus in Stargate gesetzt werden. Dieser Status soll dokumentieren, ob der Test des jeweiligen Produktes erfolgreich war. Bei Sonderversorgungen und Hotfixes versendet der RM-C/S Einladungen an die PV, die ein Produkt eingemeldet. Nur für diese Produkte wird dann ein Abnahmestatus verlangt. Weitere Einladungen werden noch an diejenigen PV versendet, bei denen Abhängigkeiten ihrer Produkte zu den geänderten bekannt sind und an einen Interessentenkreis, der sich bei Änderungen fallweise entscheidet, ob er testen wollen oder nicht.

Die Notwendigkeit der Prüfung aller Produkte bei Voll- und Updatereleases ergibt sich, da es hier keine Unterstützung durch ein Konfigurationsmanagement gibt. Es ist vorab nicht bekannt, ob durch die Änderung der eingemeldeten Produkte unveränderte Bestandsprodukte beeinträchtigt werden.

Vom RM vorgegeben wird die Dokumentation der Tests durch Setzen des entsprechenden Stargate-Status durch den PV. Es kann hier aber nicht überprüft werden, ob der Status aufgrund eines vorher erfolgreichen Tests oder einfach nur so gesetzt wurde.

Testart	Tester	Durchführung	Bemerkung
Installationstest	Verschaler	am Verschalungs-arbeitsplatz	Produkteinzeltest
Installationstest	UV	IT	durch Installation des kompletten Releases in der IT-U
Installationstest	UV	QS	durch Installation des kompletten Releases in der QS-U
Installationstest	SWV	PÜ oder Pilot	PÜ optional
Integrationstest	PV, SI	IT oder QS	
Akzeptanztest	PV mit Fachbereich	QS	
funktionale Tests	PV	IT oder QS	
Schnittstellentest	PV	IT oder QS	entscheidet PV
Last- oder Performancetest		IT oder QS	optional, auf Kundenwunsch

Tabelle 3.4: Übersicht der Testarten, Testphase im dezentralen Bereich

ITIL fordert für die Durchführung der Release-Abnahme unabhängige Personen. Im dezentralen Bereich testen jedoch neben dem Fachbereich, UV und SI auch die PV ihre eigene Software.

Im Rahmen des Release-Prozesses werden verschiedene Tests durchgeführt. Tabelle 3.4 zeigt eine Übersicht der durchgeführten Testarten zusammen mit den Testern und der jeweiligen Phase im Release-Prozess. Die Testarten werden analog des Standards IEEE 610 [IEEE610] bezeichnet. Bei der HVBInfo sind hierfür oft mehrdeutige Begriffe, wie etwa Betreibertests oder technische Tests, im Gebrauch.

Bei allen Tests werden vom RM keine spezifischen Testinhalte vorgegeben. Die Testinhalte obliegen den Entscheidungen der jeweiligen Tester. Die Freigabe des kompletten Releases ergibt sich schließlich aus der Summe der Einzelfreigaben der Produkte.

3.4.2.4 Planung des Einsatzes

Nach ITIL beinhaltet die Einsatzplanung die Erweiterung der Release-Pläne um die konkreten Installationspläne. Diese Planung führt hier der SWV und nicht das RM durch. Der SWV bestimmt die Roll-out-Termine der jeweiligen Kundenlokationen. Die Release-Implementierung kann je nach Release-Größe und Anzahl der zu versorgenden Lokationen entweder als „big bang“ oder phasenweise erfolgen. Bei der Versorgung der Bankenfilialen z.B. erfolgt die Versorgung phasenweise. Oft wird zunächst ein so genannter *Vorpilot* am eigenen Standard versorgt, ehe die eigentlichen Pilotstandorte das Release erhalten. Nach Freigabe des Piloten durch den Kunden erfolgt der endgültige Flächen-Roll-out. Die jeweiligen Termine dazu veröffentlicht der SWV im Intranet.

3.4.2.5 Kommunikation, Vorbereitung und Schulung

Im Release-Prozess fallen verschiedene Aktivitäten bzgl. Kommunikation, Vorbereitung und Schulung an. Der RM-C/S nutzt als Kommunikationsmedium das Intranet, Mail- und Newsletterversand und Stargate. Damit werden alle relevanten Termine und Phasen abgestimmt und kommuniziert. So erhalten nachgelagerte Organisationseinheiten, wie etwa die SWV oder der Anwendungsservice die Information über bevorstehende Releases. Das Change-Management wird über die bevorstehenden Pilot- und Roll-out-Termine durch Eröffnen eines Pilot- oder Rollout-Changes involviert. Dieser Change dokumentiert geplante Veränderungen an produktiven Umgebungen.

Das EM koordiniert Vorbereitungsaktivitäten auf Seiten der Kunden. Das beinhaltet die Bereitstellung von fachlichen Informationen zu den geänderten Produkten im Intranet und auch mögliche Schulungsaktivitäten.

Der jeweilige PV ist für die Weitergabe von Wartungsdokumenten oder entsprechenden Benutzerhandbüchern zuständig.

3.4.2.6 Release-Verteilung und Installation

In Abbildung 3.15 ist der Weg der Softwareänderungen von der Entwicklungsphase bis zur Produktion dargestellt. Analog des bei Brinkkemper in Abschnitt 2.2.6.4.2 beschriebenen Weiterleitungsschemas werden die Produkte bzw. Releases von Übergabeserver zu Übergabeserver bzw. Umgebung zu Umgebung weitergeleitet. Die in Abschnitt 2.2.6.6 beschriebenen Deployment-Aktivitäten finden wie folgt statt:

Paketieren von Komponenten: siehe Abschnitt (3.4.2.2)

Release erstellen: siehe Abschnitt (3.4.2.2)

Konfigurieren von Releases: siehe Abschnitt (3.4.2.2)

Installieren von Releases: der SI kopiert manuell das Release auf den jeweiligen Übergabeserver; von dort erfolgt die Installation der Produkte auf der jeweiligen Hardware per HyperDB

Update und Entfernen von Komponenten: automatisch durch Konfiguration der Installationshülle des Releases

Aktivieren bzw. Deaktivierung von Komponenten : automatisch durch Konfiguration der Installationshülle des Releases

In ITIL ([OGC04]) ist die automatische Prüfung der Zielumgebungen vorgesehen. Die Überprüfung der IT-U und QS-U führt der jeweilige UV manuell anhand von Kriterien der so genannten *Umgebungscheckliste* durch. Diese beinhaltet Prüfkriterien wie etwa technischer User ist angelegt, korrekte Registryeinträge, Testuser kann sich anmelden, Laufwerksmapping ist korrekt, Outlook startet oder der Remote-Zugriff funktioniert. Diese Checkliste überprüft der UV und hakt das jeweilige Feld ab. Diese Checkliste wird nach der Durchführung abgeheftet.

Die Produktivumgebung wird vom Systemmanagement laufend überwacht. Hierfür werden verschiedene Monitoring-Werkzeuge, wie etwa *TIVOLI*, *NetView* für Windows und *PMon* für UNIX eingesetzt. Der Status der jeweiligen Hardware wird in den Ampelfarben im so genannten *PIZ-Alarm* oder *ARGUS* bei Host-Systemen visualisiert.

Der Aufbau der Umgebungen erfolgt mit Unterstützung der so genannten *HyperDB*. Die HyperDB ist eine Eigenentwicklung für die Softwareversorgung im Windows-Bereich. Die HyperDB ist eine Datenbank, in der Lokationen, Rechner und Releases verwaltet werden. Mit ihr soll eine produktive Umgebung nachgebaut werden. Tivoli ist in diesem Prozess für die Installation, d.h. Verteilung, der Releases zuständig und mit der HyperDB werden diese auf den jeweiligen Zielumgebungen aktiviert.

3.4.3 Beschreibung des Release-Steuerungsprozesses

Nachdem das Release im dezentralen Bereich aus verschiedenen Einzelprodukten zusammengestellt wird, beinhaltet der Release-Steuerungsprozess sowohl die Steuerung des Releases als auch die Steuerung von Aktivitäten an den Einzelprodukten. Des Weiteren steuert der RM-C/S die Testaktivitäten und Bereitstellung der kontrollierten Testumgebungen.

Abbildung 3.17 auf Seite 91 und 3.21 auf Seite 96 stellen die wichtigsten Aktivitäten im Release-Steuerungsprozess dar. Dieser Ablauf repräsentiert den Standardfall. Standard-Releases und Sonderversorgungen werden nach diesem Vorgehen abgewickelt.

Die wichtigsten Phasen des Release-Steuerungsprozesses sind analog den Meilensteinen des Release-Planes die Einmeldung, Verschalung, IT-, QS- und Pilotphase. Der Release-Steuerungsprozess wird von Stargate unterstützt. Zum Starten eines definierten Releases muss dieses in Stargate eröffnet sein. Nach Release-Eröffnung steuert und überwacht der RM-C/S den Release-Status und den Status der eingemeldeten Produkte. Nachfolgend werden die Aktivitäten zur Eröffnung eines Releases und zur Steuerung des Release- und Produktstatus

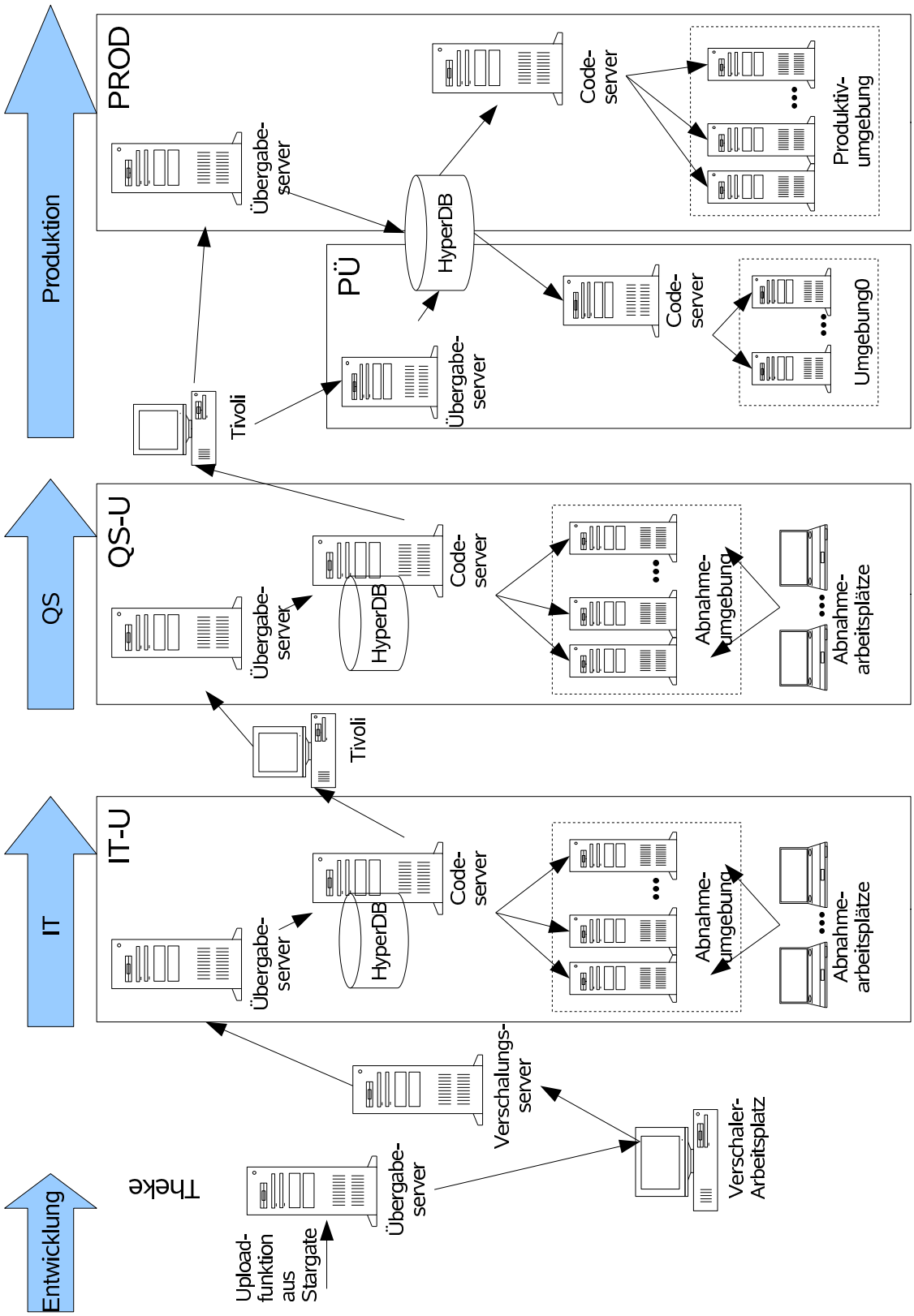


Abbildung 3.15: Logistikprozess bis zur Produktion

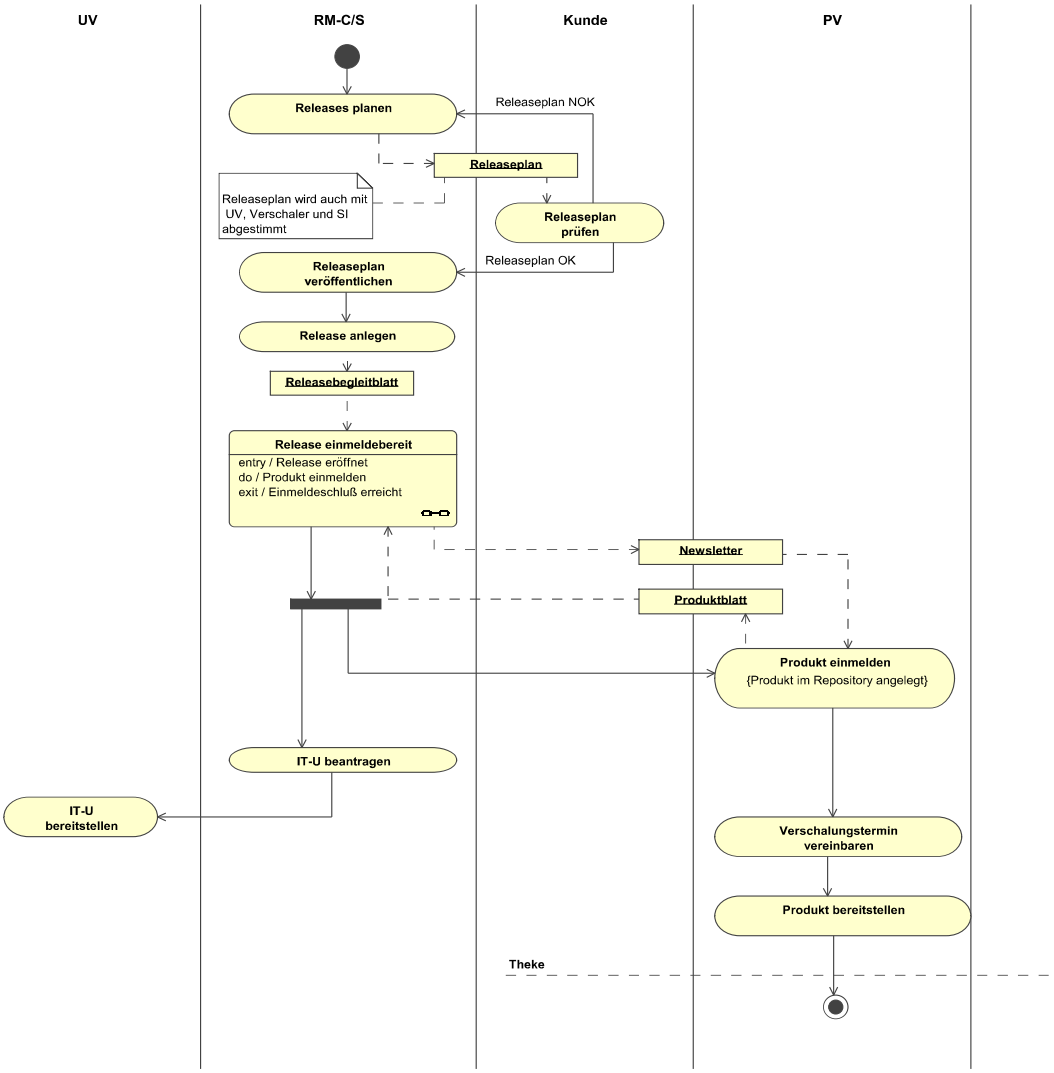


Abbildung 3.16: Ablauf des Release-Steuerungsprozesses bei der Planung

beschrieben. Ehe das Release in der produktiven Zielumgebung installiert werden kann, eröffnet der RM-C/S einen *Pilotchange* und einen *Roll-out-Change*. Diese Aktivitäten werden ab Abschnitt 3.6.3.3 beschrieben.

3.4.3.1 Release eröffnen

Mit Eröffnung eines Releases in Stargate wird der Durchlauf, d.h. der Abnahme- und Einführungsprozess, eines definierten Releases gestartet. Diese Eröffnung ist Voraussetzung für die spätere Einmeldung von Softwareprodukten durch den PV für eine Erstinstallation, Änderung oder Deinstallation in einer Zielumgebung.

Der RM-C/S vergibt hierfür im Tool Stargate einen Release-Namen. Die Releases werden ähnlich einer Dateiablage nach Jahresordnern strukturiert. Zu diesem Release legt der RM-C/S dann das so genannte *Release-Begleitblatt* und einen *Release-Plan* in Stargate an. Das Release-Begleitblatt definiert die Zustandsübergänge des Releases im Durchlauf. Abbildung 3.18 zeigt die Zustandsübergänge eines Releases im Release-Begleitblatt. Verantwortlich für die Vergabe des jeweiligen Status ist die in der Abbildung notierte Rolle. Die Rolle QS im Release-Begleitblatt entspricht dem Umgebungsverantwortlichen (UV) für die QS-Umgebung. Tabelle 3.5 ordnet den jeweiligen Status des Release-Begleitblattes zur Phase im Release-Steuerungsprozess zu.

Während des Durchlaufes wird nur dieser vorab angelegte Release-Weg durch das Tool unterstützt. Der je-

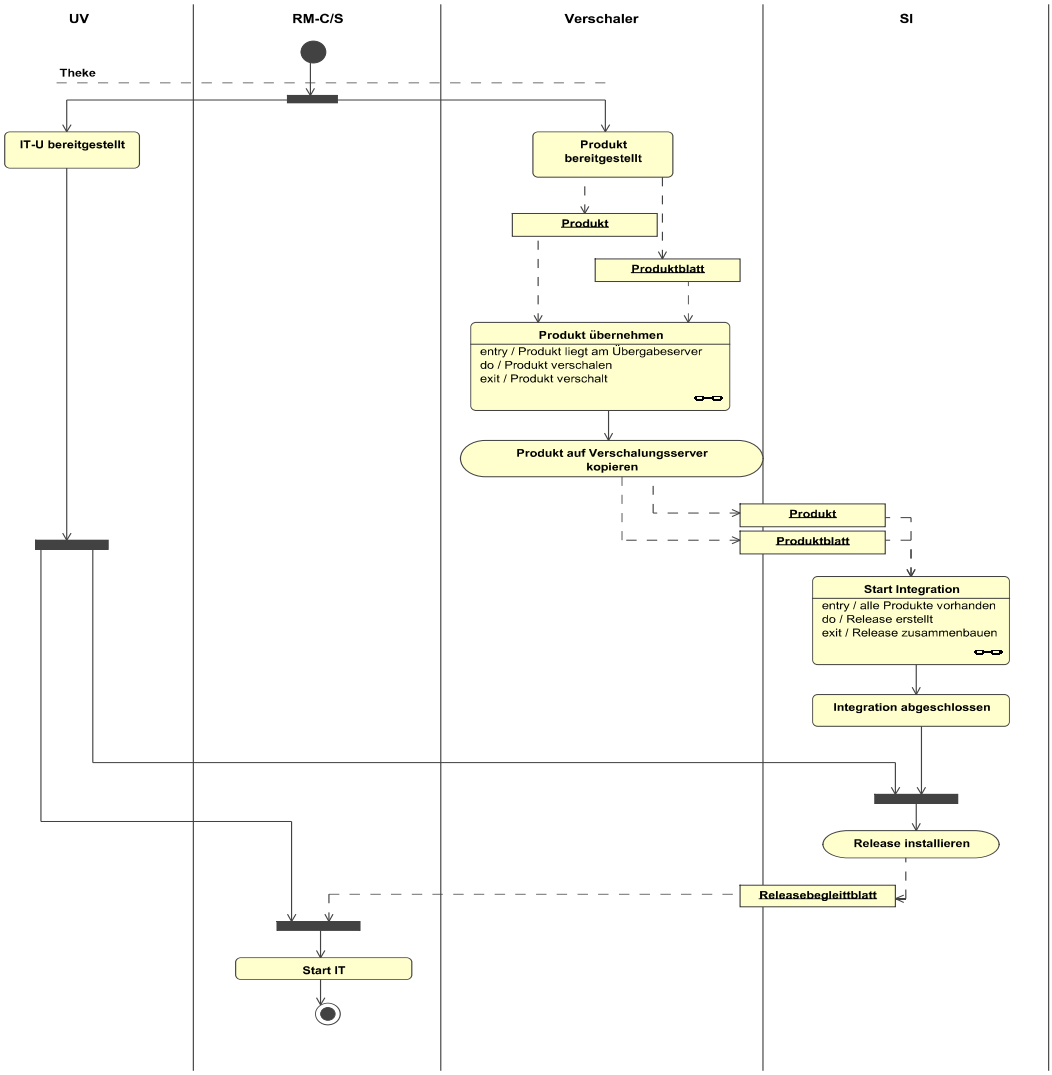


Abbildung 3.17: Ablauf des Release-Steuerungsprozesses bis zum Start der IT-Phase

weilige Status der entsprechenden Abnahmeumgebung (im Aufbau, aufgebaut, in Wartung, Störung etc.) wird hier nicht durch den Stargateworkflow unterstützt. Die Zustandsübergänge des Releases hängen von den Zustandsübergängen der zum Release gebündelten Einzelprodukte ab. Die Zustandsübergänge der Einzelprodukte werden im so genannten *Produktbegleitblatt* (siehe Abbildung 3.19) beschrieben.

Zum Release-Plan in Stargate wird derzeit lediglich der Endtermin der Einmeldephase aus der vorherigen Planung eingetragen. Dieser Termin steht in Zusammenhang mit den Einmeldungen der Release-Einheiten zum Release (siehe Abschnitt 3.4.3.2.1). Nach Einmeldeschluss können die PV keine Einmeldungen mehr vornehmen. Das kann dann nur noch über den RM-C/S erfolgen. Generell stehen dem RM-C/S zwei Arten von Release-Durchläufen zur Verfügung. Ein Standarddurchlauf und ein Hotfixdurchlauf.

Die PV werden vom RM-C/S über die Einmeldemöglichkeiten über regelmäßige Newsletter informiert. Diese Newsletter werden je Kundenset erstellt. Für die PV sind hier zunächst die Termine der Einmeldephase und Terminvereinbarung mit den Verschalern relevant.

3.4.3.2 Produkt- und Release-Status überwachen

Nachdem ein Release eine Bündelung von Einzelprodukten ist, steuert und kontrolliert der Release-Manager (RM-C/S) die Release- und Produktstatus im Release-Prozess. Die möglichen Produktzustände und Zustands-

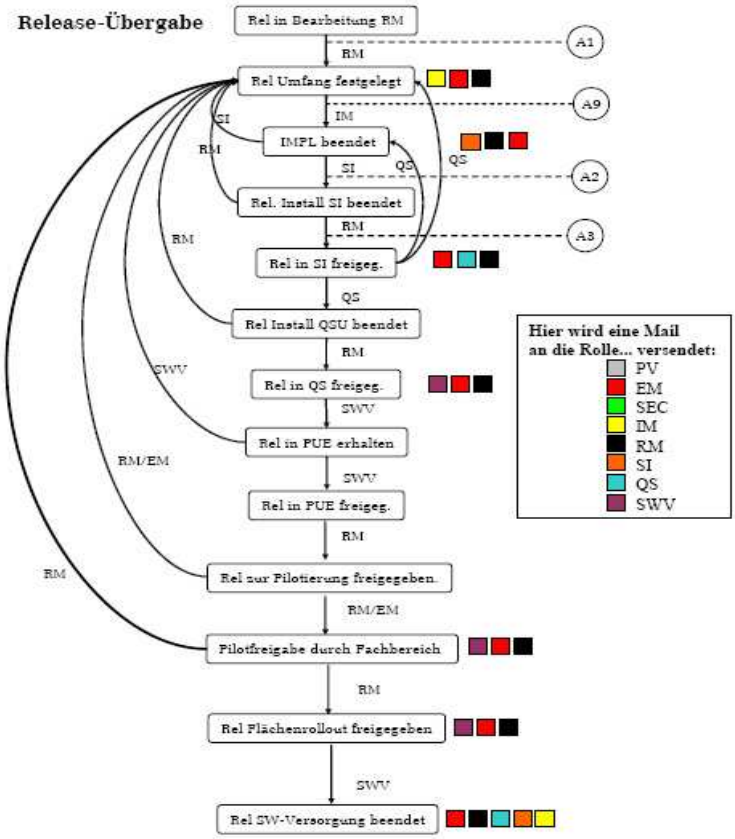


Abbildung 3.18: Zustandsübergänge des Releases definiert durch das Release-Begleitblatt

übergänge sind durch die Festlegungen im Produktbegleitblatt von Stargate festgelegt. Abbildung 3.19 zeigt die Zustandsübergänge eines Produktes im Release-Prozess.

Für die Koordination des Release-Durchlaufes benutzt der RM-C/S zusätzlich eine Checkliste. Diese Checkliste ist ein MS Worddokument. In ihr sind die Aktivitäten zur Orientierung des RM-C/S aufgelistet und können von ihm nacheinander abgehackt werden. Der jeweilige Release-Begleitblattstatus wird manuell von der in Abbildung 3.18 eingetragenen Rolle in Stargate gesetzt. Die Status des Produktblattes und des Release-Begleitblattes sind miteinander verknüpft. Wenn z.B. das Release-Begleitblatt den Status „Rel Install QSU beendet “ hat, kann das Produktblatt vom PV nicht mehr auf den Status „Prod in SI freigeg“ gesetzt werden. Allerdings ist diese Verknüpfung der Status nicht strikt. Auch wenn z.B. nicht alle Produkte sprich Produktblätter einen freigegebenen Status haben, kann der RM-C/S den Status des Release-Begleitblattes und damit das Release ohne Entfernung dieses Produktes in den nächsten Zustand setzten. Im Anschluss erfolgt die Beschreibung des Release-Steuerungsprozesses der Phasen Einmeldung, IT, QS und Pilot und Roll-out.

3.4.3.2.1 Einmeldephase In der Einmeldephase werden durch den PV Softwareänderungen zu den vorher durch den RM-/C-S eröffneten Release eingemeldet. Der EM überwacht den Verlauf der Einmeldephase. Der RM-C/S erinnert etwa zwei Wochen vor Einmeldeschluss die PV über den bevorstehenden Endtermin der Einmeldung per Newsletter. Der Hinweis für die Terminvereinbarung der Verschalung erfolgt vom RM-C/S im Newsletter ca. eine Woche vor Einmeldeschluss. Nach der Einmeldephase wird der Release-Umfang festgelegt.

Einmeldung durch den PV Die Einmeldung in ein Release ist der Vorgang durch den ein PV dem RM die Informationen über die Software mitteilt. Die Einmeldung erfolgt über die Einmeldefunktionalität von

Zustand Release-Begleitblatt	entsprechende Phase	Erläuterung
Rel in Bearbeitung RM	RM-C/S legt Release an	Vorbereitung eines definierten Release-Durchlaufes
Rel Umfang festgelegt	offizielle Einmeldung beendet	zum Release-Umfang gehört die Summe aller Einmeldungen
IMPL beendet	Ende der Verschalungsaktivitäten	ergibt sich aus Ende Verschalung je Produkt
Rel. Install SI beendet	Beginn der IT-Phase	das Release ist in der IT-U installiert
Rel in SI freigegeben	Ende QS-Phase	Ergibt sich aus Freigabe jedes einzelnen Produktes
Rel Install QSU beendet	Beginn QS-Phase	das Release ist in der QS-U installiert
Rel in QS freigeg.	Ende QS-Phase	Ergibt sich aus Freigabe jedes einzelnen Produktes
Rel in PUE erhalten	Start PÜ	Release ist an Produktionsumgebung übergeben
Rel in PUE freigeg.	PÜ	erfolgreicher Installationstest in PÜ
Rel zur Pilotierung freigegeben	Start Pilot	
Pilotfreigabe durch Fachbereich	Pilotphase	
Rel Flächen-Roll-out freigegeben	Flächen-Roll-out	
Rel SW-Versorgung beendet	Ende Roll-out	

Tabelle 3.5: Übersicht Status Release-Begleitblatt zum Status im Release-Steuerungsprozess

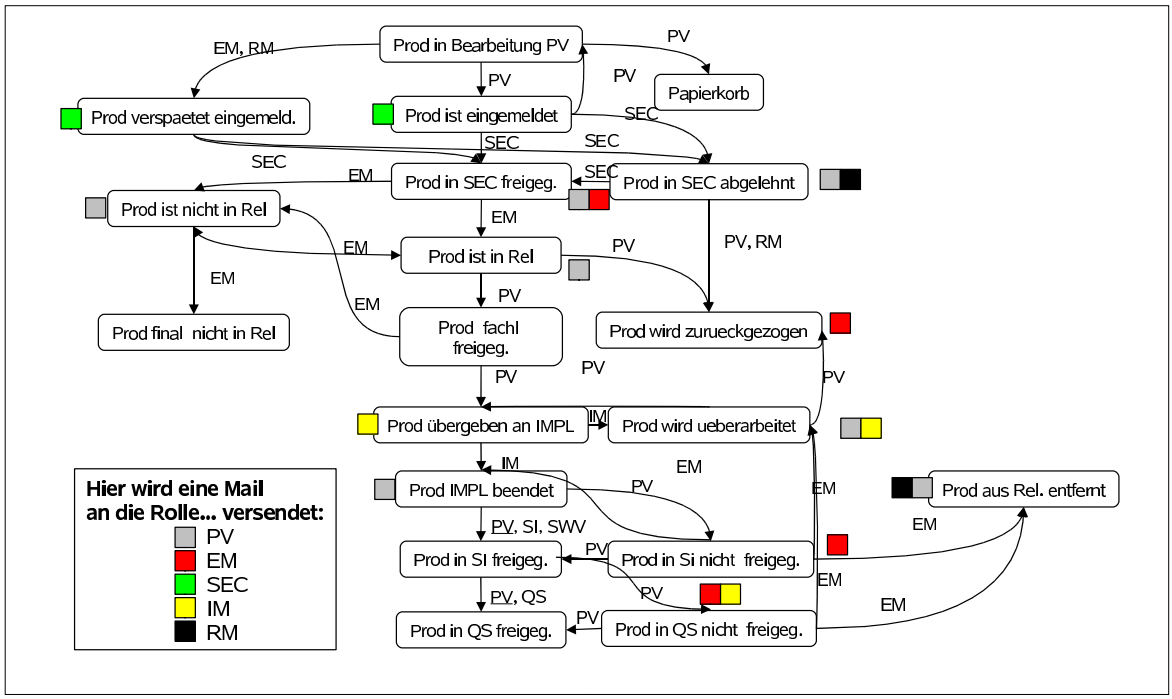


Abbildung 3.19: Zustandsübergänge eines Produktes definiert durch das Produktbegleitblatt

Stargate. Dort legt der PV das *Produktblatt* mit den entsprechenden Informationen an. Im *Repository* muss er hierfür mit der Rolle Stargate-PV eingetragen sein. Das Produktblatt beinhaltet Attribute wie etwa Produktversion, Name des PV, fachlicher Auftragsgeber, Release-Bezeichnung zu dem das Produkt eingemeldet werden soll, Hersteller, Beschreibung der fachlichen Änderung, bekannte Fehlerquellen, Größe in MB, zentrale Server Komponente, falls das Produkt einen zentralen Server benötigt oder die Angabe, ob ein *Reboot* nach der Installation nötig ist.

Der Einmelder kann hierbei neue Produkte installieren, sein geändertes Produkt updaten oder komplett deinstallieren lassen. Voraussetzung für die Einmeldung ist der Eintrag dieses Produktes im *Repository* und der Eintrag der Rolle Stargate-PV für den PV ebenfalls im *Repository*.

Der Beginn und das Ende der Einmeldephase sind in den veröffentlichten Release-Plänen dargestellt. Eine Einmeldung nach Ende der Einmeldephase kann nur noch der RM-C/S oder der EM durchführen.

Release-Umfang festlegen Frühstens nach Ende der Einmeldephase kann der Release-Umfang festgelegt werden. Meist wird sogar erst nach Abschluss der Verschalungstätigkeiten kurz vor Beginn der IT-Phase der Release-Umfang definitiv festgelegt.

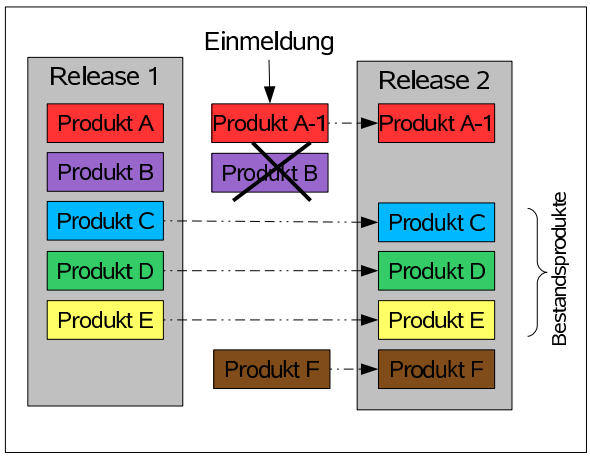


Abbildung 3.20: Release-Inhalt durch die Einmeldung

Abbildung 3.20 zeigt die Releasezusammenstellung durch die Einmeldung. Das Release 1 in der Abbildung ist das vorherige Voll-Release. Durch die Einmeldung werden Änderungen (Produkt A → A-1), Deinstallationen (Produkt B) oder Neuinstallationen (Produkt F) durch den PV eingemeldet. Für ein Updaterelease werden diese Änderungen zu einem Paket gebündelt. Für das Voll-Release werden zu den unveränderten so genannten *Bestandsprodukten* (Produkte C bis E) diese Änderungen hinzugefügt.

Üblicherweise werden alle eingemeldeten Änderungen zugelassen. Eine Ausnahme entsteht, wenn technische Zwänge dies verhindern, wie etwa das Überschreiten der maximal versorgbaren Größe. Dann entscheidet das EM über den Release-Umfang aufgrund von fachlichen Anforderungen. Dieser Schritt wird in Stargate vom RM-C/S durch manuelles Setzten des Status „Rel Umfang festgelegt“ dokumentiert.

Mithilfe der Einträge in Stargate können alle Neuprodukte extrahiert werden. Diese Auflistung sendet der RM-C/S an das Lizenzmanagement.

3.4.3.2.2 Implementierung In der Implementierungsphase erfolgen Verschalungstätigkeiten. Der SI enthält aus Stargate eine Liste aller eingemeldeten Produkte. Anhand dieser erfolgt ein Abgleich der Terminvereinbarungen mit den PV zur Verschalung. So kann überprüft werden, ob alle PV Verschalungstermine vereinbart haben. Mit den Verschalern und den UV werden dann die Abnahmetermine und ggf. Termine für Nachlieferungen festgelegt. Der RM-C/S informiert die Security, die zS- und Host-Verantwortlichen über Newsletter über die Abnahmetermine.

Verschalungsterminvereinbarung und Bereitstellung der Produkte durch den PV Der PV ist verantwortlich für die Terminvereinbarung mit der Verschalung. Er stellt weiterhin sein Produkt an einem definierten Übergabeserver bereit. Die Übergabestruktur hierfür ist definiert.

Verschalung und Release-Zusammenbau Nach Bereitstellung des Produktes durch den PV erfolgen die Verschalungs- und Konfigurationstätigkeiten. Diese wurden in Abschnitt 3.4.2.2 bereits beschrieben.

3.4.3.2.3 IT-Phase Bei jedem Release ist bei den Abnahmen in der IT-U und QS-U der so genannten *Infopoint* mit einem Release-Manager besetzt. Der Infopoint ist die Anlaufstelle für die Tester im Abnahmeprozess. Er ist mindestens mit dem RM-C/S besetzt und dieser dokumentiert Störungen und Fehler in einer Excel-Liste. Die Abnahmen werden durch einen Verschaler und UV unterstützt. Beeinträchtigungen der Abnahmeumgebungen werden vom RM-C/S über die *Abnahmeumgebungsstatus-Ampel (ASU-Ampel)* im Intranet veröffentlicht (siehe Abschnitt 3.4.3.2.3). Sie zeigt den Status der jeweiligen Abnahmeumgebung in den Ampelfarben.

Jedes eingemeldete System benötigt weiterhin eine Security-Freigabe. Diese sollte bis zum Beginn der Verschalungstätigkeiten erfolgt sein. In der IT-Phase erfolgen Tests durch den PV und den SI. Die Einladung der PV erfolgt durch den RM-C/S per Newsletter. Sowohl bei Update- als auch bei Voll-Releases werden alle PV zu Abnahmen eingeladen, deren Produkte im jeweiligen Kundenset sind. Lediglich bei Sonder-Releases oder Hotfixes werden nur die PV der eingemeldeten Produkte zu den Abnahmen eingeladen. Das Release bzw. die einzelnen Produkte werden auf Installierbarkeit, Lauffähigkeit in der Umgebung und auf Verträglichkeit mit anderen Produkten getestet. Eine erfolgreiche Abnahme des Produktes durch den PV in dieser Phase wird von ihm in Stargate durch Setzen des Produktblattstatus auf „Prod. in SI freigegeben“ (Anmerkung: SI steht hier für Systemintegrationsumgebung) dokumentiert. Der RM-C/S und auch der EM wertet diesen Status bei den Abnahmen regelmässig aus. Gegen Ende der Abnahmen werden alle PV, welche diesen Status noch nicht gesetzt haben, per Mail vom RM-C/S an ihre Aufgabe erinnert. Falls ein Produkt keine Freigabe durch den PV erhält, entscheidet der RM-C/S mit dem EM und PV das weitere Vorgehen. Mögliche Aktivitäten sind hier das Entfernen des Produktes aus dem Release, das Zulassen der Nachlieferung einer neuen Version oder das Stoppen des kompletten Release-Durchlaufes.

ASU-Ampel Mithilfe der ASU-Ampel publiziert der RM-C/S den Status der Abnahmeumgebung im Intranet. Er sammelt dazu Störungen und Informationen über Beeinträchtigungen der Abnahmeumgebung von den UV oder PV. Das können etwa Fehler sein, die sich auch auf die Abnahmen von anderen Produkten auswirken. Isolierte einzelne Produktfehler werden nicht in der ASU-Ampel aufgenommen. Der Status wird hierbei veröffentlicht für den Gesamtstatus der Abnahmeumgebung, die Möglichkeit von Remoteabnahmen (Anmerkung: es gibt extra Abnahmeplätze vor Ort bei der HVBSystems), den Status von für die Abnahme eingerichteten Testuser, den Status der Host-Anbindung, den Status von Schalter-Kasse-Systemen und weitere Funktionalitäten wie etwa Drucker. Analog den Ampelfarben symbolisiert ein grüner Status keine Störungen, ein gelber Status Einschränkungen und ein roter Status, dass keine Abnahmen möglich sind. Setzt der RM-C/S den Status auf grau, so sind ihm keine Statusinformationen bekannt.

3.4.3.2.4 QS-Phase Nach Ende der Abnahmen in der IT-Phase wird das Release in die QS-U installiert. Der Release-Manager dokumentiert den Beginn dieser Phase durch Setzen des Stargate-Status auf „Rel Install QSU beendet“ . Nach Installation des Releases in der QS-U durch den UV setzt dieser den Stargate-Status auf „Rel in QSU beendet “ . Der RM-C/S lädt dann per Mail die PV für die Abnahme in der QS-Phase ein. Laut den Dokumentationen erfolgt in dieser Phase auch die Abnahme durch den Kunden. Allerdings wird diese in der Praxis häufig in die IT-Phase vorverlegt, damit im Fehlerfall noch mehr Zeit für Korrekturmaßnahmen bleibt. Der PV dokumentiert eine erfolgreiche Abnahme seines Produktes durch den Stargate-Status „Prod in QS freigegeben “ . Der RM-C/S wertet die Status aller Abnahmen aus. Wenn der Status aller Produkte freigegeben ist und keine weitere Probleme vom UV gemeldet werden, setzt der RM-C/S den Release-Status in Stargate auf „Rel in QS freigegeben“ .

3.4.3.2.5 Produktionsübergabe, Pilot und Roll-out Am Ende der QS-Phase wird bei erfolgreichen Abnahmen das Release für die PÜ freigegeben. Das ist die so genannte Produktionsübergabe. Abbildung 3.22 zeigt die Aktivitäten in der Pilotphase. Der RM-C/S erstellt einen *Change* (siehe 3.6.3.3) für die Pilotierung, die Softwareübertragung und den Flächen-Roll-out. Der Stargate-Status des Releases wird dann auf „Rel in PUE freigegeben“ gesetzt.

Der EM hat die Übersicht über alle fachlichen Änderungen. Er ist deshalb i.d.R. für die Dauer der Pilotphase vor Ort beim Kunden, um ihn hier zu unterstützen. Nach erfolgreichen Pilottest erfolgt der Flächen-Roll-out

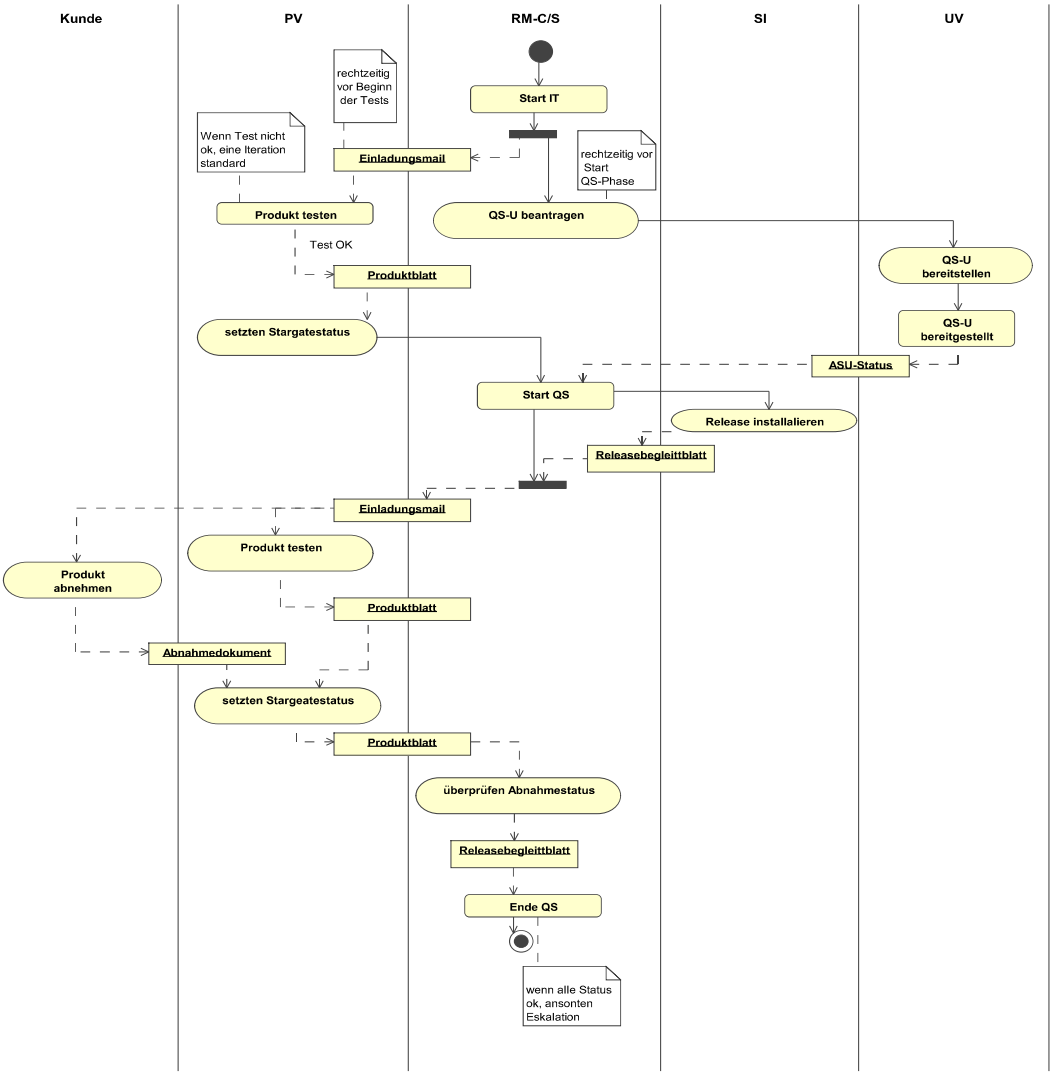


Abbildung 3.21: Ablauf des Release-Steuerungsprozesses in der IT- und QS-Phase

durch die SWV. Die jeweiligen Installationstermine sind im Verteilungsplan dokumentiert. Wenn der Flächen-Roll-out abgeschlossen ist, setzt der SWV den entsprechenden Status in Stargate. Das RM kann dann den Release-Durchlauf bewerten (siehe Abschnitt 3.4.4).

3.4.3.3 Change eröffnen

Der RM-C/S eröffnet im Tool *IMPACT* so genannte *Changes*. IMPACT wird von der Firma ASG Software Solutions [asg] entwickelt. Zur Anlage des Changes werden vom RM-C/S u.a. Art des Changes (z.B. SW-VERSORG), erwartete Auswirkung (Low, Medium, High), betroffene Umgebung (PROD), Datum der Changedurchführung oder verschiedene Anmerkungen in freier Textform eingetragen: Der neu angelegte Change hat zunächst den Status „Initial“. Durch setzten des Status auf „Open “ wird der Change wirksam. Durch die Genehmigung aller vorher festgelegten *Approver* wird dieser Change in den Status „Accepted“ gesetzt. Approver des Changes sind am Prozess beteiligte Stellen und vom Roll-out betroffene Stellen, z.B. Mitarbeiter des Anwendungsservices oder der SWV. Wenn der Change den Status „Accepted“ hat, kann die Softwareversorgung durchgeführt werden.

Falls es eine Pilotphase gibt, wird zunächst der so genannte *Pilotchange* eröffnet. Für den Roll-out wird der so genannte *Roll-out-Change* eröffnet. Dieser Change wird rechtzeitig vor Beginn des Piloten bzw. Roll-outs

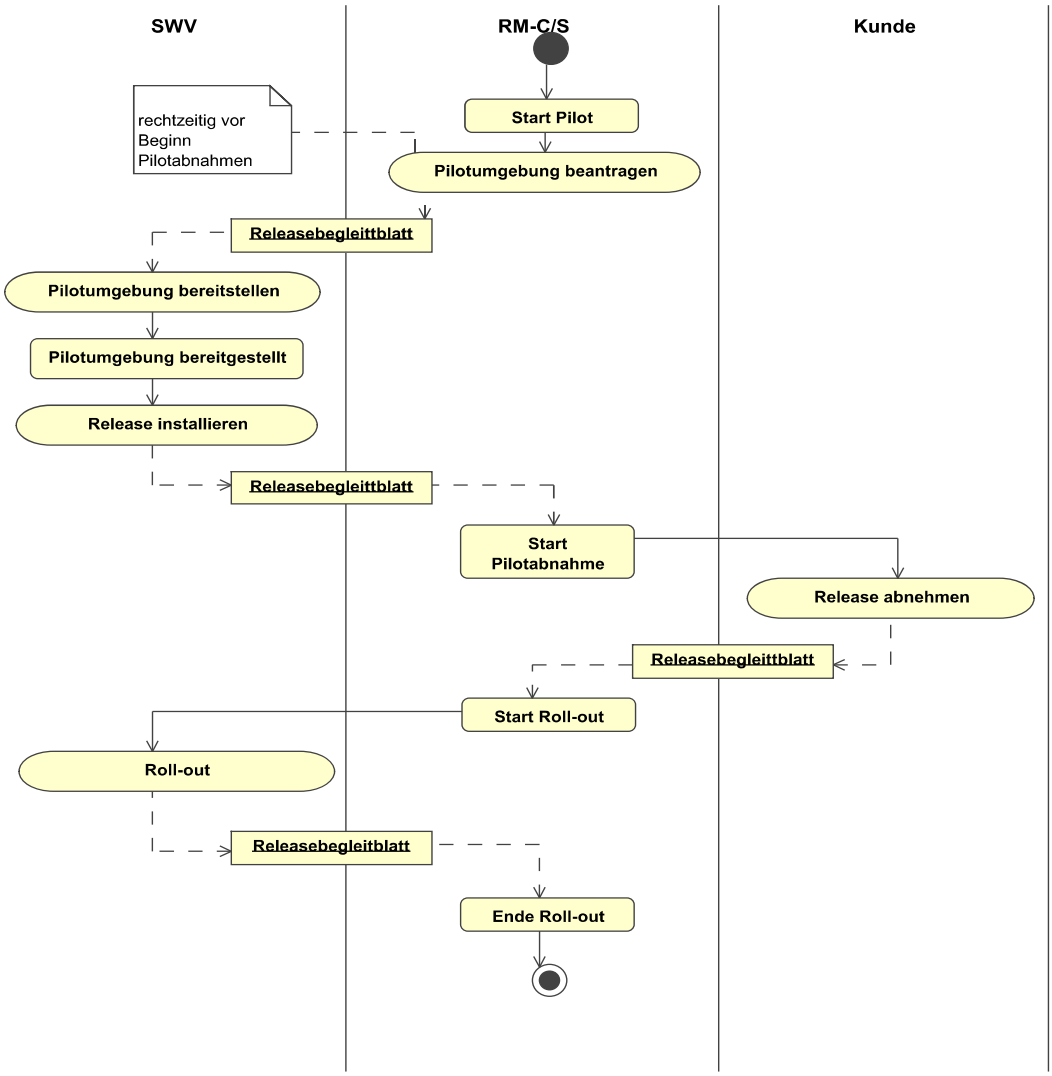


Abbildung 3.22: Ablauf des Release-Steuerungsprozesses ab Start Pilot-Phase

eröffnet. Der Change dient zum einen der Dokumentation einer Änderung an der Produktivumgebung des Kunden. Zum anderen wird anderen Organisationseinheiten angezeigt, dass sie in der Zeit in der das Release ausgerollt wird bestimmte Aktivitäten aussetzen sollen, z.B. Netzumbauten oder Unterbrechungen der Stromversorgung.

Die vom Tool automatisch vergebene Change-Nummer wird mit der Zuordnung zur Release-Nummer im Intranet veröffentlicht. Auf Anfrage des PV leitet sie der RM-C/S per Mail oder telefonisch an diesen weiter. Der PV kann mit dieser Change-Nummer als Referenz seinen eigenen Vorgang (z.B. eigener Change oder Projekt) abschliessen.

3.4.4 Bewertung des Release-Durchlaufes

Der Durchlauf der Releases wird regelmässig bewertet. Auf monatlicher Basis erstellt der RM-C/S einen Statusbericht. Input für diese Aktivitäten liefern die Auswertungen und Dokumentationen vom vorangegangen Schritt Produkt- und Release-Status überwachen (siehe Abschnitt 3.4.3.2). Mit Auswertungen aus dem Release-Prozess erstellt der RM-C/S eine Zusammenfassung der Ergebnisse und Erkenntnisse aus dem Release-Durchlauf. Dadurch können Aussagen getroffen werden bezüglich Termintreue, Release-Größe, An-

zahl Nachlieferungen aufgrund von Fehlern oder verspäteten Einmeldungen, Fehleranzahl je Phase (siehe z.B. Abbildung 3.23), Fehleranzahl je Fehlerart oder Anzahl der Hotfixes.

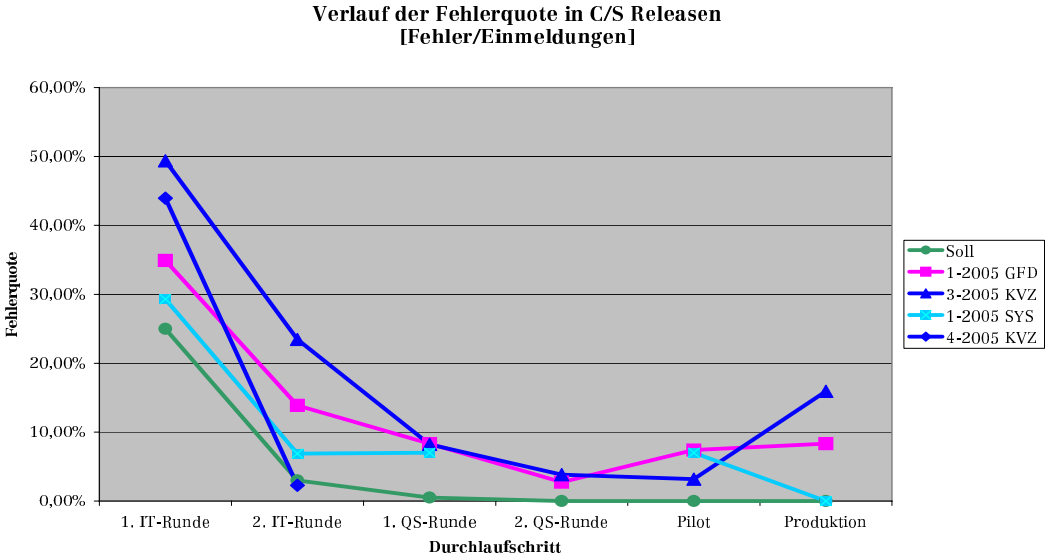


Abbildung 3.23: Fehlerverlauf im Release-Prozess

Die Kennzahlen werden mithilfe von Excel aufbereitet und im Service Information Management-Portal (SIM-Portal) veröffentlicht. Der Link zu dieser Veröffentlichung wird an den entsprechenden Verteiler versendet. Diese aufbereiteten Kennzahlen bilden die Grundlage für Verbesserungsvorschläge im Rahmen des *kontinuierlichen Verbesserungsprozesses (KVP)* .

3.4.5 Einordnung in das Lebenszyklusmodell

Das RM im dezentralen Bereich war früher bei der HVBSystems angesiedelt. Der Fokus lag wie jetzt im Bereich zS (siehe Abschnitt 3.5.5 auf Seite 112) auf der begleiteten Abnahme- und Einführungsphase und den entsprechenden Deployment-Aktivitäten. Das entspricht der Einordnung in die SWE-Sichtweise. Durch eine Reorganisation wurde das dezentrale RM zur HVBInfo übertragen. Die SWE-Sichtweise hat sich allerdings nicht geändert. Es wurden lediglich die Prozessgrenze (Theke) verschoben. Das RM im dezentralen Bereich deckt die Phasen Integration und Test des Lebenszyklusmodells ab (siehe Abbildung 3.24). Die Release-Steuerung entsprechend der im Lebenszyklusmodell findet hier nicht statt.

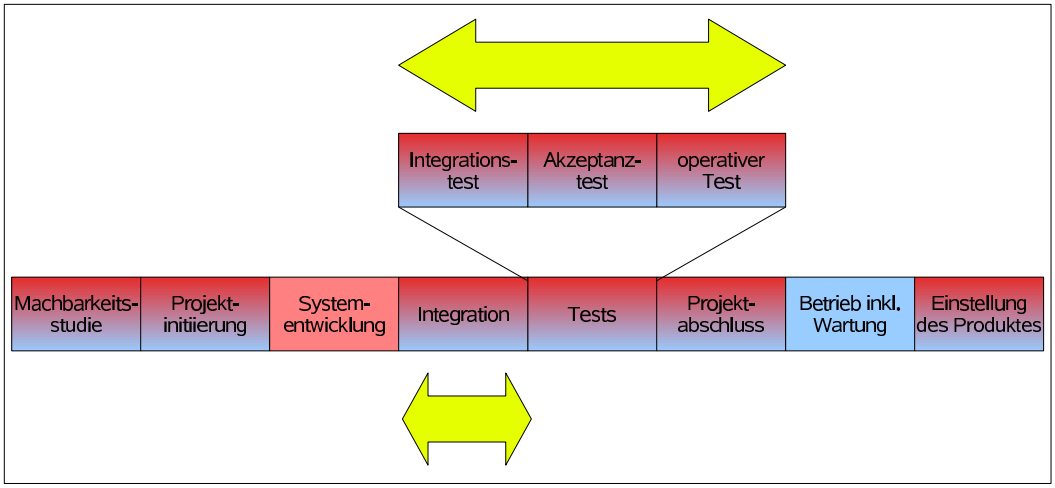


Abbildung 3.24: Einordnung des dezentralen RM in das Lebenszyklusmodell

3.5 Release-Management bei zentralen Servern

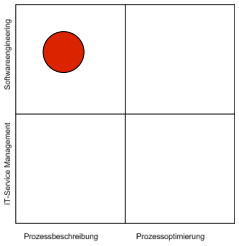


Abbildung 3.25: Einordnung RM zS

Im Bereich der zentralen Server (zS) erfolgt eine Unterscheidung zwischen Windows- und UNIX/Linux-Umgebungen. Im Bereich Windows zS erfolgen die Aktivitäten des RM für Produkte aus dem Core (Betriebssystem Windows 2000, Windows 2003 und den dazugehörigen Securitypatches) und für *HYPerDB*-versorgbare Produkte analog dem Release-Verfahren im dezentralen Bereich (siehe Abschnitt vorher 3.4). Der entsprechende Release-Manager wird hierfür zur Unterscheidung RM-zS genannt. Dieses Release-Verfahren wird hier nicht mehr beschrieben.

Für zS im UNIX bzw. Linuxbereich und für Windows-Applikationen, die nicht zum Core gehören, gibt es kein Release-Management aus Sicht der HVBInfo. Der Prozess zur Einführung von Software im Bereich zS entspricht dem im Abschnitt 2.2.6.8 auf Seite 33 beschriebenen Modells der Abnahme- und Einführungsphase. Er wird aus der Softwareentwicklung gesteuert.

In diesem Kontext ist das Release eine bestimmte Version eines Softwaresystems. Der Prozess zur Systemeinführung beschreibt Qualitätssicherungsabläufe vor und zur Einführung von entwickelter Software gegenüber dem Endanwender und gegenüber der HVBInfo als Betreiber. Bei der HVBSystems wird dieser Prozess nicht Release-Prozess sondern Produktionsübergabeverfahren mit den Teilprozessen Systemeinführung und Verifikation & Validierung genannt. Innerhalb dieser Systemeinführung werden Betriebsdokumente abgenommen und es erfolgen bestimmte Aktivitäten. Dieser Prozess ist im ALADIN dokumentiert.

Abbildung 3.26 zeigt die Schnittstellen zwischen dem Change-Management, der Softwareentwicklung und der Integration bzw. dem Betreiber. Im den nachfolgenden Abschnitten werden die beteiligten Rollen, ihre Aufgaben und der Release-Steuerungsprozess beschrieben.

3.5.1 Beteiligte Rollen und ihre Aufgaben

Nachdem hier für die Systemeinführung von Software kein Release-Prozess durch das RM definiert ist, steuert und koordiniert der jeweiligen Projektleiter eines Softwareentwicklungsprojektes den Ablauf. Dieser ist meistens ein Projektleiter von der HVBSystems. Falls die Softwareentwicklung im Rahmen von Wartungsaufgaben bestehender Software (Anwendungsbetreuung, Produktionssicherung) geschieht, steuert und koordiniert der jeweilige PV den Prozess. Für fremdentwickelte Software übernimmt der so genannten PV-light die PV-Aufgaben. Der Prozess zur Systemeinführung fremdentwickelter Software unterscheidet sich nicht zur

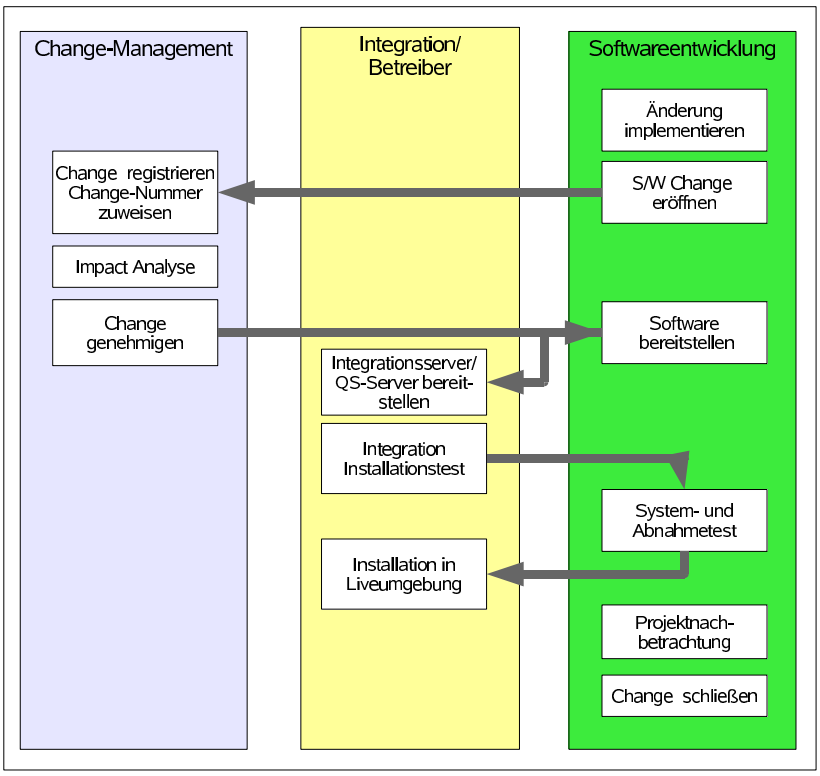


Abbildung 3.26: Übersicht Change-Management und Softwareentwicklung im Bereich zS

Einführung von eigenentwickelter Software. Aus diesem Grund wird nachfolgend nicht mehr zwischen PV und PV-light unterschieden.

Für die Anwendungsbetreuung und Produktionssicherung bestehen Wartungsverträge innerhalb derer die PV bestehenden und bereits eingeführte Software weiterentwickeln. Die Auslöser können hier vom Endbenutzer gemeldete Incidents, vom Problemmanagement gemeldete Problemreports oder so genannte *AB-Tickets* sein. AB-Ticketing ist das Incident-System der HVBSystems. Der Prozess der Systemeinführung und der Verifikation & Validierung ist für Entwicklungsprojekte und Anwendungsbetreuung bzw. Produktionssicherungsvorhaben identisch.

Zum Produktionsübergabeverfahren kommuniziert und interagiert der Projektleiter bzw. PV mit Auftraggebern dem Betreiber , Change-Manager , Designer, Endanwender, Entwickler, Fachbereichsmitarbeiter, Fachverantwortlicher, First-Level-Support, Qualitätssicherer, Produktverantwortlicher, Projektmanager, Reviewpartner, Systemintegrator , Teilprojektleiter, Hierbei sind lediglich die Rollen Betreiber, Systemintegrator und Change-Manager bei der HVBInfo angesiedelt. Die restlichen Rollen, bis auf Endanwender, Fachbereichsmitarbeiter und Auftraggeber, werden von Mitarbeitern der HVBSystems wahrgenommen. In Tabelle 3.6 sind die Rollen mit ihren Hauptaufgaben im Produktionsübergabeprozess aufgeführt.

Rolle	Aufgaben	Bemerkungen
Auftraggeber	• prüft Benutzerhandbuch u. Notfallkonzept • fachliche und technische Abnahme	siehe Abschnitt 3.5.2.3
Betreiber	• operatives Betreiben der übernommenen Systeme • durchführen von Tests	siehe Abschnitt 3.5.2.3

Rolle	Aufgaben	Bemerkungen
Change-Manager	• registrieren von Changes • genehmigen von Changes	
Designer	• unterstützt Abnahme verschiedener Dokumente	z.B. Systemarchitektur oder Sicherheitskonzept
Endanwender bzw. Benutzer	• arbeiten mit dem eingeführten System	
Entwickler	• realisieren die in den Anforderungsdokumenten beschriebenen Softwaresysteme	siehe Abschnitt 3.5.2.2
Fachbereichs-mitarbeiter	• unterstützt Entwicklungsprozess bei fachlichen Fragestellungen	z.B. bei Erstellung Arbeits-anweisungen
Fachverant-wortlicher	• vertreten der späteren Anwender • überprüfen der Berücksichtigung der fachlichen und organisatorischen Anforderungen	entspricht einem fachlichen Teilprojektleiter
First-Level-Support	• entgegennehmen von verschiedenen Dokumenten	z.B. Notfallkonzept oder Benutzerhandbuch
Qualitätssicherer	• unterstützt Entwicklungsprozess	z.B. bei Erstellung Einführungskonzept
Produktver-antwortlicher (PV)	• übernimmt das System zur Betreuung und Wartung • nimmt bei Wartungsprojekten die gleichen Aufgaben wie Projektleiter wahr	
Projektleiter	• träg Verantwortung für Entwicklungsprojekt • steuert und kontrolliert Systemeinführung • eröffnen von Changes	
Projektbera-tung	• berät zu verschiedenen Themen	z.B. Fragen zum Vorgehensmodell, IT-Sicherheit, Projektmanagement etc.
Reviewpartner	• überprüfen Meilenstein-Ergebnisse und Dokumente • dokumentieren Probleme und Änderungsanforderungen im Softwareentwicklungsprozess	üblicherweise projekt-fremde Fachleute; sie sind keine Mitglieder des aktuellen Entwicklungs-projektes

Rolle	Aufgaben	Bemerkungen
System-integrator (SI)	• definieren von Anforderungen der Zielumgebung • konfigurieren die Zielumgebung • integrieren übernommener Objekte zu einem Release • erstellen von Softwarepakete • durchführen von Integrationstests • durchführen von Installationstests	er kann sowohl bei der HVBInfo als auch bei der HVBSystems beschäftigt sein; die Integration von Sicherheitssystemen wird immer vom SI der HVBInfo vorgenommen
Teilprojektleiter	• unterstützt den Projektleiter bei großen Projekten	nachfolgend erfolgt keine Unterscheidung zwischen Teilprojektleiter und Projektleiter

Tabelle 3.6: Rollen und Aufgaben im Bereich zS

Der wesentliche Unterschied zum dezentralen Bereich besteht hier im Fehlen der Rolle UV. Diese Aufgaben werden im Bereich zS vom SI übernommen. Der SI ist hier sowohl für den Betrieb der Hardware als auch für die Integration und Installation der Software zuständig. Im Bereich zS benötigt ein AWS i.d.R. mindestens einen zentralen Server. Ausnahmen sind hier Servercluster oder Serverfarmen, wie sie etwa für Webanwendungen eingesetzt werden. Hier werden mehrere Anwendungen auf einer Serverfarm ausgeführt.

3.5.2 Release-Aktivitäten im Bereich zentraler Server

Abbildung 3.27 zeigt die Phasen zur Produktionseinführung von Änderungen im Bereich zS. In diesem Abschnitt werden die Aktivitäten der Systemeinführung anhand der ITIL Gliederung eingeordnet. Da der Prozess im Bereich zS allerdings aus der Softwareentwicklung gesteuert wird, gibt es nicht für jeden Punkt eine entsprechende Beschreibung.

3.5.2.1 Festlegen der Releasepolicy und des Release-Plans

Eine Releasepolicy und Release-Planung im ITIL-Sinne existiert im Bereich zS aufgrund der SWE-Perspektive nicht. Allerdings werden bestimmte Aspekte der ITIL-Releasepolicy bzw. des -Release-Planes im ALADIN-Vorgehensmodell unter dem Aspekt von Softwareentwicklungsprojekten beschrieben. Diese Aspekte werden nachfolgend erläutert.

3.5.2.1.1 Festlegen der Releasepolicy im Bereich zS

Nachfolgend werden die Aspekte einer ITIL-Releasepolicy beschrieben, welche im Bereich zS definiert werden.

Festlegung der Release-Einheiten Im Bereich zentraler Server erfolgt die organisatorische Handhabung der Release-Einheiten auf AWS-Ebene. In der technischen Abwicklung können allerdings auch nur Teilsysteme oder weitere Subkomponenten auftreten.

Release-Bezeichnung und -Nummerierung Ein Release entspricht hier im SWE-sinne einer bestimmten Softwareversion. Aus diesem Grund entspricht die Release-Bezeichnung dem Eintrag im *Repository* und die Release-Nummer entspricht der Softwareversionsnummer. Für die Versionierung werden bei der HVB-Systems vor allem im Microsoft-Umfeld noch mehrere verschiedene Versionierungssysteme eingesetzt. Ziel des Projektes *AgeNT* ist hier die Einführung eines zentralen Versionierungssystems. Es basiert auf dem Versionierungssystem von PVCS-Dimensions. PVCS Dimensions wird für die Versionierung im J2EE und UNIX-Bereich bereits eingesetzt.

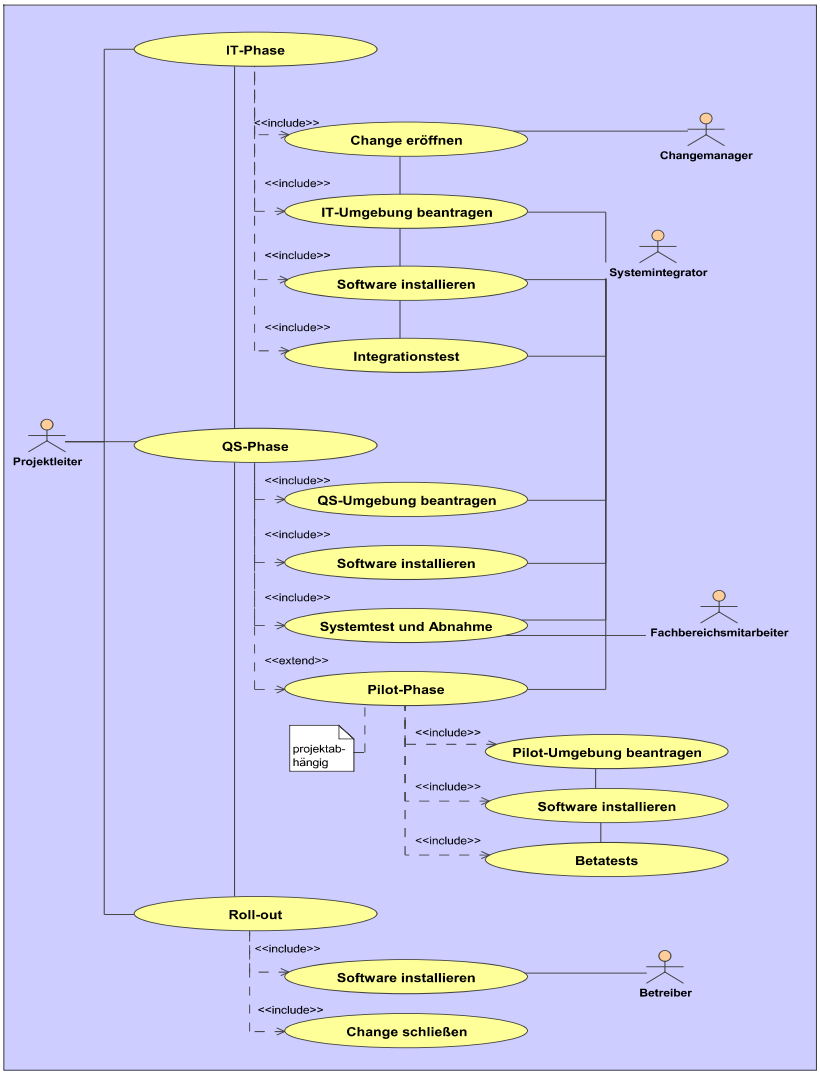


Abbildung 3.27: Übersicht der wichtigsten Phasen im Produktionsübergabeverfahren

Kategorie	Risiko, Auswirkung	Bemerkungen
0	Standard Änderung & kein RfC nötig	werden im SIM-Portal veröffentlicht
1	keine / geringe Auswirkungen/Risiko	
2	mittlere Auswirkungen/Risiko	
3	große Auswirkungen/Risiko	
E	Emergency Change	
I	Info Change	verursacht durch Dritte, z.B. Telekom

Tabelle 3.7: Kategorien und Auswirkungsklassifikation von Changes in Impact

Definition von Major- und Minor-Releases und einer Regel für Emergencyfixes Für die Begriffe Major- und Minor-Release gibt es im Bereich zS keine entsprechenden Release-Begriffe. Allerdings wird bei der Systemeinführung in die Produktivumgebung durch den Projektleiter ein Change in Impact eröffnet. Hier klassifiziert er bei der Eröffnung die mögliche Auswirkung entweder auf „low, medium oder high“. Des Weiteren erfolgt eine Einordnung des Changes in eine der in Tabelle 3.7 beschriebenen Risikokategorien. Ein Emergency Change entspricht in seiner Dringlichkeit einem Emergency Release. Er wird bei Beeinträchtigung oder Gefährdung des Service Levels oder auf Kundenwunsch durchgeführt.

Bezeichnung	verantwortliche Rolle
Sicherheitskonzept	Projektleiter
Einführungskonzept	Projektleiter
Schulungsunterlagen	Fachverantwortlicher
Fachliche und technische Abnahme	Qualitätssicherer
Systemübergabe	Projektleiter
Sicherheitskonzept	Projektleiter
Fachliches Mengengerüst	Fachverantwortlicher
Fachliche Testfälle	Fachverantwortlicher
Logisches Klassenmodell	Designer
Technische Testfälle	Qualitätssicherer

Tabelle 3.8: Auszug aus den erwarteten Begleitdokumenten aus ALADIN

Häufigkeit von Major- und Minor-Releases Nachdem im Bereich zS die Releases jeweils Entwicklungsvorhaben kennzeichnen, gibt es hierfür keine Festlegungen bezüglich der Anzahl von Major- und Minor-Releases.

Identifikation von geschäftskritischen Zeiten, in denen Implementationen vermieden werden sollen Bei der Einführung von zS-Systemen müssen ebenfalls die Zeiten von Frozen Zones berücksichtigt werden. Sie werden vom Kunden vorgegeben und werden vom Change-Management der HVBInfo bei den jeweiligen Change-Terminen überprüft.

Erwartete Begleitdokumentation für jeden Release-Typ Für jedes Softwareentwicklungsvorhaben ist eine Aufstellung der erwarteten Ergebnistypen im ALADIN dokumentiert. ALADIN bietet hierfür vorgefertigte Templates an. Des Weiteren ist in ALADIN für jeden Ergebnistyp festgelegt, wer der Ersteller ist, wofür der Ergebnistyp eingesetzt wird und bis zu welcher Projektphase er erstellt bzw. abgenommen sein muss. Bei den Einsatzarten wird in ALADIN zwischen Studien, Neuentwicklungsprojekten, Weiterentwicklungsprojekten, Infrastrukturprojekten, Consultingprojekten, Anwendungsbetreuung und Produktionssicherung unterschieden.

Zu jeder dieser Einsatzart bzw. Projekttyp sind dann die Ausprägungen der Phasen Projektinitialisierung, Analyse, Design, Realisierung, Systemeinführung, Projektmanagement und Verifikation & Validierung definiert. Abbildung 3.36 auf Seite 113 zeigt die Abdeckung des ALADIN-Vorgehensmodells mit dem Lebenszyklusmodell aus Abschnitt 2.1. Nachdem die Softwareentwicklung nicht im Fokus dieser Arbeit steht, sind hiervon lediglich die ALADIN-Phasen Systemeinführung und Verifikation & Validierung relevant. Sie entsprechen den Phasen Integration, Integrationstest und Akzeptanztest im Lebenszyklusmodell. Für operative Tests verweist das ALADIN-Vorgehensmodell auf die so genannten *Betreibertests*. Ihre Durchführung obliegt dem Betreiber, d.h. der HVBInfo.

Tabelle 3.8 zeigt einen Auszug aus ALADIN bzgl. der erwarteten Ergebnistypen für die Systemeinführung und Verifikation & Validierung.

Eine weitere Vorgabe der erwarteten Begleitdokumente ist in den so genannten *Production Readiness Kriterien (PRK)* aufgelistet. Diese Auflistung ist mit der HVBInfo abgestimmt. Abbildung 3.28 zeigt auszugsweise einen Teil der definierten PRK.

Richtlinien, wie und wo Releases dokumentiert werden sollen Diese Richtlinien sind wiederum in ALADIN dokumentiert. ALADIN ist gleichzeitig auch das Dokumentationswerkzeug.

Regel für Erstellen und Testen von Back-out-Plänen In ALADIN ist zum einen eine Vorlage für die eigentliche Installation und ein Notfallkonzept beim Auftreten von Fehlern beschrieben. Des Weiteren sieht es für das Einführungskonzept eine Beschreibung von Rückkehrverfahren im Notfall inkl. detaillierter

Nr	Production Readiness Kriterium (PRK)	Teilprozess (Beginn der Bearbeitung)	Meilenstein	Ergebnistyp / Aktivität / Nachweis	Führt aus	Ab-nahme durch
2b	Meilensteinergebnisse "Systemarchitektur und Technologie" sind abgenommen	Verifikation u. Validierung (Test & QS-Maßnahmen)	Systemarch+ Techn	ET Review	Reviewleiter, PL	PRT
5.	Abnahme und Freigabe ist erfolgt	Verifikation & Validierung (Test & QS-Maßnahmen)	Pilot-Eingang	ET Abnahmeprotokoll	PL	PV AG Betreiber
5a	Beginn Integration (IT)	Verifikation & Validierung (Test & QS-Maßnahmen)	IT-Eingang	ET Abnahmeprotokoll	PL	PV AG
5.1	Abnahme durch Betreiber ist erfolgt	Systemübergabe	Prod-Ready	AK Produktionsübergabe durchführen, ET Abnahmeprotokoll	s. Kompetenz-matrix im Aladin	Betreiber
5.1.1.1	Betriebshandbuch allgemein ist erstellt und abgenommen	Verifikation & Validierung (Test & QS-Maßnahmen)	Pilot-Eingang	ET Abnahmeprotokoll	PL	Betreiber
5.1.1.1.2	Service-/Supportzeiten für 2nd-Level Support definiert	Verifikation u. Validierung (Test & QS-Maßnahmen)	Pilot-Eingang	Abnahmeprotokoll Betriebshandbuch	PL	Betreiber
5.1.2.1	Die Bestellung (Hardware, Software, Netze, Dienste) wird rechtzeitig über den TPL HVBinfo ausgelöst.	Verifikation & Validierung (Test & QS-Maßnahmen)	Pilot-Eingang	Abnahmeprotokoll QS und Produktion	PL (Lead)	Betreiber
5.1.2.4	Das System hängt ausschließlich in Produktions-Domänen	Verifikation & Validierung (Test & QS-Maßnahmen)	Pilot-Eingang	Abnahmeprotokoll QS und Produktion	TPL Betreiber	Betreiber
5.1.2.5	Das System kann anhand übergebener standardisierter Pakete aufgebaut werden	Verifikation & Validierung (Test & QS-Maßnahmen)	Pilot-Eingang	Abnahmeprotokoll QS und Produktion	Integrator	Betreiber
5.1.3.1	Lasttest-Protokolle – mit Performance-Kennzahlen - liegen vom Kunden abgenommen vor	Verifikation & Validierung (Test & QS-Maßnahmen)	Pilot-Eingang	Abnahmeprotokoll QS Test	Integrator (Lead) mit Betreiber	Betreiber
5.1.3.8	QSU-Abnahme liegt vor (technisch sowie vom Fachbereich)	Verifikation & Validierung (Test & QS-Maßnahmen)	Pilot-Eingang	Abnahmeprotokoll QS Test	Integrator (Lead) mit Betreiber	Betreiber
5.1.4	K-Fall-Vorsorge ist getroffen	Verifikation & Validierung (Test & QS-Maßnahmen)	Pilot-Eingang	ET Abnahmeprotokoll	PL	Betreiber
5.1.4.7	Das System bezieht Strom aus mehreren, voneinander unabhängigen Phasen. Die Stromanschlüsse sind entsprechend gekegt.	Systemübergabe	Prod-Ready	Abnahmeprotokoll K-Fall-Vorsorge	TPL Betreiber	Betreiber
5.1.5.4	Recovery-Verfahren: Test durchgeführt, Testprotokolle verfügbar	Systemübergabe	Prod-Ready	Abnahmeprotokoll Test Pilotphase	TPL Betreiber	Betreiber
5.1.6.1	Schriftliche Abnahme der Pilotphase durch den Kunden und den PV	Systemübergabe	Prod-Ready	Abnahmeprotokoll Organisatorische Betriebsübernahme	PL	PRK-Team
5.1.6.4	Schulung in den Systemen für 1st Level und 2nd Level Einheiten ist erfolgt	TP Produktionsübergabe	Prod-Ready	Abnahmeprotokoll Organisatorische Betriebsübernahme	PL	Betreiber
5.1.6.7	SLA (Betrieb, Prosi, AB) mit Kunden liegt unterschrieben vor. Sondervereinbarungen zu Servicezeiten, Einschränkungen und Betriebsrisiken sind aufgenommen	TP Technisches Design	Prod-Ready	SLA	PL	Betreiber
5.2.1	Abnahme durch Auftraggeber (Kunde) in der Entwicklungsumgebung ist erfolgt	Realisierung	IT-Eingang	ET Abnahmeprotokoll	PL AG	AG
5.2.2	Benutzerhandbuch ist erstellt	Realisierung	Pilot-Eingang	ET Benutzerhandbuch	PL	AG

Abbildung 3.28: Auszug aus den PRK der HVBSystems

Prozessbeschreibung und notwendig einzuschaltende Schnittstellen vor. Der Projektmanager ist für die Umsetzung zuständig.

Zuständigkeiten des RM Für den Bereich zS ist kein RM definiert. Ausnahme sind eingangs beschriebene Corereleases im Windows-Bereich. Diese werden nach einer internen Umstrukturierung jetzt vom bestehenden Release-Management mit abgewickelt. Hierfür ist auch eine eigene Rolle Release-Manager für zentrale Server (RM-zS) festgelegt. Das Verfahren ist identisch dem im dezentralen Bereich beschriebenen.

Beschreibung des Release-Steuerungsprozesses Der Release-Steuerungsprozess entspricht hier der Steuerung des Teilprozesses Systemeinführung und Verifikation & Validierung durch den Projektleiter. Beides ist in ALADIN dokumentiert und wird in Abschnitt 3.5.3 näher beschrieben. Abbildung 3.29 zeigt den Zusammenhang dieser Teilprozesse. Hier gibt es ebenfalls die IT- und QS-Phase (vgl. IT- und QS-Phase im dezentralen Bereich, Abschnitt 3.4.3.2.3). Die in der Abbildung als Qualitätstore bezeichneten Markierungen definieren Abnahmekriterien. Diese müssen bis zum Abschluss der entsprechenden Phase abgenommen sein und sind in den PRK festgelegt. Die Zuordnung der wichtigsten Aktivitäten in diesen Phasen zeigt Abbildung 3.30.

Dokumentation der exakten DSL-Konfiguration Eine DSL ist aufgrund der SWE-Perspektive nicht definiert.

3.5.2.1.2 Release-Planen Die Planung eines Releases entspricht der Planung eines Softwareentwicklungsvorhabens im Rahmen eines Projektes. Diese Planung wird hier nicht betrachtet, da die Softwareentwicklung nicht Gegenstand dieser Arbeit ist. Im Rahmen der Systemeinführung wird der Schulungsbetrieb geplant. Die eigentliche Release-Einführung in die Liveumgebung des Kunden wird ebenfalls vom Projektleiter bzw. PV geplant und mit den Betroffenen abgestimmt.

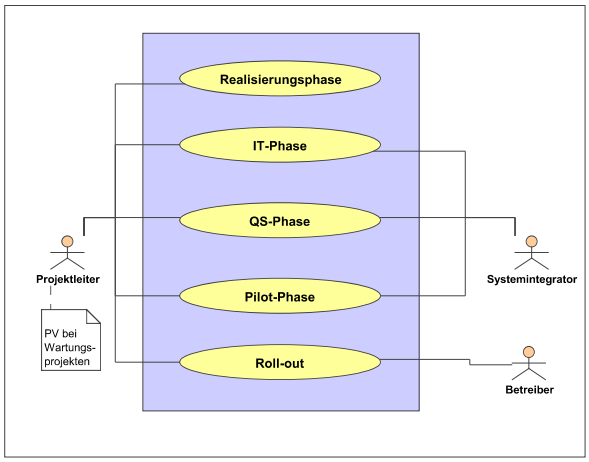


Abbildung 3.29: Zusammenhang Realisierung, Verifikation & Validierung aus ZAD HVBSystems

3.5.2.2 Entwurf, Release-Zusammenstellung und -Konfiguration

Ein Release entspricht einer bestimmten Softwareversion. Aus diesem Grund entsprechen der Entwurf, die Release-Zusammenstellung und Konfiguration reinen Softwareentwicklungstätigkeiten. Sie sind deshalb nicht im Fokus dieser Diplomarbeit und werden nicht weiter beschrieben. Bestimmte Konfigurationsaufgaben im Rahmen des *lokalen Customizing* (vgl. Einführungsphase bei Steinweg, siehe Abschnitt 2.2.6.8.2), wie etwa die Anpassung der Software an die Zielumgebung, werden im Rahmen der IT- und QS-Phase durch den SI durchgeführt (siehe Abschnitt 3.5.3).

3.5.2.3 Testen und Abnahme des Releases

Der Test- und Abnahmeprozess eines Releases im zS-Bereich erfolgt in der IT- und QS-Phase (siehe Abbildung 3.29). Aus dieser Abbildung wird der wesentliche Unterschied zum dezentralen Bereich sichtbar. Die Realisierungsphase reicht hier bis zum Ende der QS-Phase. Im dezentralen Bereich hingegen wird davon ausgegangen, dass die Realisierung bis zum Beginn der IT-Phase bereits abgeschlossen ist.

Dieses phasenweise Vorgehen bei den Testaktivitäten findet sich auch im V-Modell wieder (vgl. Abschnitt 2.2.1). Abbildung 3.31 zeigt die Testabstufungen im V-Modell vom Unit-Test über Integrationstest, Systemtest bis zum Abnahmetest. Im Testprozess der HVBSystems sind ebenfalls die Teststufen Modultest (anstatt Unit-Test), System-, Integration-, Abnahme- und Betreibertest beschrieben.

Die erste fachliche Abnahme erfolgt laut ALADIN durch den Auftraggeber und den PV nach den erfolgreichen Testen in der Entwicklungsumgebung. Dokumentiert wird diese Abnahme durch unterschriebene Abnahmeprotokolle. Diese Abnahme ist laut der Festlegung im Vorgehensmodell die Voraussetzung für die nachfolgende Übergabe an die Implementierung. Weil die Entwicklungsumgebung jedoch oft nicht den Ansprüchen nach produktionsnahen Bedingungen genügt bzw. die entsprechende Hardware nicht zur Verfügung steht, werden diese Abnahmen von der EU in die IT-U bzw. QS-U verschoben.

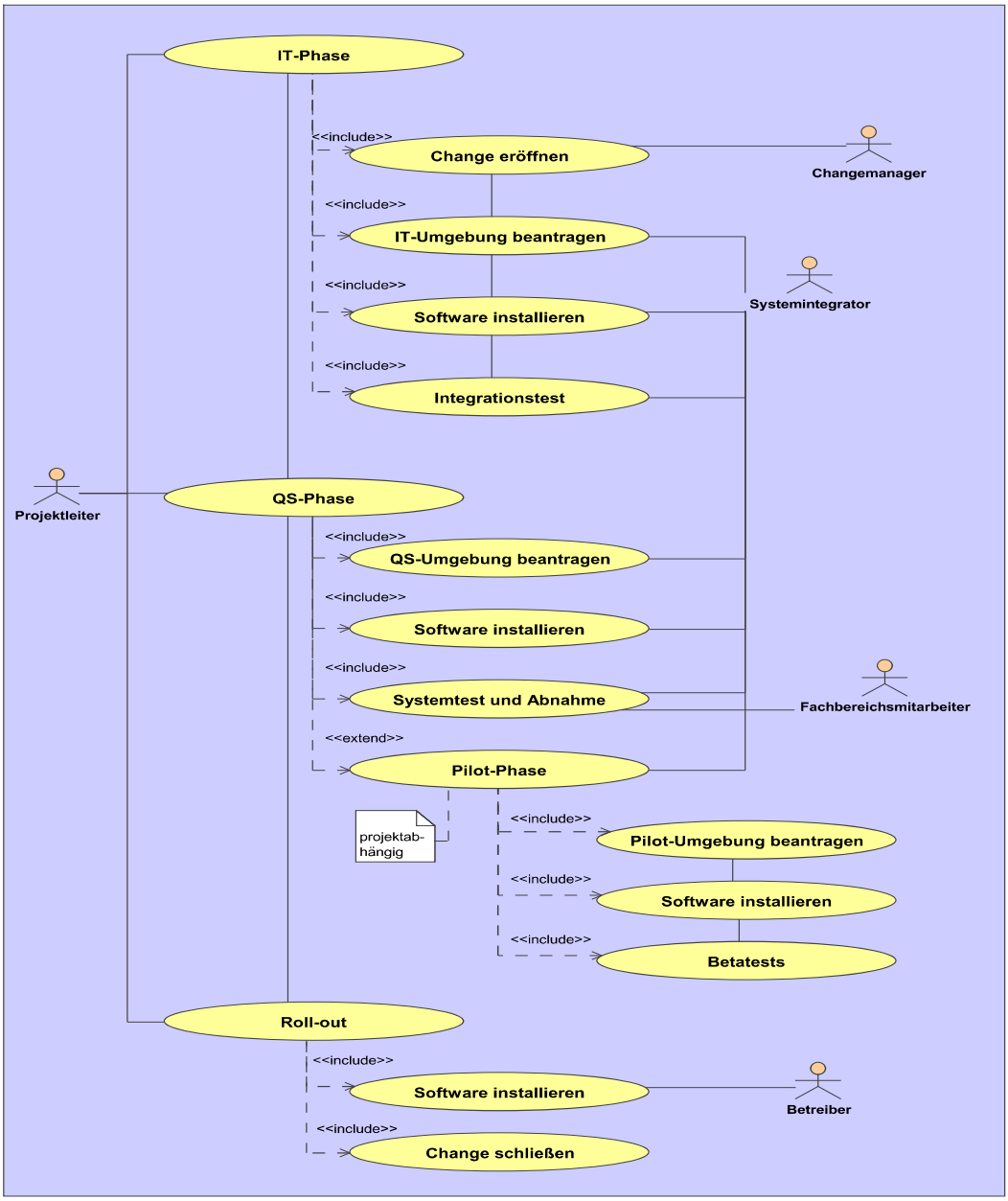


Abbildung 3.30: Übersicht der wichtigsten Phasen im Produktionsübergabeprozess

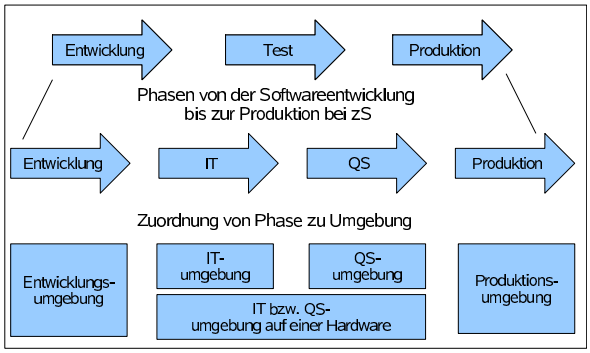


Abbildung 3.32: Zuordnung von Prozessphasen zu Umgebungen im Bereich zS

Abbildung 3.32 zeigt die Zuordnung von Prozessphasen zu den benötigten Umgebungen. Diese werden vom Projektleiter bzw. PV bei der HVBInfo angefordert. Er ist auch für die Beschaffung und Bereitstellung der spezifischen Hardware verantwortlich. Aus dieser Abbildung ist ersichtlich, dass die gleiche Hardware teilweise sowohl für die IT- als auch für die QS-Phase benutzt wird. Für manche Projekte hingegen steht jeweils eigene Hardware sowohl für die IT- als auch für die QS-Phase zur Verfügung.

Tabelle 3.9 zeigt die Übersicht der Testarten mit den dafür benötigten Umgebungen und durchführenden

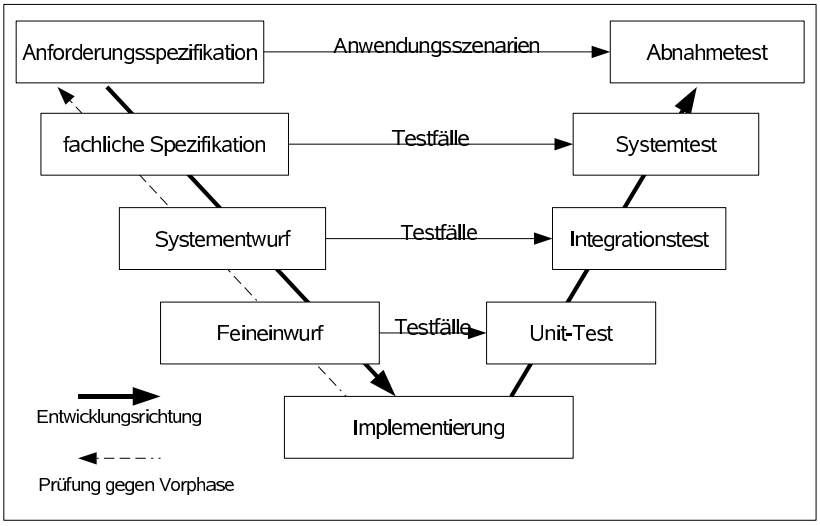


Abbildung 3.31: Testprinzip im V-Modell aus [WSK 05]

Testart	Tester	Umgebung
Modultest	HVBSystems	Entwicklungsumgebung (EU)
Systemtest	HVBSystems	EU
Integrationstest	HVBSystems	Test- und Abnahmeumgebung
Abnahmetest	Kunde	Test- und Abnahmeumgebung
Betreibertest	HVBInfo	Pre-Life-Umgebung

Tabelle 3.9: Übersicht der Testarten, Testumgebung und Tester im Bereich zS laut Testkonzept der HVBSystems

Testern laut Testkonzept der HVBSystems. Der Integrationstest im Modell der HVBSystems versteht sich als Test der Integration der Software in einer produktiven Systemlandschaft. Hier soll die Querverträglichkeit mit Nachbarsystemen oder Außenschnittstellen getestet werden.

Als Betreibertest werden alle Testaktivitäten durch den Betreiber, d.h. die HVBInfo verstanden. Hierzu zählen Installationstests in den Test- bzw. Abnahmeumgebungen, Deinstallationstests, die das rückstandslose Entfernen des Systems aus einem Gesamtsystem sicherstellen, Systemmanagementtests, d.h. die Überprüfung der Software in den Überwachungstools (z.B. Tivoli, Argus) und auf Kundenwunsch durchgeführten Performance- und Lasttests.

Die Betreibertest werden jeweils AWS- bzw. projektbezogen durchgeführt. Es gibt hierfür keine übergreifenden Vorschriften, was getestet werden soll. Die Dokumentation der erfolgten Test erfolgt dezentral, projekt- und produktbezogen. Teilweise kann die Testdurchführung nur über Logfiles nachvollzogen werden. Die Testmethoden wurden hier von einem Mitarbeiter auch als „rustikal “ bezeichnet.

3.5.2.4 Planung des Einsatzes

Im Rahmen der Systemeinführung wird vom Projektleiter bzw. PV das Einführungskonzept erstellt. Unterstützt wird er hierbei vom Fachbereichsmitarbeiter, SI, Qualitätssicherer und Betreiber. Das Einführungskonzept enthält die Grundstrategie zur Systemeinführung und damit verbundene Maßnahmen. Darin legt der Projektleiter fest, ob die Einführung etwa stufenweise, parallel oder als „big-bang“ erfolgt. Ein weiterer Bestandteil des Einführungskonzeptes ist das Migrationskonzept für die Belieferung mit Datenbeständen.

Der eigentliche Roll-out-Termin wird bei Eröffnung des Changes angegeben. Der Change-Manager prüft die Durchführbarkeit. Er kontrolliert z.B., ob es zum vorgesehenen Roll-out-Termin nicht zu Kollisionen mit ande-

ren Änderungen wie z.B. Netzumbauten oder Arbeiten an der Stromversorgung kommt. Changes ab Kategorie 2 (siehe Tabelle 3.7) werden zusätzlich noch von einem täglich zusammenkommenden Changeboard überprüft.

3.5.2.5 Kommunikation, Vorbereitung und Schulung

Die Planung von Unterstützungsmaßnahmen für die Einführung bzgl. Terminen und betroffener Mitarbeiter wird im Einführungskonzept vom Projektleiter beschrieben. Es enthält außerdem einen Schulungsplan. Der Schulungsplan beschreibt Schulungsverfahren, Umfang der Schulungen und Termine. Die eigentlichen Schlungungunterlagen werden vom Fachverantwortlichen erstellt. Der Projektleiter bzw. PV ist hier für die Koordination aller Aktivitäten verantwortlich.

3.5.2.6 Release-Verteilung und Installation

Der Projektleiter bzw. PV ist verantwortlich für die Erstellung einer Installationsanleitung für die zentralen Server. Unterstützt wird er hierbei von den Entwicklern, SI und Fachverantwortlichen. Der Betreiber muss dieses Dokument überprüfen.

Nach Genehmigung des Changes wird die Änderung durch den SI in die IT-U bzw. QS-U installiert. Für die Installation der Produktivsysteme gibt es teilweise eigene Verantwortliche, ebenfalls Betreiber genannt. Sie erhalten das Release vom SI und installieren es auf den produktiven Systemen.

Für die Verteilung und Installation werden verschiedene Verfahren eingesetzt. Diese unterscheiden sich je nach Serverbetriebssystem und dem zu installierenden System. Tabelle 3.10 zeigt die Übersicht der Versorgungs- bzw. Verteil- und Installationsverfahren. Die Verteilung und Installation per HYPerDB unterscheidet sich nicht zum dezentralen Bereich (vgl. Abschnitt 3.4.2.6). Das Staging-Verfahren ist in Abbildung 3.33 skizziert. Über einen zentralen Stagingserver erfolgt die Versorgung der jeweiligen IT-, QS- und Produktionsserver. *Tivoli* ist ein Produkt von IBM [IBM 06]. Es unterstützt den Deployment-Prozess bei Konfigurations- und Installationsaufgaben.

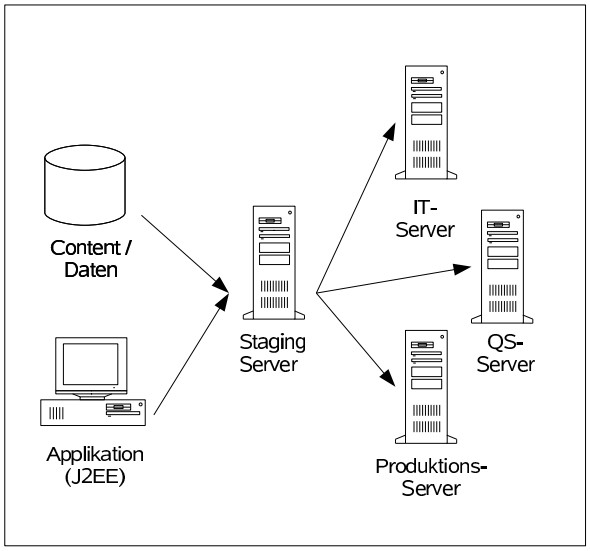


Abbildung 3.33: Darstellung der Softwareversorgung mit Staging im Bereich zS

3.5.3 Beschreibung des Release-Steuerungsprozesses

Der Projektleiter bzw. PV steuert und kontrolliert den Release-Steuerungsprozess. Abbildung 3.30 zeigt die Übersicht der wichtigsten Aktivitäten zur Erstellung und Produktionseinführung eines Releases. Die Aktivitäten der IT- und QS-Phase werden in Abbildung 3.34 dargestellt. Auslöser für den Beginn der IT-Phase ist der

zu installierendes System	zS Solaris	zS Linux	zS Windows
Content	• I: per Staging	-	• I: manuell
Applikation	• V: Staging, manuell oder TIVOLI-Configmanager • I: Skript aus Staging, PKG, Tivoli PKG	• V: Staging, manuell, TIVOLI-Configmanager • I: Skript aus Staging, PKG, Tivoli PKG	• V: Staging, manuell, HyperDB • I: Skript aus Staging, manuell, HyperDB
Middleware	• V: manuell, TIVOLI-Configmanager • I: PKG, Tivoli PKG	• V: manuell, TIVOLI-Configmanager • I: PKG, Tivoli PKG	• V: HyperDB • I: HyperDB
Datenbank	• V: manuell • I: PKG	• V: manuell • I: PKG	• V: manuell, HYPerDB • I: -, HYPerDB
Betriebssystem	• V: manuell, TIVOLI-Configmanager • I: PKG, Tivoli PKG	• V: manuell, TIVOLI-Configmanager • I: PKG, Tivoli PKG	• V: NDM+HYPerDB • I: HYPerDB
Installation (I) , Versorgung (V) , Paketmanager (PKG) und. Netview Datasupply Manager (NDM)			

Tabelle 3.10: Übersicht der Verteil- und Installationsarten im Bereich zS

erfolgreiche Systemtest. Der Projektleiter fordert hierfür vom SI die entsprechende Umgebung an. Diese Anforderung erfolgt per Mail oder telefonisch. In dieser Umgebung wird die Software integriert. Der Projektleiter bzw. PV übergibt sie dazu an den SI. Dieser überprüft die Software und übergebenen Begleitdokumente anhand der Festlegungen in den PRK. Der SI integriert dann zum einen die übernommenen Softwarekomponenten zu einem Produkt bzw. AWS (SW-SW-Integration), zum anderen integriert bzw. installiert er die Software in die Zielumgebung (SW-HW-Integration). Der SI testet dann sowohl die Integration als auch die Installation. Beim Scheitern dieser Tests wird diese Phase wiederholt. Im Vergleich zum dezentralen Bereich, wo standardmäßig eine Iteration vorgesehen ist, gibt es hier keine Vorgaben bzgl. der Häufigkeit der Iterationsschleifen.

Wenn die Integration erfolgreich war, fordert der Projektleiter vom SI die entsprechende QS-U an. In dieser Umgebung erfolgen in der QS-Phase Integrations-, Installations- und Betreibertests. Diese Phase kann im Falle des Scheiterns ebenfalls wiederholt werden. Die Pilotphase ist im Bereich zS optional. Sie wird projektabhängig durchgeführt. Falls die Pilotphase stattfindet, erfolgt sie ebenfalls unter Steuerung und Kontrolle des Projektleiters. Er fordert wiederum beim SI die entsprechende Pilotumgebung an. Der Kunde hat dann die Möglichkeit zum Testen.

Obwohl der SI meist bei der HVBInfo angesiedelt ist, wird erst jetzt die Software offiziell an den Betreiber übergeben. Dieser Wechsel der Verantwortlichkeiten entspricht der Übergabe an der Theke. Abbildung 3.35 zeigt die Aktivitäten im Produktionsübergabeprozess ab der Theke.

Die bevorstehende Änderung der produktiven Serversysteme wird durch das Eröffnen eines Changes durch den Projektleiter dokumentiert. Dieser Change wird vom Change-Manager, Betreiber und optional vom SI genehmigt. Es gibt keine Festlegung, wann genau die Change-Eröffnung zu erfolgen hat. Er kann deshalb auch schon bei Beginn der Realisierungsaufgaben eröffnet werden, muss jedoch spätestens vor Roll-outbeginn erfolgt sein.

Anschließend integriert und installiert der Betreiber die Software in der Liveumgebung des jeweiligen Kunden.

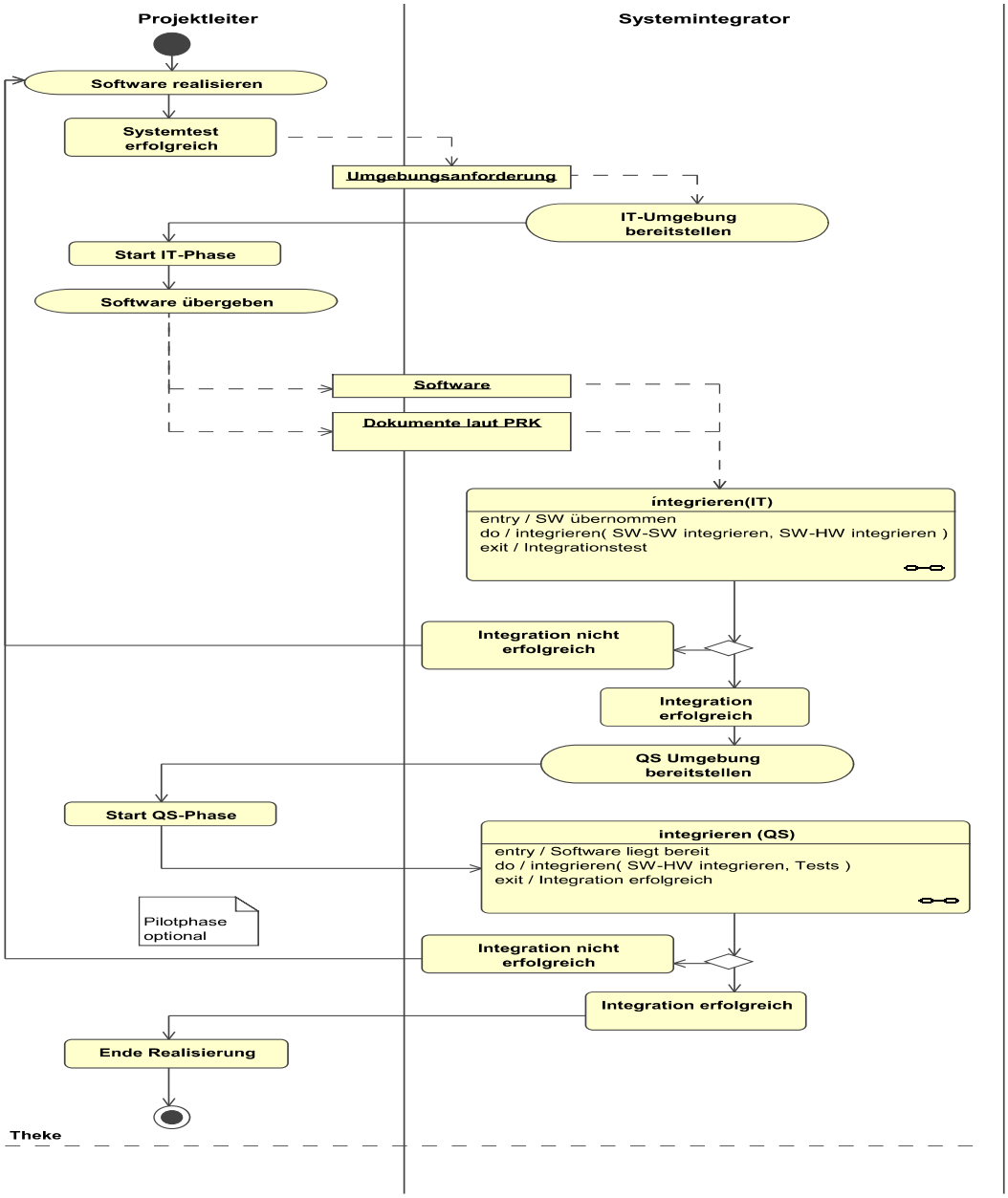


Abbildung 3.34: Darstellung der Aktivitäten bis zur Theke

Nach erfolgreicher Installation schließt der Projektleiter den zugehörigen Change und schließt das Entwicklungsprojekt offiziell mit der Projektnachbetrachtung (siehe nächsten Abschnitt) ab.

3.5.4 Bewertung des Release-Durchlaufes

Am Ende des Entwicklungsprojektes erfolgt eine Projektnachbetrachtung. Diese soll für zukünftige Projekte Anregungen und Hilfen geben. Im so genannten *Projektabschlussbericht* dokumentiert der Projektleiter den offiziellen Abschluss des Projektes. Mit diesem Dokument haben der Auftraggeber, das Projektteam, das Management und das Controlling die Möglichkeit, den Erfolg eines Projektes rückblickend zu bewerten. ALADIN bietet hierfür ein Template an. In diesem werden hauptsächlich quantitative Aspekte wie etwa geplanter und tatsächlicher Personal- und Sachaufwand erfasst. Bei Softwareänderungen im Rahmen der Anwendungsbetreuung braucht dieser Bericht nicht erstellt zu werden.

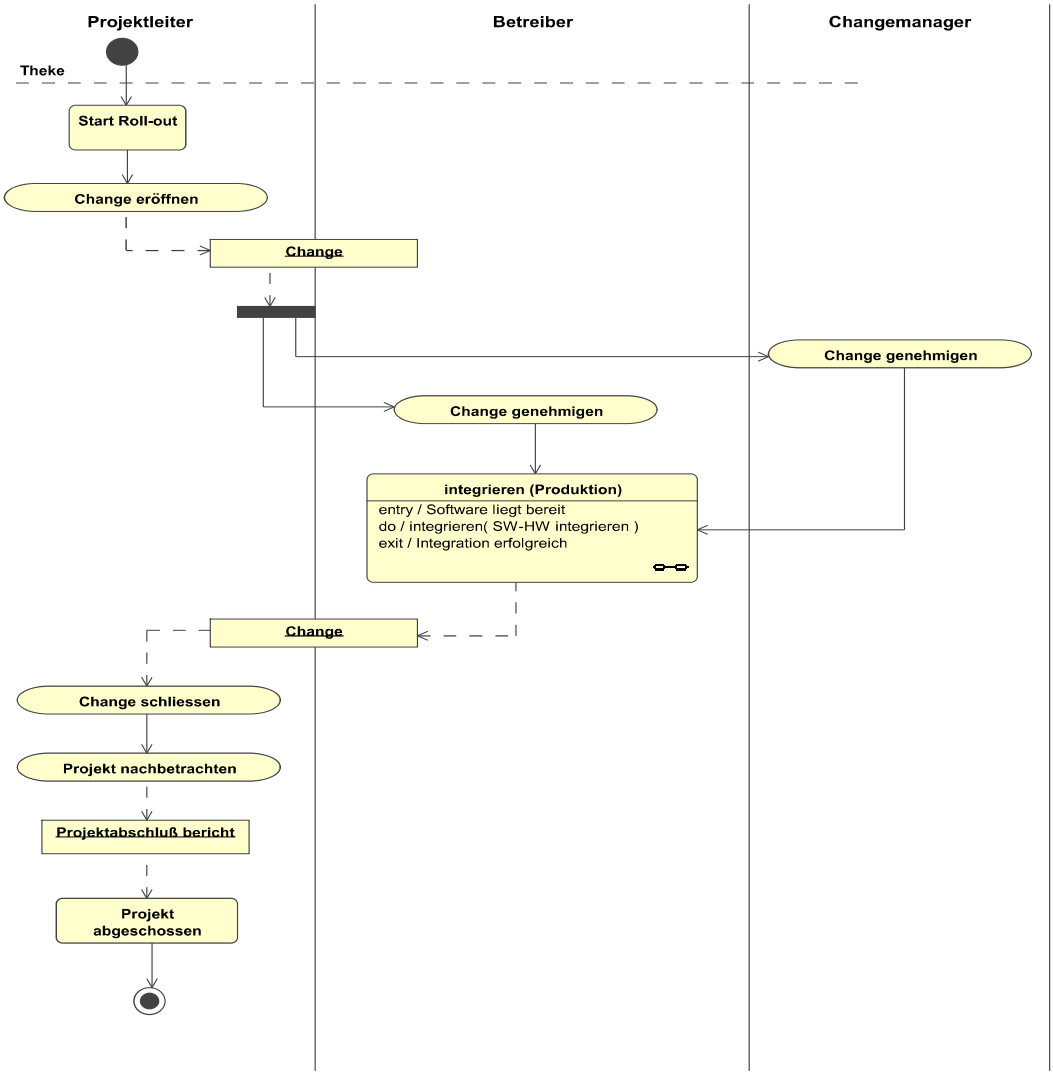


Abbildung 3.35: Darstellung der Aktivitäten zur Produktionsübergabe

3.5.5 Einordnung in das Lebenszyklusmodell

Die Einführung neuer und geänderter Produkte im Bereich zS entspricht dem im Abschnitt 2.2.6.8 beschriebenen Ablauf der Abnahme- und Einführungsphase im SWE. Das bei der HVBSystems entwickelte ALADIN-Vorgehensmodell ist ein SWE-Modell. Aus diesem Grund deckt es alle wesentlichen Bereiche des SWE ab (siehe Abbildung 3.36).

Die Teilprozesse Systemeinführung und Verifikation & Validierung entsprechen den Phasen Integration und Test im Lebenszyklusmodell. Die Lebenszyklusphase Release-Steuerung findet im Bereich zS nicht statt.

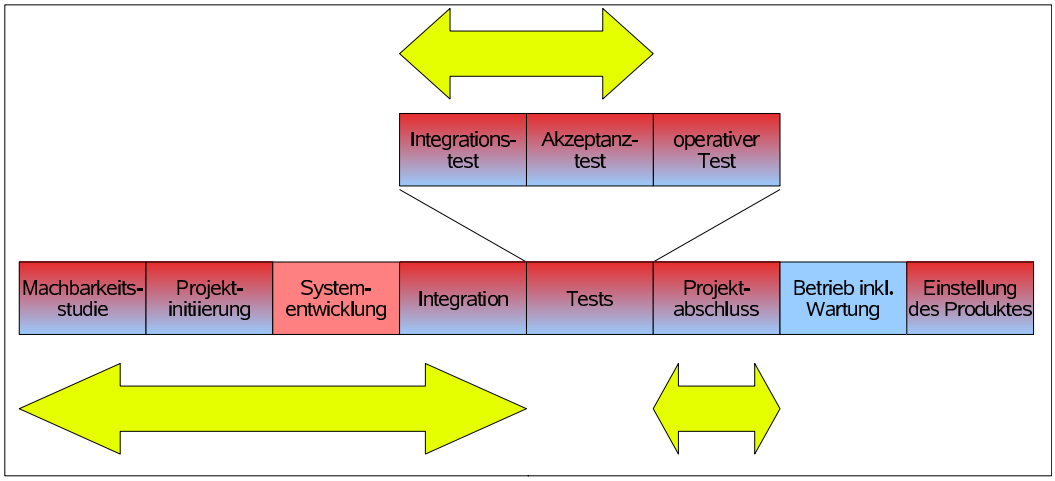


Abbildung 3.36: Abdeckung des Lebenszyklusmodell durch das ALADIN-Vorgehensmodell

3.6 Release-Management im Host-Bereich

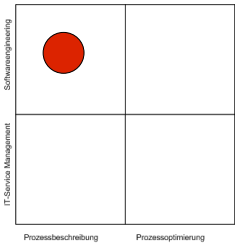


Abbildung 3.37: Einordnung RM Host

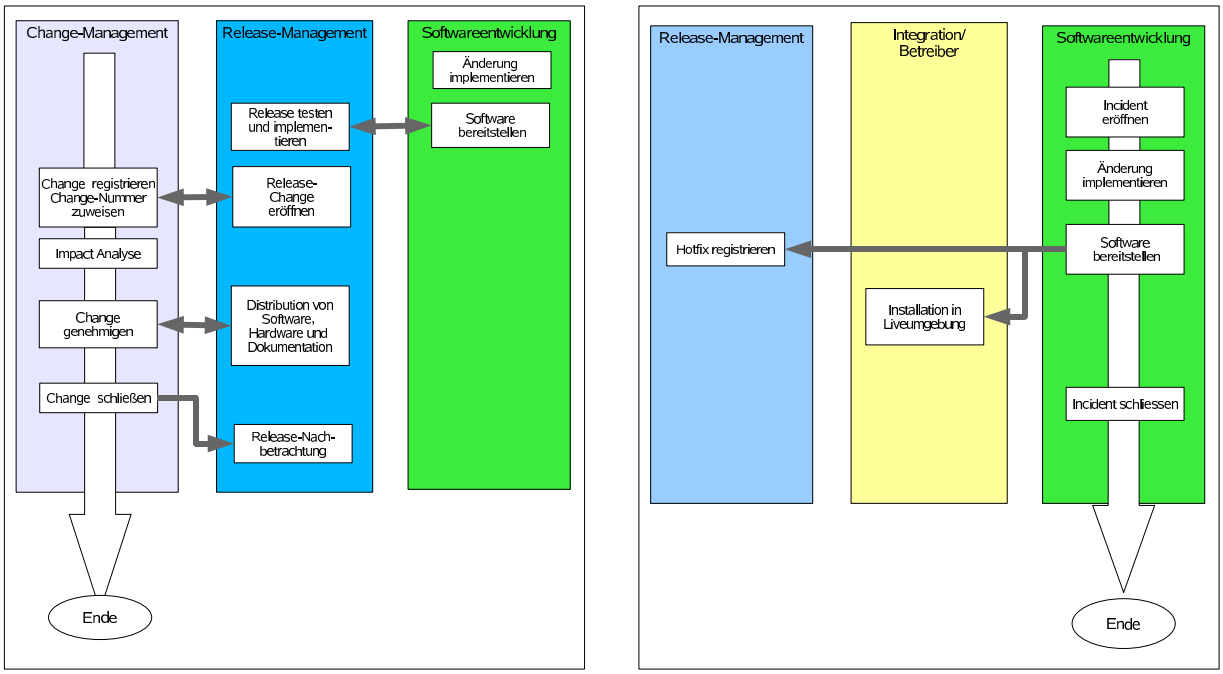
Das Release-Management im Host-Bereich befindet sich derzeit durch das Projekt (AQUA) in der Einführungsphase. Aus diesem Grund gibt es hier derzeit zwei parallele Prozesse. Zum einen werden die so genannten Hotfixes ähnlich dem Produktionsübergabe (PÜ) -prozess im Bereich zS vom Projektleiter bzw. AWSV gesteuert. Zum anderen werden Standard- und Sonder-Releases über den Release-Manager im Host-Bereich (RM-Host) als Release-Prozesse analog dem Release-Prozess im dezentralen Bereich abgewickelt.

Abbildung 3.38 zeigt die Gegenüberstellung des Zusammenspiels von Change- und Release-Management, Softwareentwicklung bzw. Integration. Teilbild a) zeigt den Ablauf für Standard- und Sonder-Releases im Host-Bereich analog dem Ablauf im dezentralen Bereich (vgl. Abschnitt 3.4.1). Teilbild b) zeigt den Ablauf im Host-Bereich für Hotfixes und Hotreleases. Er wird ähnlich dem Ablauf im Bereich zS vom Projektleiter bzw. AWSV gesteuert.

Aufgrund der Ähnlichkeit der Abläufe mit den vorher beschriebenen Verfahren werden im nachfolgenden Abschnitt nur noch die Unterschiede im Host-Bereich herausgearbeitet. Zunächst erfolgt die Beschreibung der an den Prozessen beteiligten Rollen und ihren Aufgaben und anschließend werden die jeweiligen Verfahren dargestellt. Bei Unterschieden zwischen dem Standard-/Sonder-Release-Prozess und dem Hotfix-Prozess werden diese direkt gegenübergestellt. Auf der linken Spalte sind dann jeweils die Beschreibungen zum Release-Prozess, auf der rechten Spalte die Beschreibung zum Hotfix-Prozess. Leere Spalten bedeuten, dass es keine entsprechenden Aktivitäten im jeweiligen Bereich gibt. Hotreleases bilden eine Ausnahme bei dieser Darstellung. Sie stellen eine Mischform zwischen der reinen vom Projektleiter bzw. AWSV gesteuerten und der vom RM gesteuerten Abläufe dar. Deshalb werden sie in nachfolgender Beschreibung unterschiedlich eingeordnet.

3.6.1 Beteiligte Rollen und ihre Aufgaben

Zur Unterscheidung der Release-Manager wird der Release-Manager im Host-Bereich nachfolgend als RM-Host bezeichnet. Die am Prozess beteiligten Rollen und ihre Aufgaben entsprechen denjenigen im dezentralen Bereich (vgl. Abschnitt 3.4.1) bzw. im Bereich zentraler Server (vgl. Abschnitt 3.5.1). Unterschiede bestehen lediglich in der Rolle des Systemintegrators. Er wird im Host-Bereich Anwendungsintegrator (AI) genannt. Die Rolle des SWV wird im Host-Bereich als Einsatzverantwortlicher (EI) bezeichnet. Anstelle des PV tritt im Host-Bereich der AWSV auf.



(a) Übersicht Prozessablauf im Host-Bereich bei Releases

(b) Übersicht Prozessablauf im Host-Bereich bei Hotfixes

Abbildung 3.38: Gegenüberstellung der Prozessabläufe im Zusammenspiel mit Release- und Change-Management, Softwareentwicklung und Integration

3.6.2 Release-Aktivitäten im Host-Bereich

In diesem Abschnitt werden die Release-Aktivitäten im Host-Bereich beschrieben. Die Gliederung erfolgt ebenfalls anhand der Release-Aktivitäten in ITIL, auch wenn es aufgrund der teilweisen SWE-Sicht nicht zu jedem Punkt eine Entsprechung gibt.

3.6.2.1 Festlegen der Releasepolicy und des Release-Plans

Dieser Abschnitt beschreibt die Festlegungen der Releasepolicy und des Release-Planes. Unterschiede zwischen Standard-Releases/Sonderversorgungen, Hotreleases und Hotfixes werdem im Text gegenübergestellt.

3.6.2.1.1 Festlegen der Releasepolicy im Host-Bereich

Die Releasepolicy für Standard-Releases entspricht den Richtlinien des dezentralen Bereiches (vgl. Abschnitt 3.4.2.1.1). Die Festlegungen werden auch hier vom Service Level-Manager und vom RM-Host festgelegt. In den SLA steht dann die Anzahl der durchzuführenden Standard-Releases. Hotreleases werden seit Mitte 2005 durchgeführt. Sie bündeln im Prinzip auf wöchentlicher Basis die für den Produktionseinsatz (entspricht Roll-out beim Host) bevorstehenden Softwareänderungen. Die Releasepolicy hierfür wird vom RM-Host festgelegt

Bei Hotfixes entspricht ein Release analog dem Bereich zS einer neu erstellten Softwareversion. Die Releasepolicy ist daher die Regel zur Erstellung dieser Version. Das Vorgehen hierfür ist im ALADIN-Vorgehensmodell dokumentiert. Es wird aber hier nicht näher erläutert, da die reine Softwareerstellung nicht im Fokus der Diplomarbeit steht.

Festlegung der Release-Einheiten Für die Aufgabenstellung der Diplomarbeit sind nur Systeme relevant, welche sich per KVS verwalten lassen. Allerdings unterscheiden sich auch hier die organisatorisch gehandhabten Einheiten von den technischen.

Beim Release-Prozess der Standard-Releases und Sonder-Releases erfolgt die organisatorische Abwicklung durch den RM-Host mithilfe des so genannten *RM-Tool-Host*. Es basiert wie Stargate ebenfalls auf PVCS Dimensions und wird zur Unterscheidung von Stargate des dezentralen Prozesses RM-Tool-Host genannt. Im Rahmen dieser organisatorischen Abwicklung erfolgt die Einmeldung der geänderten KVS-Teilsysteme auf AWS-Ebene. Der technische Verteilungsprozess jedoch handhabt so genannte KVS-Packages.

Bei Hotfixes und Hotreleases werden KVS-Packages betrachtet. Ein Hotfix entspricht hier i.d.R. einem KVS-Package.

KVS-Package
Abbildung 3.39 zeigt den Zusammenhang zwischen AWS, KVS-Package und Komponente.

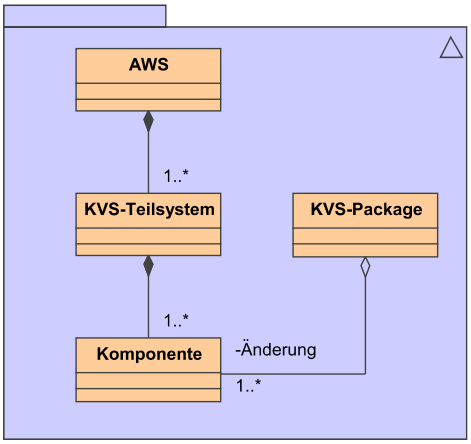


Abbildung 3.39: Zusammenhang zwischen AWS, KVS-Package und Komponente

Die KVS-Packages stellen die Übergabeobjekte dar. Der durch ein KVS-Packages definierte Übergabeauftrag muss entweder vollständig durchgeführt oder vollständig wieder zurückgenommen werden. Die Package-Art definiert dabei den Ablauf der Softwareverteilung (siehe Abschnitt 3.6.2.6).

Ein KVS-Package entspricht per Konvention einer fachlichen Änderung. Es kann daher mehrere von der fachlichen Änderung betroffene KVS-Teilsysteme bündeln. Allerdings wird diese Konvention nicht überprüft. Deshalb können auch mehrere fachliche Änderungen in einem KVS-Package gebündelt werden.

KVS-Packages werden mithilfe des Komponentenverwaltungs- und Übergabesystems (KVS) erstellt. Es basiert auf der Software *Environment for Development and Operation* (ENDEVOR-/MVS) von Computer Associates. In der neuesten Version als *AllFusion Endevor Change Manager* ist diese Software als kompatibel zum ITIL-Release-Management zertifiziert ([Comp 04]) worden.

Nach der ITIL-Definition entsprechen die Standard-, Sonder-Releases und Hotreleases im Host-Bereich einer Bündelung von Delta-Releases und somit sind sie auch Packagereleases. Ein Hotfix entspricht einem Delta-Release. Obwohl täglich zahlreiche Hotfixes durchgeführt werden, werden sie nicht zu einem Packagerelease gebündelt.

Release-Bezeichnung und -Nummerierung

Die Bezeichnung der Standard-Release setzt sich wie folgt zusammen:
$\langle Jahr \rangle$ $\langle Rel. \rangle$ _ $\langle laufendeNummer \rangle$.
Die Nummerierung der Sonder-Releases erfolgt analog. Die Bezeichnung der wöchentlichen Hotreleases ergibt sich aus der Kalenderwoche. Ein Hotrelease wird dann als
$\langle Jahr \rangle$_ $\langle Hotrelease - KW \rangle$ - $\langle Nr.Kalenderwoche \rangle$ bezeichnet. Bei der Release-Bezeichnung gibt es hier keinen Bezug zum Kunden, da die Host-Hardware als solche für alle Kunden eingesetzt wird.

Hotfixes sind durch ihre jeweilige PRUEV-Nummer (siehe Abschnitt 3.6.2.1.1) gekennzeichnet.

Risikoklasse	Risikoeinstufung	Merkmale
1	äußerst kritisch	massive Beeinträchtigung des Geschäftsbetriebes, Bankkunden sind empfindlich betroffen
2	sehr kritisch	Gefährdung des ordentlichen Betriebes trifft nach ca. einem Tag auf, Bankkunden sind betroffen
3	kritisch	„...“nach ca. 2-3 Tagen auf, Verzögerungen bei Bankkunden sind tolerierbar
4	weniger kritisch	Informationslücken können auftreten, Bankkunden sind nicht oder nicht kritisch betroffen

Tabelle 3.11: Übersicht Einstufungskriterien eines Systems in die Risikoklasse

Definition von Major- und Minor-Releases und einer Regel für Emergencyfixes Tabelle 3.2 auf Seite 72 zeigt die Zuordnung der bestehenden Release-Arten zu den ITIL-Major- bzw. Minor-Releases. Die Release-Prozesse unter Steuerung des RM sind derzeit in der Entstehung. Der Standard-Release-Prozess wird derzeit pilothaft für mit vier AWS durchgeführt. Für die Auswahl zwischen dem Hotfix- und Hotreleaseprozess sind noch keine expliziten Entscheidungskriterien vorgeschrieben. Für die Durchführung eines Hotfixes wird derzeit lediglich das Vorhandensein eines entsprechenden registrierten *Incidents* bzw. Changes gefordert.

Dieser Incident bzw. der entsprechende Incidentrecord entspricht im ITIL-Sinne einem Problem-Record. Die Eröffnung eines Changes ist nur für AWS der Risikoklasse 1 bzw. für die Top 50 AWS erforderlich. Die Einstufung in eine Risikoklasse erfolgt nach den in Tabelle 3.11 aufgelisteten Merkmalen. Die jeweilige Einstufung eines AWS ist im Repository gepflegt. Sowohl der Incident als auch der Change werden vom AWSV selbst eröffnet und nach erfolgtem Produktionseinsatz auch wieder geschlossen.

Häufigkeit von Major- und Minor-Releases Die Häufigkeit von Standard-Releases wird wie im dezentralen Bereich in den SLA festgelegt. Die Häufigkeit von Hotreleases ergibt sich implizit aus der wöchentlichen Durchführung. Der der Planung der Termine für die Hotreleases ist der jeweilige Umfang noch nicht bekannt. Dieser ergibt sich wie im dezentralen Bereich nach Ende der Einmeldephase. Weder bei Sonder-Releases noch bei der Häufigkeit von Hotfixes sind Werte für deren Häufigkeit festgelegt.

Identifikation von geschäftskritischen Zeiten, in denen Implementationen vermieden werden sollen Die Zeiten, in denen Implementation vermieden werden sollen, ergeben sich auch hier durch die vom Kunden vorgegebenen Frozen Zones.

Erwartete Begleitdokumentation für jeden Release-Typ

Standard-, Sonder- und Hotreleases werden durch ein Release-Dokument im RM-Tool-Host (entspricht Release-Begleitblatt im dezentralen Bereich) dokumentiert. Die Dokumentation der fachlichen Änderung erfolgt über KVS. Der PRUEV-Auftrag spezifiziert Aufträge für den AI (siehe Abschnitt 3.6.2.2).

Für Hotfixes wird vom RM die Eröffnung eines Incidents oder Changes gefordert. KVS-Packages bündeln auch hier wieder die fachlichen Änderungen. Die Erstellung eines PRUEV-Auftrages ist derzeit bei Hotfixes nicht zwingend vorgeschrieben. Andere Begleitdokumente wie etwa Benutzerhandbücher oder sind wie im Bereich zS (vgl. Abschnitt 3.5.2.1.1) in ALADIN vorgegeben.

Richtlinien, wie und wo Releases dokumentiert werden sollen

Das Release wird bei Standard-Releases, Sonderversorgungen und Hotreleases durch das Release-Dokument vom RM-Host im RM-Tool-Host dokumentiert.

Ein Hotfix ist im SWE-sinne eine definierte Softwareversion. Die Dokumentation hiervon ist das KVS-Package.

Regel für Erstellen und Testen von Back-out-Plänen Im Host-Bereich gibt es keine explizite Regel für das Erstellen und Testen von Back-out-Plänen. KVS automatisiert die Deployment-Aktivitäten und damit auch das Back-out. Abbildung 3.43 zeigt die Aktivitäten und Zustandsübergänge eines KVS-Packages. Durch die Aktivität „Übergabe zurücksetzen“ist ein automatischer Back-out möglich.

Zuständigkeiten des RM Im Host-Bereich ist das Release-Management unter Steuerung und Kontrolle des RM-Host in der Einführung.

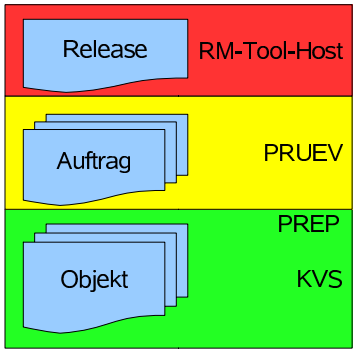


Abbildung 3.40: Toolzuordnung im Host-Bereich

Aus diesem Grund plant, steuert und kontrolliert der RM-Host derzeit Standard- und Sonder-Releases. Bei den Hotreleases legt der RM-Host auf wöchentlicher Basis die Termine fest. Ob jedoch ein AWSV seine Änderungen zu einem Hotrelease einmeldet oder ob er sie über den Hotfix-Prozess in die Produktion einbringt, bleibt dem jeweiligen AWSV überlassen. Nachdem Hotreleases eine Mischform zwischen vom Release-Manager gesteuerten Releases und vom Projektleiter eingeführten Änderungen darstellen, werden sie sowohl vom RM-Host als auch vom Projektleiter bzw. AWSV gesteuert.

Hotfixes werden vom Projektleiter bzw. AWSV gesteuert. Durch die Werkzeugintegration der Ebenen Release (RM-Tool-Host), Auftrag (PRUEV, PREP) und Objekt (KVS)(siehe Abbildung 3.40) wird der RM-Host zumindest von der Durchführung der Hotfixes in Kenntnis gesetzt.

Beschreibung des Release-Steuerungsprozesses

Der Release-Steuerungsprozess für Standard- und Sonder-Releases ist analog der Beschreibung im dezentralen Bereich (vgl. Abschnitt 3.4.2.1.1) im SIM-Portal durch den RM-Host veröffentlicht. Diese Veröffentlichungen umfassen derzeit die Prozessbeschreibung zur Produktionsübernahme und Release-Management im Host-Bereich und eine Verfahrensanweisung zur Produktionsübernahme. Die Beschreibung für Standard-Releases wird aufgrund des Projektfortschrittes von AQUA ständig erweitert.

Da Hotfixes und Hotreleases aus der Softwareentwicklung gesteuert werden, ist die Beschreibung des Release-Steuerungsprozesses analog im Bereich zS in ALADIN beschrieben (vgl. Abschnitt 3.5.2.1.1).

Dokumentation der exakten DSL-Konfiguration Weder der RM-Host noch der AWSV haben eine DSL als solche definiert.

3.6.2.1.2 Festlegen des Release-Plans Im Host-Bereich erfolgt keine kundenspezifische Release-Planung. Die Host-Hardware als solche wird für alle Kunden verwendet. Die Zuordnung zum jeweiligen Kunden erfolgt über die Einträge im *Repository*. Dort ist zu jedem AWS bzw. Teilsystem auch das Konzernfeld registriert. Der Kundensetbegriff kommt daher im Host-Bereich nicht zum Einsatz.

Die Planung der Standard- und Sonder-Releases erfolgt analog der Planung im dezentralen Bereich. Nachdem es hier keine kundenspezifische Release-Planung gibt, erfolgt die zeitliche Synchronisation der IT- und QS-Phase mit den IT- und QS-Phasen des grössten Kunden. Die Abstimmung erfolgt hier manuell.
Der resultierende Release-Plan wird im Excel-Release-Plan des dezentralen Bereichs mit veröffentlicht (siehe Abbildung 3.12). Aus diesem Gesamtplan wird der Host-spezifische Release-Plan abgeleitet (siehe Abbildung 3.41). Im Vergleich zum dezentralen Release-Plan berücksichtigt dieser zusätzlich die Zeiten zum Aufbau der GPL und die so genannte *Freeze Phase*. In der *Freeze Phase* dürfen keine Änderungen mehr eingespielt werden. Die Tests und vor allem der Batch laufen wie gehabt weiter. Ziel der *Freeze Phase* ist das Testen der Stabilität des Batch-Ablaufes durch Durchführen eines ununterbrochenen Batch-Durchlaufes.
Zum Testen des Batches muss bei Standard-Releases zusätzlich die Verfügbarkeit der für den Batch-Ablauf relevanten zS und für die Abnahme nötigen dezentralen Abnahmearbeitsplätze gewährleistet werden. Hier erfolgt die Abstimmung ebenfalls manuell. Der RM-Host dokumentiert dies in einer eigenen Excel-Liste.
Die Termine für den Produktionseinsatz der Hotreleases werden vom RM-Host auf Jahresbasis festgelegt. Für Hotreleases wird keine Testumgebung aufgebaut, da sie direkt in die Produktion überführt werden. Dadurch ist die Planung vereinfacht.

Nachdem die Erstellung von Hotfixes wie im Bereich zS (vgl. Abschnitt 3.5.2.1.2) einem Entwicklungsvorhaben gleicht, entspricht hier die Release-Planung der Planung eines Softwareentwicklungsvorhabens. Diese Planung wird hier nicht betrachtet, da die Softwareentwicklung nicht Gegenstand dieser Arbeit ist. Falls im Rahmen der Systemeinführung Schulungen der Endanwender nötig sind, wird dies vom Projektleiter bzw. PV geplant und mit den Betroffenen abgestimmt.

Nr.	Vorgangsname	Dauer	Anfang	Ende
57	**HOST Rel. 5-2006**	**37 Tage**	**Mi 06.09.06**	**Do 26.10.06**
58	**Einmeldeschluß**	0 Tage	Mi 06.09.06	Mi 06.09.06
59	**Abgabeschluss bei INF-IA3**	0 Tage	Do 07.09.06	Do 07.09.06
60	**Aufbau GPL**	**6 Tage**	**Fr 08.09.06**	**Fr 15.09.06**
61	Spiegel	2 Tage	Fr 08.09.06	Mo 11.09.06
62	Test Batchablauf "Alt"	4 Tage	Di 12.09.06	Fr 15.09.06
63	**IT-Phase**	**11 Tage**	**Fr 22.09.06**	**Fr 06.10.06**
64	Installation	6 Tage	Fr 22.09.06	Fr 29.09.06
65	IT-Abnahme	5 Tage	Mo 02.10.06	Fr 06.10.06
66	Synchronisierte Abnahme (C/S IT-Phase)	2 Tage	Mi 04.10.06	Do 05.10.06
67	**Freeze Phase**	5 Tage	Mo 09.10.06	Fr 13.10.06
68	**QS-Phase**	4 Tage	Mo 16.10.06	Do 19.10.06
69	Synchronisierte Abnahme (C/S QS)	2 Tage	Mi 18.10.06	Do 19.10.06
70	**Prod.Einführung**	5 Tage	Fr 20.10.06	Do 26.10.06

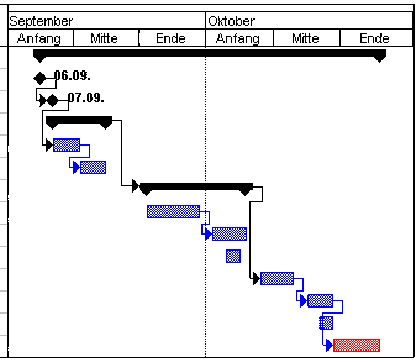


Abbildung 3.41: Auszug aus dem Host-Release-Plan für Standard-Releases

Testart	Tester	Umgebung
Modultest	HVBSystems	GET
Systemtest	HVBSystems	GET
Integrationstest	HVBSystems	GET
Abnahmetest	Kunde	GPL-IT oder GET-TEST-Private
Betreibertest	HVBInfo	GPL-QS

Tabelle 3.13: Übersicht der Testarten, Testumgebung und Tester im Host-Bereich laut Testkonzept der HVBSystems

3.6.2.2 Entwurf, Release-Zusammenstellung und -Konfiguration

Weder Standard-Releases und Sonder-Releases noch Hotfixes und Hotreleases werden vom RM-Host entworfen. Der jeweilige Entwurf wird durch die Projektleiter bzw. AWSV im Rahmen der Softwareentwicklung definiert.

Die Zusammenstellung von Sonder-, Standard- und Hotreleases als Bündelung einzelner Softwareänderungen durch KVS-Packages ergibt sich aus der Summe der eingemeldeten Softwareänderungen. Diese sind dem RM-Host vorab nicht bekannt. Sie werden durch den jeweiligen Projektleiter bzw. AWSV eingemeldet.

Jeder Hotfix entspricht einer Softwareänderung und damit einem Release an sich. Der Hotfix wird deshalb auch im Rahmen des Softwareentwicklungsprojektes als KVS-Package vom Softwareentwickler zusammengestellt.

Die jeweilige Konfiguration ist in den KVS-Packages und PRUEV-Aufträgen spezifiziert.

PRUEV-Auftrag PRUEV ist ein Workflowtool zur Unterstützung des Produktionsübergabeprozesses im Host-Bereich. Es bietet für jede Übergabe so genannte *Übergabeauftragstypen* an. Der Übergabeauftragstyp bestimmt dann die jeweils durchzuführenden Arbeitsschritte. Festgelegte Übergabeauftragstypen sind z.B. „neue Batch-Anwendung“oder „Änderung einer Batch-Anwendung“. Über das Tool Produktionseinsatz und -planung (PREP) werden releaserelevante Informationen aus PRUEV in das RM-Tool-Host überspielt. PRUEV standardisiert somit auch Kommunikations- und Dokumentationsaufgaben bei der Übergabe.

3.6.2.3 Testen und Abnahme des Releases

Tabelle 3.13 zeigt die Übersicht der Testarten laut dem veröffentlichten Testkonzept der HVBSystems. Modultests bis einschließlich Integrationstests erfolgen durch die jeweiligen AWSV in der GET. Die GET teilt sich wiederum auf in (siehe auch Abbildung 3.42):

ENTW-Kern: Entwicklungsumgebung für Kernsysteme

ENTW-Public: Entwicklungsumgebung für allgemeine Systeme

TEST-Private: optionale Testumgebung für komplexe Systeme oder Großprojekte; diese kann beim Umgebungsmanager beantragt werden; dadurch sind Tests unabhängig von der Weiterentwicklung möglich; die Datenversorgung wird hier mit beauftragt

Die Klassifikation eines AWS in ein Kernsystem ist im *Repository* vermerkt. Diese Klassifikation wird an die Teilsysteme vererbt. Weiterhin ist im Repository bei jedem AWS ein so genanntes GPL-Kennzeichen hinterlegt. Es legt fest, ob das AWS bzw. seine Teilsysteme generell über die GPL in die Produktion oder ob sie direkt von der GET in die Produktion gebracht werden.

Inhalt	Host	Bemerkungen
KVS-Systeme	KVS-Verfahren	Standardverfahren, nur dieses wird im Release-Prozess betrachtet
Nicht-KVS-Systeme bzw. externe Systeme	manuelles Verfahren	
Betriebssystem	SMPE	für Datenbanken gibt es hierfür ein abgewandeltes Verfahren

Tabelle 3.15: Übersicht der Verteil- und Installationsarten im Bereich Host

Bei Standard- und Sonder-Releases werden die Abnahme- und Betreibertest in der GPL durchgeführt. GPL-IT und GPL-QS entsprechen hierbei der jeweiligen Prozessphase. Vom RM-Host wird eine schriftliche Dokumentation der erfolgten Abnahme analog dem dezentralen Bereich gefordert.

Die GET-TEST-Private wird lediglich auf Anfrage des Projektleiters bereitgestellt. In diesem Falle erfolgen die Abnahmetests in der GET-TEST-Private, ansonsten in der GET. Der RM-Host überprüft hier jedoch nicht die Durchführung der Abnahmetests. Es kann somit von Seiten der HVBInfo keine Aussage getroffen werden, ob diese auch tatsächlich stattfinden.

3.6.2.4 Planung des Einsatzes

Der Einsatzverantwortliche (EI) führt das Release in die Produktion ein. Hierfür erstellt er in PREP den Produktionseinsatzplan nach dem er die Software in der Produktion installiert.

3.6.2.5 Kommunikation, Vorbereitung und Schulung

Für Vorbereitungs- und Schulungsmaßnahmen ist das EM zuständig.

Bei Standard- und Sonder-Releases erfolgt die jeweilige Abstimmung über das RM. Die Kommunikation über die jeweiligen ReleaseAktivitäten erfolgen durch den RM-Host über Newsletter.

Bei Hotfixes und Hotreleases kümmert sich der jeweilige AWSV oder Projektleiter um die Abstimmung mit dem EM für die notwendigen Vorbereitungs- und Schulungsmaßnahmen. Er ist auch für die Kommunikation mit den am Übergabeprozess beteiligten Rollen verantwortlich. Der RM-Host wird durch die Toolintegration von RM-Tool-Host - PRUEV - PREP - KVS über bevorstehende Hotfixes informiert.

3.6.2.6 Release-Verteilung und Installation

In Tabelle 3.15 sind alle Verteil- und Installationsarten im Host-Bereich aufgeführt. Für die Aufgabenstellung in der Diplomarbeit werden derzeit nur die KVS-Systeme betrachtet. Grund hierfür ist die derzeitige Einführung des Release-Managements im Host-Bereich. Hier werden ebenfalls nur durch KVS verteilbare Systeme betrachtet.

KVS automatisiert hierbei die Deployment-Aktivitäten. Die Art eines KVS-Packages definiert den Verteilprozess. Es gibt hierbei das *Freigabe-Package*, *Produktions-Package* und das *Fix-Package*. Verteilt werden die Packages über die verschiedenen KVS-Ebenen (siehe Abbildung 3.42). Eine KVS-Ebene entspricht einem Status. Ein Freigabe-Package wird von VORPROD nach IT1 bzw. IT2-4 nach PROD übergeben, ein Produktions-Package wird von PROD nach PRODHV übergeben und ein Fix-Package von der FIX- auf die jeweilige PROD-Ebene.

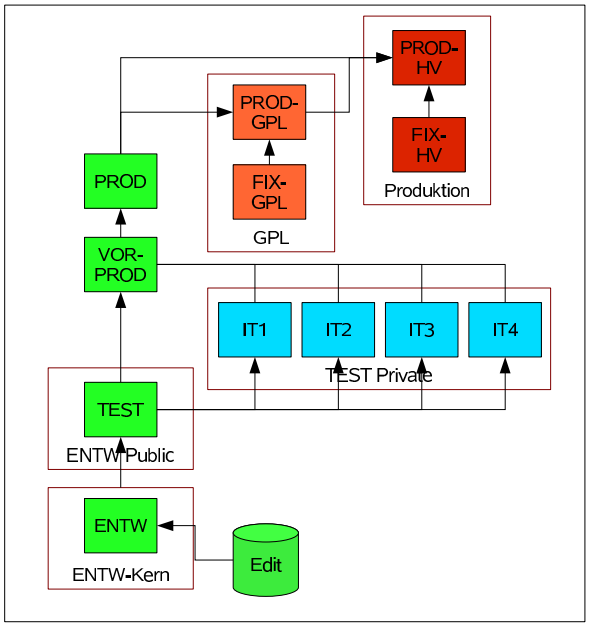


Abbildung 3.42: Ebenen des KVS

Abbildung 3.43 zeigt die Aktivitäten und Zustandsübergänge beim so genannten *Package Processing* in KVS. Ausgehend von einem nicht existierenden Package wird durch die Tätigkeit des Erstellens ein KVS-Package erstellt. Hier wird auch die Art des Packages festgelegt. Solange das KVS-Package noch nicht fixiert ist, kann es weiter durch den AWSV bearbeitet und geändert werden. Das Fixieren entspricht der Erstellung einer *Baseline*. Nach der Festlegung der *Baseline* muss das KVS-Package durch definierte *Approver* genehmigt werden. Diese sind im Repository hinterlegt. Dieser Genehmigungsprozess erfolgt automatisch über KVS und ist eine softwareentwicklungsinterne Prozedur. Approver sind technisch oder fachlich verantwortliche Mitarbeiter. Hiermit soll ein 4-Augen-Prinzip gewährleistet werden. Nach der Genehmigung wird die durch die Package-Art definierte Aufgabe ausgeführt, z.B. das Übergeben in die nächste Ebene. Durch die Deaktivierung der Übergabe erfolgt ein automatischer Back-out (entspricht Roll-back). Dieser kann durch die Reaktivierung der Übergabe wieder zurückgenommen werden. Nach der Freigabe eines KVS-Packages ist es im Zustand *Commited*. Ab dann ist es in der jeweiligen Ebene aktiv.

3.6.3 Beschreibung des Release-Steuerungsprozesses

Nachfolgend werden die Aktivitäten des Release-Steuerungsprozesses beschrieben. Unterschiede im Prozess zwischen Standard bzw. Sonder-Releases und Hotfixes werden im Text gegenübergestellt.

Release-Steuerungsprozess für Standard-Releases: Der Release-Steuerungsprozess für Standard-Releases, Sonder-Releases und Hotreleases entspricht im Wesentlichen dem im dezentralen Bereich (vgl. Abschnitt 3.4.3). Die Funktionalitäten des RM-Tool-Host sind an den Funktionalitäten von Stargate orientiert.

Release-Steuerungsprozess für Hotfixes: Bei Hofixes erfolgt die Steuerung der Systemeinführung und Verifikation & Validierung analog dem Bereich zS durch den Projektleiter bzw. AWSV.

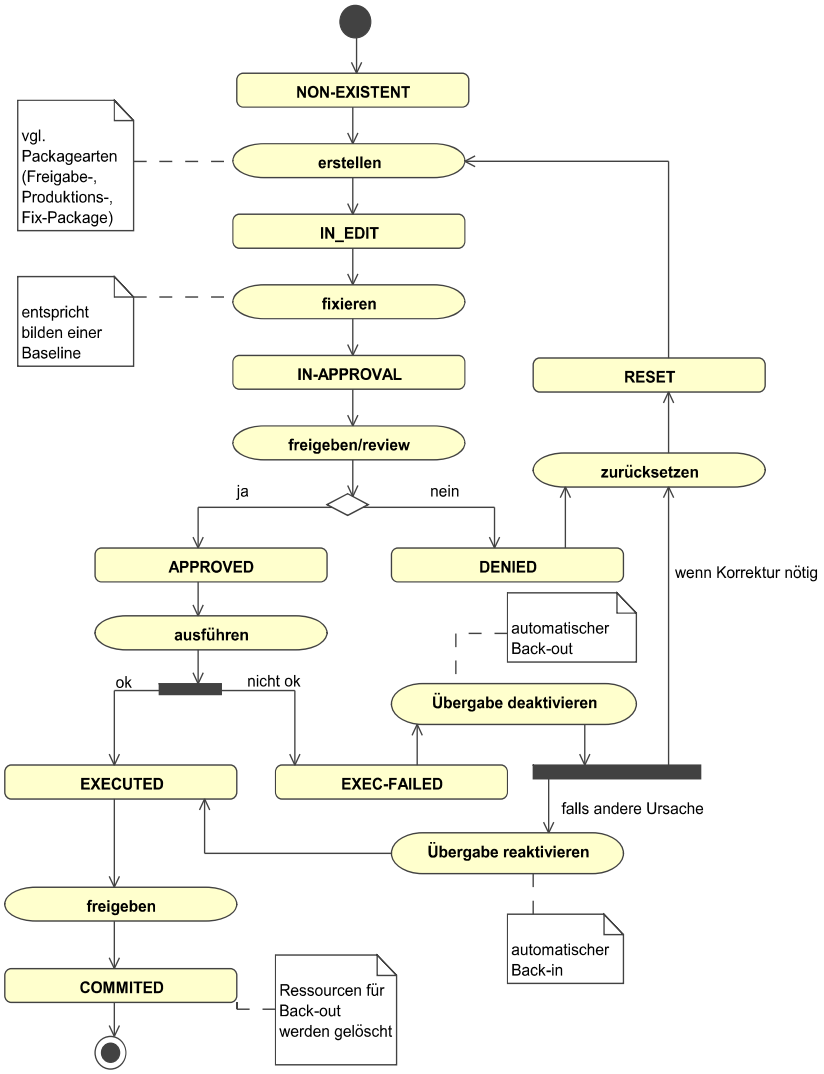


Abbildung 3.43: Übersicht Ablauf Weiterleitung durch KVS-Ebenen

3.6.3.1 Release eröffnen

Die Aktivitäten bei der Eröffnung eines Releases entsprechen denjenigen im dezentralen Bereich (siehe Abschnitt 3.4.3.1). Nach der Eröffnung ist ein Release für die Einmeldung von Änderungen durch den AWSV bereit und der jeweilige Release-Durchlauf kann beginnen.

3.6.3.2 Produkt- und Release-Status überwachen

Im Host-Bereich sind durch die Bündelung von Änderungen an einzelnen Host-Komponenten der Status des Releases und der Status der jeweiligen Komponenten verknüpft. Aus diesem Grund überwacht der RM-Host sowohl den Komponenten- als auch den Release-Status in den nachfolgend beschriebenen Phasen.

Bei Hotfixes entspricht ein Release im SWE-Sinne einer bestimmten Softwareversion. Jeder Hotfix entspricht deshalb einem Release. Die Überwachung des Hotfix-Status erfolgt durch den Projektleiter bzw. AWSV.

3.6.3.2.1 Einmeldephase

Im Host-Bereich erfolgt die Einmeldung durch den AWSV auf AWS Ebene. Zusätzlich zum dezentralen Bereich erfolgt hier noch die logische Zuordnung der KVS-Packages und PRUEV-Aufträge zur jeweiligen Einmeldung (vgl. Abbildung 3.44). Der AWSV kann hier für jede fachliche Änderung eine Einmeldung generieren.

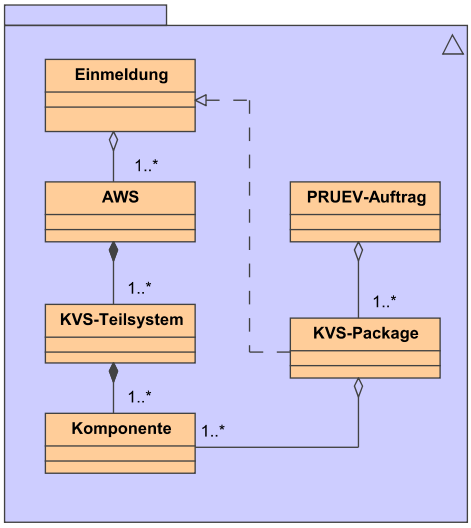


Abbildung 3.44: Zusammenhang zwischen Einmeldung, AWS, KVS-Package und PRUEV

Release-Umfang festlegen

Der Release-Umfang ergibt sich aus der Summe der Einmeldungen. Nachdem der Standard-Release-Prozess erst pilothaft eingeführt wurde, gibt es hierfür noch keine weiteren Einschränkung bzgl. der Anzahl der Einmeldungen wie etwa im dezentralen Bereich. Ebenso ergibt sich bei den Hotreleases der Release-Umfang durch die Summe der jeweiligen Einmeldungen. Auch hier gibt es derzeit keine Einschränkungen.

Der Umfang eines Releases bei Hotfixes ergibt sich aus dem jeweiligen Entwicklungsvorhaben. Die Anzahl der jeweils durchgeführten einzelnen Hotfixes wird durch den RM-Host über das RM-Tool-Host registriert.

3.6.3.2.2 Implementierung In dieser Phase werden die KVS-Packages und PRUEV-Aufträge vom AWSV bereitgestellt. Der AI integriert sie durch Bearbeitung der Änderungen in PRUEV und KVS.

Bei Standard-Releases und Sonderversorgungen baut der UV vor Beginn der IT-Phase die GPL auf und testet ihren Grundzustand. Hierfür spiegelt er die Produktion und betreibt die GPL unter Produktionsbedingungen.

3.6.3.2.3 IT- und QS-Phase

Sowohl die IT- und die QS-Phase finden im Host-Bereich bei Standard- und Sonder-Releases in der GPL statt. Bei Hotreleases erfolgen diese Phasen in der GET. Sie werden dann direkt von der Entwicklung in die Produktion gebracht.
Der UV implementiert die Release-Bündel in der GPL. Der Integrationsplan hierfür ist im *PREP* festgelegt. PREP ist ein Tool zur Unterstützung des Integrationsprozesses. In Abbildung 3.40 ist die Tool-Zuordnung im Host-Prozess dargestellt.
Im Rahmen der Tests werden vom AWSV, AI und Kunden die Abnahmen durchgeführt, der Batch-Ablauf geprüft und die Fehlerdokumentation erstellt. Die IT- und QS-Phase sind durch die so genannte *Freeze-Phase* getrennt. In der *Freeze Phase* dürfen keine Änderungen mehr nachgeliefert werden. Hier werden dann auch die vom AWSV geöffneten Incidents geschlossen.

Bei Hotfixes gibt es laut ALADIN-Vorgehensmodell ebenfalls die IT-und QS-Phase. Diese finden in der GET statt und werden vom jeweiligen Projektleiter oder AWSV dezentral gesteuert und überwacht. Allerdings schreibt ALADIN hier nicht zwingend den Durchlauf dieser Phasen vor. Sie können u.U. bei Host-Systemen auch entfallen. Die Art und der Umfang der Tests entscheidet dann auch der Projektleiter bzw. AWSV.

3.6.3.2.4 Produktionsübergabe Im Host-Bereich gibt es keine Pilotphase. Allerdings besteht prinzipiell die Möglichkeit, dass die Host-Umgebung explizit für die Pilotphase im Bereich C/S oder zS angefordert wird.

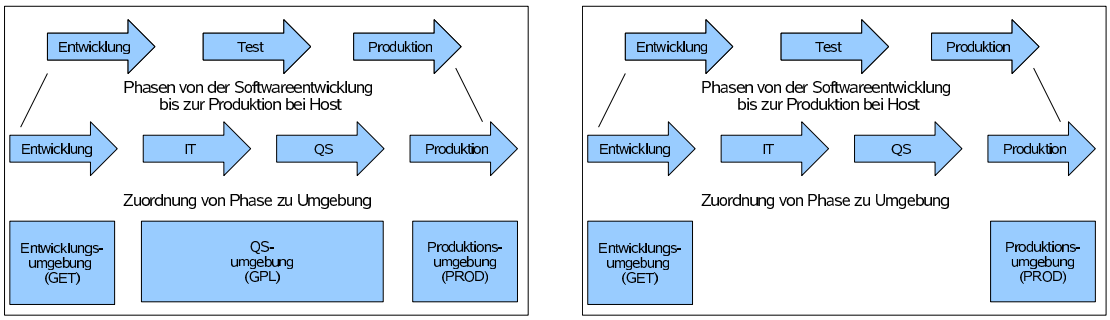
Für die Produktionsübergabe erstellt der EI den Produktionseinsatzplan in PREP. Anschließend wird das Release automatisch in der Produktion installiert.

3.6.3.3 Change eröffnen

Analog dem dezentralen Bereich eröffnet der RM-Host vor der Installation des Releases einen Change (siehe Abschnitt 3.6.3.3).

Bei Hotfixes und Hotreleases wird lediglich bei den als äußerst kritisch eingestuften AWS (vgl. Tabelle 3.11) oder bei den zu den Top50 gehörenden AWS ein so genannter releasebegleitender Change durch den Projektleiter bzw. AWSV eröffnet.

Zur Unterscheidung der Change-Begriffe wird der durch den Release-Manager eröffnete Change auch als *Release-Change* und der durch den Projektleiter bzw. AWSV eröffnete Change auch *release-begleitender Change* bezeichnet.



(a) Zuordnung von Prozessphasen zu Umgebungen bei Standard- und Sonder-Releases

(b) Zuordnung von Prozessphasen zu Umgebungen im Host-Bereich bei Hotfixes und Hotreleases

Abbildung 3.45: Gegenüberstellung der Prozessphasen und benötigten Umgebungen bei den verschiedenen Release-Prozessen

3.6.4 Bewertung des Release-Durchlaufes

Bei Standard- und Sonder-Releases ist der RM-Host für die Bewertung des Release-Durchlaufes zuständig. In der Release-Nachbetrachtung erstellt er Kennzahlen zu z.B.:

- Termintreue
- Release-Umfang
- Anzahl Nachlieferungen aufgrund von Fehlern oder verspäteten Einmeldungen
- Fehleranzahl je Phase (siehe z.B. Abbildung 3.6.4)

Die Informationen zu den Fehlerzahlen kann der RM-Host direkt aus einer Datenbank extrahieren. Dort dokumentieren die Prozessbeteiligten (AI, UV, AWSV) die aufgetretenen Fehler.

Bei Hotfixes und Hotreleases erfolgt analog dem Bereich zS eine Nachbetrachtung des Entwicklungsprojektes durch den Projektleiter bzw. AWSV. Nachdem mittlerweile der RM-Host auch über die Durchführung von Hotfixes informiert wird, erstellt dieser auch hier Kennzahlen. Diese Kennzahlen betreffen jedoch hauptsächlich quantitative Aspekte wie etwa Anzahl und Durchsatz von Hotfixes.
Über qualitative Aspekte kann der RM-Host keine Aussagen treffen, da die Tests dezentral unter Kontrolle des jeweiligen Projektleiters bzw. AWSV erfolgen.

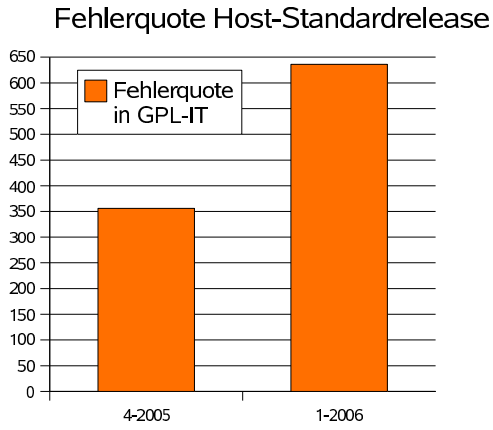


Abbildung 3.46: Fehlerverlauf Standard-Release Host in IT-(GPL)-Phase bei 4 AWS

3.6.5 Einordnung in das Lebenszyklusmodell

Die Einordung der Release-Aktivitäten in das Lebenszyklusmodell stellt Abbildung 3.47 dar. Bei Hotfixes und Hotreleases wird der Prozess aus der Softwareentwicklung entsprechend den Vorgaben im ALADIN-

Vorgehensmodell gesteuert. Die Abdeckung des ALADIN-Vorgehensmodells mit dem hier eingeführten Lebenszyklusmodell zeigt Abbildung 3.36 auf Seite 113. Die ALADIN-Phasen der Produktionsübergabe und Verifikation & Validierung entsprechen im Lebenszyklusmodell der Phase Integration und Test mit den Testarten Integrationstest, Akzeptanztest und operativer Test.

Sowohl bei Standard-Releases als auch bei Hotfixes und Hotreleases sind diese Phasen relevant. Allerdings erfolgt bei Standard- und Sonder-Releases die Steuerung und Kontrolle dieser Phasen zentral durch den RM-Host. Bei Hotfixes und Hotreleases dezentral durch den jeweiligen Projektleiter bzw. AWSV.

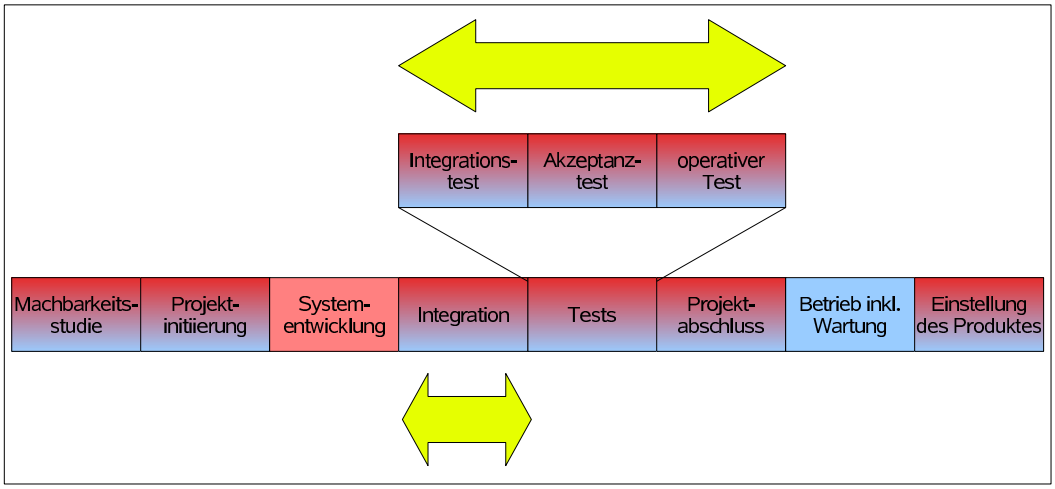


Abbildung 3.47: Einordnung des RM-Host in das Lebenszyklusmodell

3.7 Zusammenfassung

In diesem Kapitel wurden die bestehenden plattformspezifischen Release-Prozesse im Bereich C/S, zS und Host vorgestellt. Zu jedem Prozess wurden die beteiligten Rollen und ihre Aufgaben und der Ablauf des Release-Steuerungsprozesses beschrieben. Jedes dieser Prozesse ist aus Sicht des Lebenszyklusmodells in die SWE-Sichtweise einzuordnen. Der Schwerpunkt der bestehenden Release-Prozesse liegt in der begleiteten Einführungs- und Abnahmephase, wie sie in Abschnitt 2.2.6.8 ab Seite 33 bei Balzert oder Steinweg beschrieben wurden. Technische Subprozesse bilden die so genannten Deployment-Prozesse. Diese beinhalten Aktivitäten wie Installieren, Verteilen oder Aktivieren (vgl. Abschnitt 2.2.6.6 ab Seite 29).

Unterscheidungsmerkmale der bestehenden Release-Prozesse sind neben der Ausrichtung an der Technologie der jeweiligen Plattform die Frequenz, mit der sie durchgeführt werden, die Durchlaufzeit, die Zuständigkeiten oder der Umfang. Tabelle 3.48 zeigt die Übersicht der wichtigsten Unterscheidungsmerkmale.

Plattform Release-Arten und Merkmale	C/S	zS	Host
Standard-Release	√	-	√ (in Einführung)
Frequenz	getaktet, kundenabhängig (∼ 5 Standard-Releases je Kunde)		getaktet, kundenabhängig, meist angepasst an C/S
Durchlaufzeit	Vollrelease ca. 21 Wochen, Updaterelease ca. 17		ca. 8-10 Wochen
Zuständig	RM-C/S		RM-Host

Plattform Release-Arten und Merkmale	C/S	zS	Host
Umfang	Vollrelease (ca. 380 Produkte bei KVZ), Updaterelease (ca. 180 KVZ)		4 AWS (Pilotprojekt)
Sonderversorgung	√	-	√
Frequenz	nicht definiert		nicht definiert
Durchlaufzeit	ca. 6 Wochen		noch nicht festgelegt
beteiligte Organisationen	HVBInfo, HVBSystems, Kunde		HVBInfo, HVBSystems, Kunde
Zuständig	RM-C/S		RM-Host
Umfang	wenige Produkte		wenige AWS
Hotfix	√	-	√
Frequenz	nicht definiert		nicht definiert, bei Bedarf täglich
Durchlaufzeit	ca. 2 Wochen		1 Tag
Zuständig	RM-C/S		AWSV
Umfang	wenige Produkte		$\approx$ 500 – 1000 PRUEV-Aufträge je Monat ($*$)
Securityfix	√	√	-
Frequenz	bei Bedarf	monatlich	
Durchlaufzeit	ca. 2 Wochen	ca. 1-2 Wochen	
Zuständig	RM-C/S	RM-zS	
Umfang	wenige Produkte	wenige Produkte (Sicherheits-Patches)	
Corerelease	-	√ (Windows zS)	-
Frequenz		ca. 2-3 pro Jahr	
Durchlaufzeit		ca. 10 Wochen	
Zuständig		RM-zS	
Umfang		Core-Produkte (Betriebssystem und betriebssystemnahe Produkte)	
Hotrelease	-	-	√ (bündelt Hotfixes)
Frequenz			wöchentlich
Durchlaufzeit			ca. 5 Tage
Zuständig			RM-Host, AWSV
Umfang			$\approx$ 500 – 1000 PRUEV-Aufträge je Monat ($*$)

Plattform Release-Arten und Merkmale	C/S	zS	Host
Change bzw. Applikations-Release	-	√	-
Frequenz		projektabhängig (∗∗)	
Durchlaufzeit		projektabhängig, zwischen wenigen Tagen und mehreren Wochen	
Zuständig		AWSV, PV oder Projektleiter	
Umfang		i.d.R.ein AWS	
(∗) Zuordnung PRUEV - Änderung siehe Abbildung 3.44 auf Seite 123 (∗∗) im Jahr 2005 gab es 950 Changes im Bereich Windows zS, davon 133 Emergencychanges			

Abbildung 3.48: Übersicht der wichtigsten Merkmale aller Release-Arten

Die Integration dieser heterogenen Prozesslandschaft zu einem einzigen plattformübergreifenden Release-Prozess hat technische, organisatorische und finanzielle Gründe. Vorteile technischer Art sollen sich durch eine Verkürzung der Prozesslaufzeit und eine Reduzierung des Testaufwandes ergeben. Hieraus resultieren weitere Qualitätsverbesserungen durch ein Reduzierungspotenzial bei Incidents und bei der Möglichkeit zur Durchführung kompletter Geschäftsprozesstests im Rahmen des Release-Prozesses, was bisher überhaupt nicht möglich war. Durch die Integration entsteht zukünftig ein einziger stabiler Prozess, der einheitlich dokumentiert ist. Dadurch ergeben sich organisatorische Vorteile, wie etwa klare Verantwortlichkeiten, definierte Kommunikationswege und festgelegte Abläufe im Prozess. Finanzielle Vorteil sollen sich aus der Reduktion von Hotfixes, einer besseren Auslastung der Testumgebung und des technischen Personals ergeben.

Die bestehenden Release-Verfahren werden im nächsten Kapitel analysiert. Aufgrund dieser Analyse werden die Anforderungen an den neu zu konzipierenden Gesamtprozess definiert. Neben den Aspekten aus der Analyse liefern dazu auch die in Kapitel 2 beschriebenen Release-Prozesse des ITSM Vorgaben.

4 Analyse des bestehenden Release-Managements und Ableitung von Anforderungen an das integrierte Release-Management

Inhaltsverzeichnis

4.1 Analyse des Release-Managements im dezentralen Bereich 130
4.1.1 Verbindung zum Change-Management 131
4.1.2 Fehlendes Konfigurationsmanagement 132
4.1.3 Allgemeine Defizite im bestehenden Release-Management im dezentralen Bereich . 133
4.1.3.1 Unterschiede im Release-Steuerungsprozess 133
4.1.3.2 Unzureichende Workflow-Unterstützung 135
4.2 Analyse des Release-Managements im Bereich zentraler Server 135
4.2.1 Fehlendes Release-Management 136
4.2.2 Allgemeine Defizite im bestehenden Release-Management im Bereich zentraler Server 137
4.2.2.1 Unkoordiniertes Umgebungsmanagement 137
4.2.2.2 Ungenügende technische Standards 137
4.2.2.3 Zentrales Qualitätsmanagement fehlt 137
4.3 Analyse des Release-Managements im Host-Bereich 138
4.4 Ableitung von Anforderungen an das zu konzipierende, plattformübergreifende Release-Management 140
4.4.1 Orientierung des RM an ITIL 140
4.4.1.1 Einheitliche Festlegung der Prozessgrenzen 140
4.4.1.2 Verknüpfung mit dem Change-Management 141
4.4.1.3 Einführung des Konfigurationsmanagements 142
4.4.2 Qualitätsmanagement 143
4.4.3 Umgebungsmanagement 143
4.4.3.1 Geregelte Beauftragung 143
4.4.3.2 Virtualisierte Umgebungen 144
4.4.4 Anforderungen aufgrund der höheren Komplexität des integrierten Release-Managements 144
4.4.4.1 Anpassung der Taktung 144
4.4.4.2 Werkzeugunterstützung 145
4.4.4.3 Einführung von plattformübergreifenden Rollen 146
4.4.4.4 Einführung von Standards 146
4.5 Zusammenfassung 147

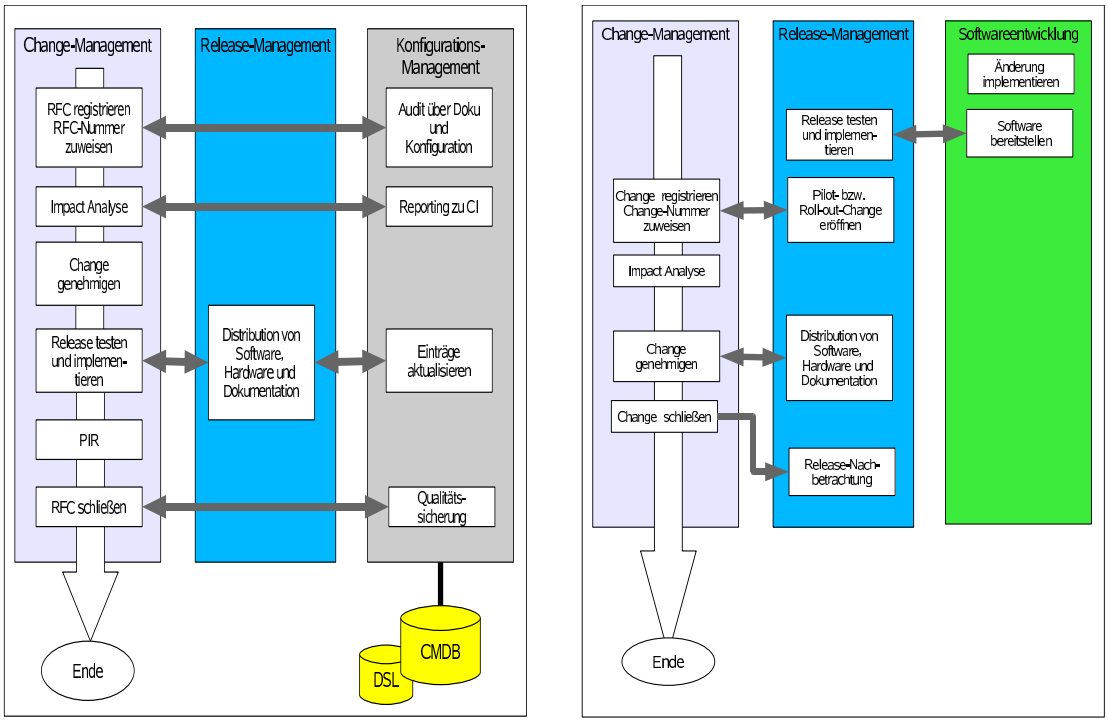
In diesem Kapitel wird das in den vorangegangenen Abschnitten 3.4 bis 3.6 bestehende Release-Management (RM) bei der HVBInfo analysiert. Die Ausgangslage für diese Analyse bildet zum einen ITIL als Grundlage des neuen RM, zum anderen werden erkannte Defizite des bestehenden RM dargelegt. Solche Defizite wurden aufgrund eigener Beobachtungen und anhand der Befragung der am Release-Prozess beteiligten Mitarbeiter hergeleitet. Nachdem die Abschnitte der Beschreibung des bestehenden RM bereits nach der ITIL-Gliederung erfolgt sind, konnten Abweichungen von ITIL leichter ermittelt werden. Diese Abweichungen werden nachfolgend dargestellt. Zunächst erfolgt im Abschnitt 4.1 die Analyse des RM im dezentralen Bereich, gefolgt von

der Analyse des RM im Bereich zS in Abschnitt 4.2 und der Analyse des RM im Host-Bereich ab Abschnitt 4.3 Mithilfe dieser Analyse werden in Abschnitt 4.4 die Anforderungen an das zu konzipierende, integrierte RM festgelegt.

4.1 Analyse des Release-Managements im dezentralen Bereich

Das RM im dezentralen Bereich unter der Steuerung und Kontrolle eines Release-Managers ist am längsten im operativen Einsatz. Dennoch gibt es auch in diesem Bereich noch Optimierungsmöglichkeiten. Der Prozess im dezentralen Bereich wurde von der Softwareentwicklung bei der HVBSystems organisatorisch eingebettet in die Betreiberorganisation bei der HVBInfo. Als der Prozess noch bei der HVBSystems angesiedelt war, lag der Fokus im SWE-Bereich auf einer begleiteten Abnahme- und Einführungsphase beim Betreiber. Durch die Reorganisation hat sich diese Sichtweise nicht geändert. Der derzeitige Release-Prozesse ist weiterhin eher ein SWE-Prozess als ein ITSM-Prozess. Im Mittelpunkt stehen die begleiteten Abnahmen im Rahmen der IT- und QS-Phase.

Im Rahmen dieser Abnahmen ist es die Hauptaufgabe des RM-C/S, die nötige Testinfrastruktur bereitszustellen und die Abnahmen und die Einführung zu organisieren und koordinieren. Abbildung 4.1 zeigt nochmals die komprimierte Darstellung des Release-, Change- und Konfigurations-Managements in ITIL im Vergleich zum Release- und Change-Management zusammen mit der Softwareentwicklung bei der HVBInfo im dezentralen Bereich.



(a) ITIL Prozesse Release- , Change- und Konfigurations-Management nach [OGC04]

(b) HVBInfo Prozesse Release- und Change-Management und Softwareentwicklung

Abbildung 4.1: Gegenüberstellung der ITIL-Prozesse und der Prozesse der HVBInfo im dezentralen Bereich

Aus dieser Gegenüberstellung fällt auf, dass es im dezentralen Bereich keine Unterstützung des RM durch das Konfigurationsmanagement gibt. In diesem Zusammenhang gibt es auch keine *CMDB* und keine *DSL*. Des Weiteren ist das Zusammenspiel mit dem Change-Management verschieden zu ITIL und die Schnittstelle zur Softwareentwicklung ist im ITIL-Ablauf ebenfalls so nicht definiert. Der Prozessablauf unterscheidet sich ebenfalls. So ist z.B. der Auslöser für die Einführung von Softwareänderungen im dezentralen Bereich die *Einmeldung*, während bei ITIL der *RfC* am Anfang des Prozesses steht. Diese Aspekte werden nachfolgend

detaillierter beschrieben. In Abschnitt 4.1.2 werden Nachteile durch das fehlende Konfigurationsmanagement beschrieben. In Abschnitt 4.1.1 werden Defizite durch die zu ITIL verschiedene Verknüpfung des RM mit dem Change-Management dargelegt und in Abschnitt 4.1.3 werden weitere allgemeine Defizite dargelegt.

4.1.1 Verbindung zum Change-Management

Bei ITIL ist der Change-Prozess für die Genehmigung von Änderungswünschen zuständig. Diese Genehmigung erfolgt vor der eigentlichen Umsetzung dieser Änderung. Bei der HVBInfo wird der Release-Prozess aus Sicht des Change-Managers jedoch als paralleler Change-Prozess aufgefasst. Das RM ist hierbei zuständig für die kontrollierte Genehmigung und Einführung von Softwareänderungen an der produktiven IT-Infrastruktur. Das Change-Management sieht sich eher für Änderungen an Hardwaresystemen verantwortlich. Die Genehmigung von Softwareänderungen durch das Change-Management wird aus Aufwandsgründen unterlassen. Aus diesem Grund erfolgt die Zusammenarbeit mit dem Change-Management erst durch Eröffnen des Pilot- bzw. Roll-out-Changes am Ende des Release-Prozesses. Diese beiden Changes signalisieren dem Change-Management die bevorstehende Änderung an produktiven Systemen.

Durch dieses im Vergleich zu ITIL Nebeneinander der Prozesse im Release- und Change-Management entstehen Instransparenzen bzw. Inkonsistenzen durch den unterschiedlichen Informationsgehalt des Change-Antrages im Vergleich zur Einmeldung. Tabelle 4.1 zeigt einen Vergleich der wichtigsten Informationen im Change-Antrag und in der Einmeldung.

Merkmal	**Change**	**Einmeldung**
Datum	Registrierungsdatum und Datum der geplanten Change-Durchführung	Einmeldedatum
Beschreibung	textuelle Beschreibung der Änderung	keine Angaben
Ursache der Änderung	Anwendungsbetreuung, Projekt, Produktionssicherung, Wartung (reines Abrechnungsmerkmal)	keine Angaben
erwartete Auswirkung	Klassifikation low, medium, high	keine Angaben
Klassifizierung	Stufen 1-3 und Emergency	keine Angaben
Status	Status des Änderungsantrages nachvollziehbar, aber nicht Status der Änderung	Status der Änderung = Stargatestatus
Hardware	Angabe des Anwendungssystemnamens und -typs, z.B. HyperPC - Windows	keine Angaben
Backup, *Recovery*	Freitext mit Informationen zu *Recovery* und *Backup*	keine Angaben
betroffene Anwendungen	Freitext	keine Angaben
Testaktivitäten	Freitext	IT- und QS-Phase
technische Attribute	keine Angaben	Produktblatt

Tabelle 4.1: Informationsvergleich von Change und Einmeldung

Bei der Einmeldung ist kein formaler Genehmigungsprozess vorgesehen. Zwar können Einmeldungen vom RM-C/S zurückgenommen werden, allerdings passiert dies nur bei z.B Überbuchung eines Releases. Dieser Fall ist in der Praxis noch nie vorgekommen. Beim Change-Prozess wird jedoch jeder Change durch festgelegte *Approver* genehmigt. Bei der Einmeldung ist im Gegensatz dazu auf Anhieb nicht ersichtlich, ob, wann und von wem die vom PV eingemeldete Änderung genehmigt wurde. Beim Change-Antrag wird als Änderungsursache zumindest ein Abrechnungskriterium, z.B. Projekt oder Wartung, angegeben. Auf dieses verrechnet der PV seine verbrauchte Arbeitszeit.

Weder beim Change-Antrag noch bei der Einmeldung wird eine Klassifikation des Änderungsgrundes vorgenommen. Durch das Fehlen der Angaben zum Änderungsgrund hat weder der Change-Manager noch der RM-C/S eine Grundlage zur Beurteilung der erwarteten Auswirkung und Dringlichkeit der geplanten Änderung,

wie es im ITIL-Vorgehen vorgesehen ist. Damit fehlt eine Entscheidungsgrundlage zur Priorisierung bei der operativen Durchführung der Softwareänderung. Der RM-C/S kann so z.B. bei einer möglichen Überbuchung oder Verzögerung nicht entscheiden, ob und welche Produkte aus einem Release ohne negative Auswirkungen entfernt werden können, um sie einem anderen Release zuzuteilen.

Aufgrund dieser Nachteile wird für das neue RM eine an ITIL orientierte Ausrichtung des Release- und Change-Managements gefordert. Die geforderte neue Verknüpfung zum Change-Management wird in Abschnitt 4.4.1.2 beschrieben.

4.1.2 Fehlendes Konfigurationsmanagement

Abbildung 4.1 verdeutlicht das Fehlen des Konfigurationsmanagements im dezentralen Bereich im Vergleich zu ITIL. In ITIL unterstützt das Konfigurationsmanagement das RM durch Informationen zu *CI*. Solche Informationen sind z.B. der Status des CI, Angaben zu verantwortlichen Personen, Abhängigkeiten zu anderen CI.

Durch diese fehlende Unterstützung sind bei der Einmeldung von Softwareänderungen keine spezifischen Angaben zu der betroffenen Hardware und zu abhängigen anderen Einheiten vorhanden. Weiterhin erfolgt die Einmeldung lediglich auf Produktebene (vgl. Release-Einheiten der HVBInfo in Abbildung 3.2 auf Seite 69). Übergeben werden allerdings oft nur die geänderten Komponenten dieser Produkte. Durch diese fehlenden Informationen in der Einmeldung arbeiten die von der Softwareänderung betroffenen technischen Stellen (SI,UV) im so genannten Überlebensmodus wie in Abschnitt 2.1.1.7.2 beschrieben . Diese technisch verantwortlichen Mitarbeiter wissen oft erst bei der physischen Softwareübergabe, welche Softwarekomponenten sie erhalten und welche Hardwareeinheiten betroffen sind. Das ist spät im Prozessablauf. Dadurch entstehen häufig ein *Backlog* bzw. Auslastungspitzen bei den jeweiligen Bearbeitern. Aus diesem Grund wird die Anpassung der Release-Einheiten an die technischen Anforderungen gefordert.

Obwohl im dezentralen Bereich die Verknüpfung der Daten der eingemeldeten Teilsysteme mit denjenigen im *Repository* vorliegt, reichen die dort gepflegten Daten für die weitere Bearbeitung nicht aus. Die Systeme und damit die Hardware im dezentralen Bereich sind noch weiter strukturiert in z.B. Filial-Server, Clientsoftware-Anteil oder Filer-Produkte. Diese Merkmale werden im *Repository* nicht gepflegt. Die geplante lokale Einführung der Datenbank zur Konfigurationsunterstützung (PARIS) zeigt das Fehlen eines *Konfigurationsmanagements* und das Fehlen einer *CMDB* mit den entsprechenden Daten.

Die Daten im *Repository* oder auch in der HyperDB sind derzeit nicht ausreichend, um den Inhalt eines Kundensets (Kundenset siehe Abschnitt 3.2.2.4) verbindlich festzulegen. Häufig wird ein Kundenset durch einfaches Kopieren eines bestehenden Kundensets zusammengestellt oder es werden Produkte, die eigentlich als AddOn für wenige Einzelarbeitsplätze gehandhabt werden müssen, manuell auf Einzelarbeitsplätzen oder komplett in der Fläche installiert. Diese manuell installierten Produkte sind dann nicht registriert. Das kann im Fall von z.B. Hardwarefehlern dazu führen, dass der PC komplett ausgetauscht wird. Der mit dem Voll-Release neu installierte PC hat dann allerdings nicht mehr alle bisherigen Applikationen des Anwenders. Bei Software dagegen, die anstelle auf einzelnen Arbeitsplätzen auf allen installiert wird, kann es zu einer Überschreitung der vorhandenen Lizenzen kommen. Dann kann u.U. dieses Softwareprodukt nicht mehr gestartet und benutzt werden. Aus diesem Grund wird im Rahmen des Konfigurationsmanagements die Einführung der Rolle des Kundensetmanagers gefordert. Er ist für die Definition des Inhaltes von Kundensets zuständig.

Aufgrund der fehlenden Informationen zu Abhängigkeiten zwischen den geänderten und unveränderten Produkten werden zu den Testaktivitäten in der IT- und QS-Phase prinzipiell alle PV mit Produkten im jeweiligen Kundenportfolio durch den RM-C/S eingeladen. Lediglich bei Hot- und Securityfixes werden nur die PV der eingemeldeten geänderten Produkte und ein kleiner Kreis von an allen Änderungen interessierten Personen zu den Tests eingeladen. Jeder PV hat häufig die Verantwortung für mehrere Produkte. Aus diesem Grund führt dieses Vorgehen zu einer hohen Anzahl von Tests je PV. Alleine für Abnahmen zu Releases des Kundensets KVZ wurden im Januar 2006 15 Einladungs- und Erinnerungsmails vom RM-C/S versandt. Nachdem manche PV mehrere Produkte betreuen, erscheinen sie teilweise überhaupt nicht zum Testen, wenn sich ihre Produkte nicht geändert haben. Sie setzen lediglich auf Aufforderung des RM-C/S den entsprechenden Status in Stargate.

Durch die fehlenden Unterstützung des Release-Pozesses durch ein Konfigurationsmanagement verlängert sich oft der jeweilige Release-Durchlauf. Beim Auftreten von Fehlern in der IT-, QS- oder Pilot-Phase kann niemand die Auswirkungen durch das Entfernen von fehlerhaften Produkten aus dem Release-Bündel abschätzen. Dadurch werden beim Auftreten von Fehlern die jeweilgen Phasen so lange verlängert oder wiederholt, bis keine Fehler mehr oder nur noch vertretbare *Known-Errors* im Release auftreten. Für das Release 4-2005 für das Kundenset KVZ erfolgten deshalb z.B. drei IT- und zwei QS-Runden.

Die Einführung eines Konfigurationsmanagements ist wegen der hier aufgezählten Nachteile notwendig. Das wird in Abschnitt 4.4.1.3 beschrieben.

4.1.3 Allgemeine Defizite im bestehenden Release-Management im dezentralen Bereich

In diesem Abschnitt werden allgemeine Nachteile des bestehenden RM dargelegt. Diese Nachteile ergeben sich zum einen durch einen von ITIL abweichenden Release-Steuerungsprozess und zum anderen aufgrund von nichtoptimalen Abläufen. Die Abweichungen vom ITIL-Release-Steuerungsprozess werden aus der Beschreibung des momentanen Status quo in Abschnitt 3.4.3 abgeleitet. Andere Defizite werden aus den Befragungen der am Prozess beteiligten Personen oder aufgrund eigener Beobachtungen hergeleitet. Probleme entstehen in diesem Bereich durch einen im Vergleich zu ITIL verschiedenen Release-Steuerungsprozess (siehe Abschnitt 4.1.3.1). Dadurch erfolgen häufig Verzögerungen im Testprozess (siehe Abschnitt 4.1.3.1.2). Weitere Nachteile entstehen durch eine unzureichende Werkzeugunterstützung des RM (siehe Abschnitt 4.1.3.2).

4.1.3.1 Unterschiede im Release-Steuerungsprozess

Nachfolgend werden Nachteile des derzeitigen RM beschrieben, die durch Unterschiede des bestehenden Release-Steuerungsprozesses im Vergleich zum Release-Steuerungsprozess in ITIL hervorgerufen werden. Die wichtigsten Unterschiede betreffen den Ablauf der Release-Planung und der Schnittstelle des Release-Prozesses zum Prozess der Softwareentwicklung.

4.1.3.1.1 Reaktive Release-Planung Ein wesentlicher Unterschied im Vergleich zu ITIL besteht im Ablauf der Release-Planung. Bei ITIL ergibt sich der Umfang bzw. Inhalt und der Termin eines Releases aus den vorliegenden, genehmigten RfC. Im dezentralen Bereich jedoch werden die Anzahl und die Termine von Standard-Releases und Sonderversorgungen im Rahmen einer Jahresplanung vorab festgelegt. Zum Zeitpunkt dieser Planungsaktivitäten ist hier jedoch noch nicht klar, welche und wie viele Änderungen in Form von Einmeldungen bis zum jeweiligen Stichtag (Einmeldeschluss) gehandhabt werden müssen. Im Vergleich zu ITIL ist das RM spät in den Änderungsprozess eingebunden. Der definitive Release-Umfang ist nicht planbar und ergibt sich erst nach dem Einmeldeschluss durch die Summe der Einmeldungen.

Die von Zarnekow zitierte Mauer zwischen Softwareentwicklung und Betreiber (vgl. Abschnitt 3.2) führt zu einer Isolation des Release-Prozesses. Durch dieses Fehlen einer proaktiven Planung arbeiten die am Release-Prozess beteiligten Mitarbeiter reaktiv. Erkennbar ist das daran, dass sich die Mitarbeiter im RM selbst als „Feuerwehrtruppe“ bezeichnen, die jeweils auftretende „Brände“ in Form von spät mitgeteilten Änderungen im Release-Prozess löschen müssen. Offensichtliche reaktive Aktionen reichen von einer angedachten Beschränkung der Maximalanzahl der Einmeldungen bis hin zur Einführung von Verschalungskategorien (siehe Abschnitt 3.4.2.1.2). Beides sind Reaktionen auf die bestehende *Backlog*-Situation. Um diese Situation zu ändern, wird in Abschnitt 4.4.1 die Orientierung des RM an ITIL gefordert.

Die derzeitige Release-Planung wird mit Excel-Listen durchgeführt. Allerdings wird die Darstellung der Planungsübersicht in Excel schnell unübersichtlich, wenn neben den Abnahmen und Abnahmeumgebungen des dezentralen und Host-Bereiches auch noch Umgebungen für ca. 100-150 zS berücksichtigt werden müssen. Aus diesem Grund wird in Abschnitt 4.4.4.2.1 ein Werkzeug zur Planungsunterstützung gefordert.

4.1.3.1.2 Unklare Schnittstelle zwischen Release-Prozess und Softwareentwicklung Als Schnittstelle zwischen dem Softwareentwicklungsprozess und dem Release-Prozess ist die so genannte Theke

festgelegt. Allerdings wird diese Schnittstelle von den Softwareentwicklern so oft nicht wahrgenommen. Der Grund hierfür liegt in der historischen Entwicklung des RM.

Im ursprünglichen Produktionseinführungsprozess der HVBSystems endete die Realisierungsphase mit dem Ende der QS-Phase (vgl. Abbildung 3.29 auf Seite 106). Im dezentralen Bereich soll die Realisierungsphase allerdings nach der Einmeldephase abgeschlossen sein. Diese Umstellung haben die Softwareentwickler teilweise noch nicht nachvollzogen. Das hat eine zeitliche Ausdehnung der Entwicklungsaktivitäten und damit eine späte, endgültige Produktübergabe zur Folge. Die vom RM-C/S stets eingeplante zweite IT-Phase verleitet die Entwickler zusätzlich, erst in der zweiten IT-Phase die fertige Softwareversion bereitzustellen. Daraus resultiert wiederum eine für die zweite IT-Phase zu hohe Fehleranzahl, wie Abbildung 3.23 auf Seite 98 verdeutlicht. Das führt dann nochmals zu einer Verlängerung der Testaktivitäten. Durch einen Gewöhnungseffekt rechnen die Entwickler dann schon von vornherein mit einer Verlängerung der Testphasen und stellen somit ihre Software noch später fertig.

Weitere Nachteile durch diese langen Durchlaufzeiten aufgrund der verlängerten Testphasen und aufgrund der hohen Fehlerzahlen (siehe nächster Abschnitt) entstehen durch aufgedeckte Fehler (*Known Errors*), die zeitlich nicht mehr durch Nachlieferungen innerhalb des momentanen Release-Durchlaufes behoben werden können. Da derzeit niemand fehlerhafte Produkte aus einem Release entfernt, weil wie oben beschrieben die Auswirkungen auf andere Produkte nicht abschätzbar sind, entsteht oft schon während des Release-Durchlaufes die Notwendigkeit zur Durchführung weiterer Hotfixes. Mit diesen Hotfixes sollen diese Known Errors behoben werden. Beim ersten Release im Jahr 2006 für das Kundenset KVZ hat sich z.B. im Release-Durchlauf bereits die Notwendigkeit von drei weiteren Hotfixes ergeben, obwohl das Release zu diesem Zeitpunkt noch nicht in der Fläche bei den Kunden verteilt war.

Aus diesen Gründen wird in Abschnitt 4.4 die Definition einer klaren und einheitlichen Schnittstelle zwischen der Softwareentwicklung und dem RM gefordert.

4.1.3.1.3 Nachteile durch die bestehenden Testabläufe

Der momentane Testablauf ist phasenweise und orientiert sich am Testablauf des V-Modells. Spillner [Spil 03] kritisiert an diesem Verfahren die spät beginnenden Testaktivitäten, die unklare Verbindung von der Testaktivität zur Testbasis und eine undeutliche Trennung von Test-, Debug- und Änderungsaktivitäten. Er schläg als Abhilfe das so genannte **W-Modell** vor (siehe Abbildung 4.2). Beim W-Modell beginnt der Testprozess früher und damit parallel zur Entwicklung.

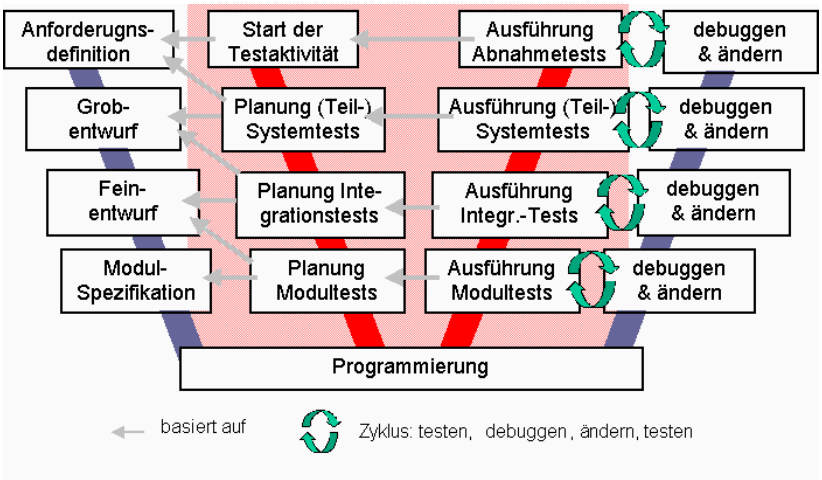


Abbildung 4.2: W-Modell aus [Spil 03]

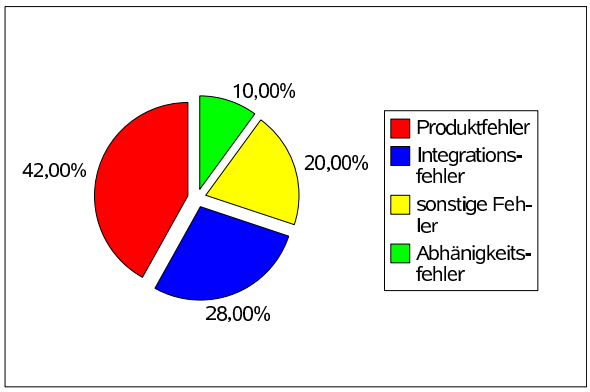


Abbildung 4.3: Fehlertypen und Häufigkeit bei Releases 2005 (KVZ)

Ein weiterer Grund für Verlängerungen oder Wiederholungen der Testphasen sind zum einen eine geringe Testbeteiligung und zum anderen hohe Fehlerzahlen im Testverlauf. Diese geringe Testbeteiligung, verbunden mit einer häufig späten Softwarefertigstellung, führt oft zu einer Verlängerung des Testprozesses.

Im Release 1-2006 des Kundensets MCV lag die Testbeteiligung in der ersten IT-Phase bei 25%. Alleine hierfür wären vier Iterationen nötig, um eine 100%ige Testbeteiligung zu erhalten. Aufgrund hoher Fehlerzahlen wurden im Jahr 2005 bei allen vier durchgeführten Standard-Releases beim Kundenset KVZ der Release-Prozess um eine zweite QS-Phase verlängert. Abbildung 4.3 zeigt die Fehlerverteilung in der ersten IT-Phase im Jahr 2005, bezogen auf das Kundenset KVZ. Produkt- und Integrationsfehler stellen hierbei bereits 70% der Fehler dar.

Nachteilig am derzeitigen Testablauf ist auch, dass die Entwickler ihre eigene Software testen. Nach Zeller ist derjenige, der ein Produkt definiert, entwirft und implementiert, am schlechtesten für die destruktive Betrachtung der Ergebnisse seiner Arbeit geeignet [Zell]. Damit die Testaktivitäten zukünftig früher beginnen, ein zusätzliches 4-Augen-Prinzip eingeführt wird und die Qualitätssicherung unabhängig ist, wird die Einführung eines unabhängiges Testteams gefordert (siehe Abschnitt 4.4.2). Die Tester dieses Testteams tragen zur Entlastung der PV bei, indem sie sie bei der Erstellung von Testfällen unterstützen und indem sie anstelle der PV die Tests durchführen. Dadurch können zukünfigt wie im W-Modell beschrieben die Testaktivitäten deutlich früher beginnen und die Qualitätssicherung ist unabhängig.

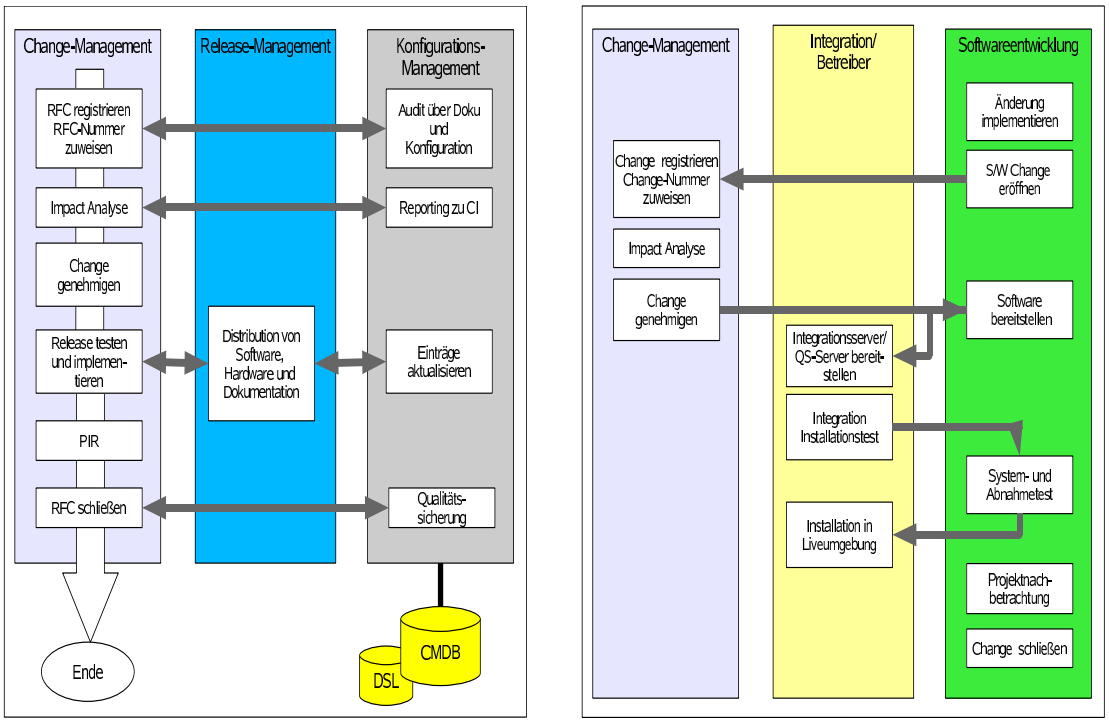
4.1.3.2 Unzureichende Workflow-Unterstützung

Optimierungspotenzial im derzeitigen RM besteht durch die unzureichende Unterstützung des RM durch ein geeignetes Werkzeug. Heute arbeitet der RM-C/S mit Stargate, Excel, Outlook und Project. Stargate auf Basis von PVCS als Werkzeug zur Unterstützung des RM-C/S wurde bei der HVBSystems ausgewählt, als das RM noch dort angesiedelt war. Der damalige Fokus lag im Bereich der Softwareversionierung zur Unterstützung des SCM. Der eigens hinzuprogrammierte Stargate-Workflow unterstützt den RM-C/S nur unzureichend. Der durch das Release-Begleittblatt (siehe Abbildung 3.18 auf Seite 92) definierte Prozessablauf sieht beispielsweise nur eine einzige IT- und eine QS-Phase vor. Vorher geplante oder neue Iterationen der IT- oder QS-Phase können so durch Stargate nicht gesteuert werden. Ebenfalls erlaubt Stargate nicht die Festlegung kundenspezifischer Release-Durchläufe.

Die Auswirkung der unzureichenden Workflow-Unterstützung zeigt sich auch im Führen zusätzlicher Listen. Der RM-C/S führt eine Excel-Checkliste zur Koordination der Aktivitäten des Release-Durchlaufes, er erstellt eine Liste neuer Produkte für das Lizenzmanagement, er führt eine Excel-Liste zur Störungsdokumentation am Infopoint oder eine Liste der zu verschalenden Produkte für die Verschaler usw. Aufgrund der ungenügenden Werkzeugunterstützung wird in Abschnitt 4.4.4.2 hierfür eine Verbesserung gefordert.

4.2 Analyse des Release-Managements im Bereich zentraler Server

In diesem Abschnitt erfolgt die Analyse der RM im Bereich zS. Ausgangslage für diese Analyse ist wiederum das RM aus ITIL. Nachfolgend werden sowohl Defizite im Vergleich zum ITIL-Release-Management beschrieben, die sich aufgrund der erkennbaren Unterschieden des im Abschnitt 3.5 beschriebenen bestehenden RM ergeben als auch Nachteile, die sich durch eigene Beobachtungen oder durch Befragung von am RM beteiligten Personen ergeben haben. Ein wesentliches Defizit im Bereich zS entsteht, weil das RM hier ein reiner SWE-Prozess ist. Das RM entspricht hier einer durch den Projektleiter bzw. Softwareentwickler



(a) ITIL Prozesse Release- , Change- und Konfigurations-Management nach [OGC04]

(b) HVBInfo Prozesse Change-Management, Integration und Softwareentwicklung bei zS

Abbildung 4.4: Gegenüberstellung der ITIL-Prozesse und der Prozesse der HVBInfo im Bereich zS

begleiteten Einführungs- und Abnahmephase. Wie Abbildung 4.4 verdeutlicht, ergeben sich hier Unvollständigkeiten durch das gänzliche Fehlen eines Release-Managements (siehe Abschnitt 4.2.1). Das Fehlen des Konfigurationsmanagements tritt auch hier auf. Es wird deshalb nicht mehr explizit beschrieben.

Im Vergleich zum dezentralen Bereich gibt es eine Verknüpfung mit dem Change-Management in dem Sinne, dass jede Änderung vorab genehmigt ist. Allgemeine Defizite treten hier im Release-Prozess durch ein unkoordiniertes Umgebungsmanagement (siehe Abschnitt 4.2.2.1), durch ungenügende technische Standards (siehe Abschnitt 4.2.2.2 und durch das Fehlen eines zentrales Qualitätsmanagements (siehe Abschnitt 4.2.2.3) auf.

4.2.1 Fehlendes Release-Management

Im Bereich zS gibt es bis auf wenige Ausnahmen im Windows-Umfeld (vgl. Abschnitt 3.5) kein RM innerhalb der HVBInfo. Die Einführung von Softwareänderungen erfolgt durch Projektleiter bzw. PV der HVBSystems. Dieser steuert sein jeweiliges Entwicklungs- oder Wartungsprojekt bis zur Einführung der Software in die Produktion. Aufgrund des Fehlens eines RM gibt es hierfür auf Seiten der HVBInfo keine definierten Prozesswege und keine definierten Kommunikationswege. Deshalb erfahren die am Prozess beteiltigten Stellen oft erst im Rahmen des Change-Prozesses, dass eine Änderung durchzuführen ist. Von den ca. 950 Changes im Bereich Windows zS im Jahr 2005 mussten 12% am selben Tag durchgeführt werden. Lediglich bei 44% der Changes bestand eine Vorlaufzeit von mehr als fünf Tagen, d.h. nur hier haben die Integratoren und Umgebungsverantwortlichen entsprechende Vorbereitungs- und Planungsmöglichkeiten. Hierbei ergeben sich, wie im dezentralen Bereich bereits beschrieben, Nachteile durch eine reaktive Arbeitsweise. Wegen des Fehlens des RM in diesem Bereich werden ab Abschnitt 4.4.1 die Ausrichtung an ITIL und einheitliche Schnittstellen gefordert.

4.2.2 Allgemeine Defizite im bestehenden Release-Management im Bereich zentraler Server

In diesem Abschnitt werden Defizite im bestehenden RM beschrieben. Sie beruhen auf eigenen Beobachtungen oder auf durchgeführten Befragungen der Mitarbeiter im RM bzw. der am Prozess beteiligten Mitarbeiter. Nachteile im Prozessablauf zur Einführung von Softwareänderungen ergeben sich durch ein unkoordiniertes Umgebungsmanagement (siehe Abschnitt 4.2.2.1), durch ungenügende technische Standards (siehe Abschnitt 4.2.2.2) oder durch das Fehlen eines zentralen Qualitätsmanagements (siehe Abschnitt 4.2.2.3).

4.2.2.1 Unkoordiniertes Umgebungsmanagement

Die Beantragung und Bereitstellung von für Tests und Integrationsaufgaben benötigte Umgebungen erfolgt unkoordiniert. Der jeweilige Projektleiter bzw. PV beauftragt die Bereitstellung der entsprechenden Umgebungen für die IT- und QS-Phase. Bei umfangreichen Projekten geschieht dies häufig schon koordiniert über eine rechtzeitige Beantragung durch den Projektleiter. Im Vergleich dazu erhält der SI bei kleineren Wartungaktivtäten durch den PV erst kurzfristig Kenntnis von den bevorstehenden IT- und QS-Aktivitäten. Hier beantragt der PV die Bereitstellung der Umgebungen „auf Zuruf“ oder der SI erhält erst über einen neuen Change-Antrag die entsprechenden Informationen. Wenn der Change-Antrag erst kurz vor der Produktionseinführung eröffnet wird, muss der SI dann kurzfristig mit der Bereitstellung der entsprechenden Umgebung reagieren. Oft verbleiben ihm nur wenige Tage.

Da es kein plattformübergreifendes Umgebungsmanagement gibt, konkurrieren die verschiedenen Prozesse um die zS Ressourcen. Das betrifft neben den notwendigen Testumgebungen auch die Mitarbeiter, die zur Durchführung von Änderungen benötigt werden. Verschiedene zentrale Server werden sowohl im Rahmen der dezentralen Releases vom RM-C/S für die Abnahmen beantragt, als auch für die Batch-Abläufe im Host-Prozess benötigt. Des Weiteren werden bei durch den RM-zS gesteuerten zS-Corereleases und Securityfixes ebenfalls zS-Umgebungen benötigt. Es gibt für den Einsatz und die Verfügbarkeit der zS keine zentrale Übersicht, welche zS für dezentrale Abnahmen bzw. Batch-Abläufe im Host-Bereich benötigt werden und im welchem Status (z.B. Wartung, bereit, Fehler) sie sich gerade befinden. Aus diesem Grund wird hier die Einführung eines plattformübergreifenden Umgebungsmanagements gefordert (siehe Abschnitt 4.4.3). Eine geregelte Beauftragung der Umgebungsbereitstellung durch PRUEV analog der Workflow-Unterstützung des Host-Bereiches ist hierfür bei der HVBInfo bereits spezifiziert, allerdings noch nicht umgesetzt.

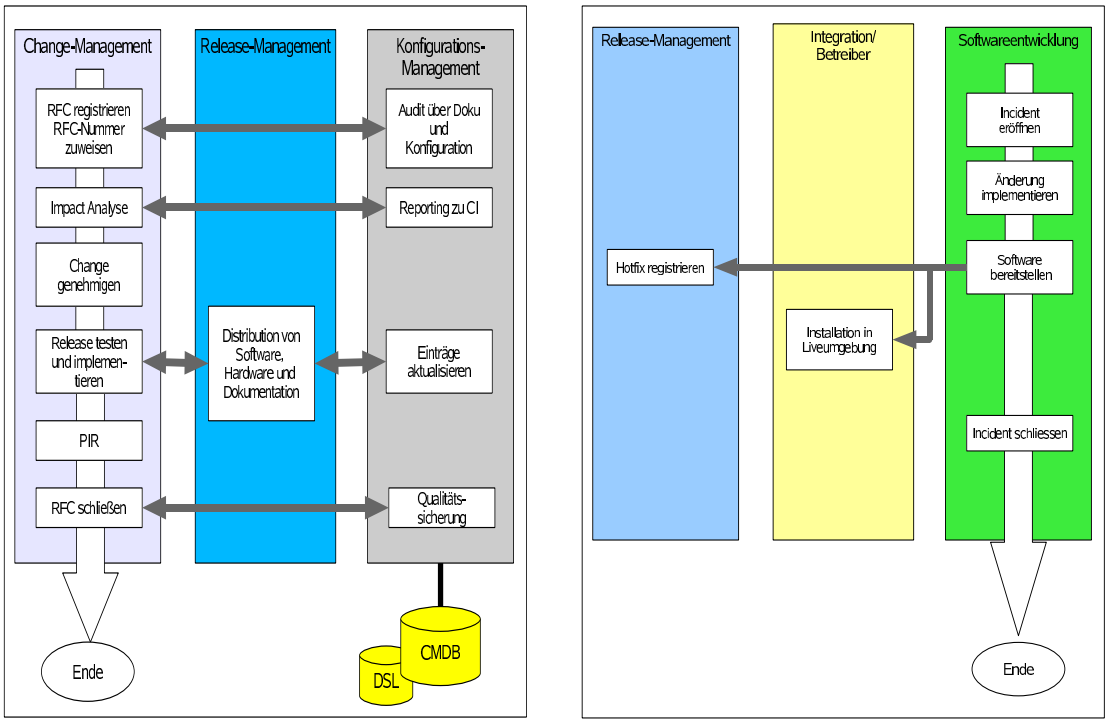
Nachdem die gleiche Hardware im Bereich zS teilweise für die IT- und die QS-Phase eingesetzt wird, wird zur Beseitigung von evtl. Engpässen die Umstellung auf virtuelle Umgebungen gefordert (siehe Abschnitt 4.4.3.2).

4.2.2.2 Ungenügende technische Standards

Nachteile im Bereich zS entstehen aufgrund nicht durchgesetzter technischer Standards, insbesondere bei den technischen Depyloyment-Aktivitäten. Es gibt hier verschiedene Verfahren für die Verteilung und Installation von Releases (vgl. Tabelle 3.10 auf Seite 110). An eine Konsolidierung zu einem einzigen Verteilverfahren wird derzeit noch nicht gedacht. Eine einheitlicher Standard hier bietet jedoch die Ausgangslage zur Automatisierung der Deployment-Prozesse, wie es heute im Host-Bereich schon umgesetzt ist (vgl. z.B. Abschnitt 3.6.2.6). Zumindest sollen die noch bestehenden manuellen Verfahren umgestellt werden. Deshalb wird in Abschnitt 4.4.4.4 die Einführung von technischen Standards gefordert.

4.2.2.3 Zentrales Qualitätsmanagement fehlt

Im dezentralen Bereich nimmt das RM auch Qualitätsmanagementaufgaben wahr. Nachdem der Prozess bei den zS jeweils aus dem Entwicklungsprojekt gesteuert wird, gibt es hier kein zentrales Qualitätsmanagement. Die jeweiligen Testabläufe und die Fehlerdokumentation im Testprozess werden individuell gehandhabt. Teilweise erfolgt die Dokumentation der Tests nur über *Logfiles*. Dieses individuelle Vorgehen erlaubt keine umfassende statistische Auswertung als Grundlage für den KVP. Die Analyse von Fehlerverläufen und Fehler-



(a) ITIL Prozesse Release- , Change- und Konfigurations-Management nach [OGC04]

(b) Übersicht Prozessablauf im Host-Bereich bei Hotfixes

Abbildung 4.5: Gegenüberstellung der Prozessabläufe im Zusammenspiel mit Release- und Change-Management, Softwareentwicklung und Integration im Host-Bereich

klassifikation wäre nur durch mühsames einzelnes Zusammentragen der Daten möglich. Aus diesem Grund wird in Abschnitt 4.4.2 die Einführung eines zentralen Qualitätsmanagements gefordert.

4.3 Analyse des Release-Managements im Host-Bereich

Nachfolgend wird das RM im Host-Bereich analysiert. Das RM im Host-Bereich ist derzeit im Entstehen und orientiert sich am RM des dezentralen Bereiches. Auch hier soll das RM an ITIL ausgerichtet werden. Durch die Orientierung des neuen Ablaufes am Ablauf des dezentralen Bereiches ergeben sich die gleichen Defizite wie im obigen Abschnitt beschrieben. Sie werden deshalb nicht nochmals aufgeführt.

Die Mehrzahl der Softwareänderungen wird derzeit immer noch über den so genannten Hotfix-Prozess eingeführt. Aus diesem Grund ergeben sich Defizite durch die noch nicht erfolgte Umstellung auf den durch den RM-Host gesteuerten Release-Prozess. Wie Abbildung 4.3 verdeutlicht, steuert der Projektleiter bzw. AWSV die Einführung von geänderter Software. Im Vergleich zum Ablauf im Bereich zS jedoch erfolgen hier keine Tests in einer IT- oder QS-Phase. Die Software wird direkt von der Entwicklung in die Produktion verbracht.

Nachdem sich die Einführung des Host-RM immer noch in einer Umstellungsphase befindet, können die Projektleiter bzw. AWSV selbst entscheiden, mit welchem Verfahren sie ihre Produkte in die Liveumgebung bringen. Über den Standard-Release-Prozess werden derzeit lediglich vier der ca. 690 AWS abgewickelt. Die Teilnahme an diesem Standard-Release-Prozess ist noch freiwillig. Es gibt auch bis jetzt keine definitive Aussagen, ab wann dieser Prozess zum Standard erklärt wird.

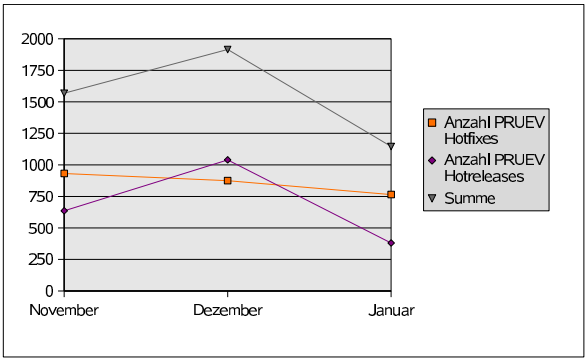


Abbildung 4.6: Anzahl der PRUEVs in Hotrelease und Hotfix

Abbildung 4.6 zeigt die Anzahl der per Hotfix- und Hotrelease-Prozess in die Produktion verbrachten Änderungen als Summe der PRUEV-Aufträge. Hotfixes werden erst seit Mitte Oktober vom RM-Host registriert. Deshalb liegen erst ab November 2005 vergleichbare Zahlen vor. Die Anzahl der PRUEV-Aufträge durch Hotfixes unterscheidet sich im Durchschnitt nicht wesentlich von der Anzahl der PRUEV-Aufträge in den Hotreleases. Ein Hotfix entspricht im ITIL-Sinne eher dem Emergencyrelease, wobei hier für die Legitimation des Hotfixes lediglich das Vorhandensein eines registrierten Incidents gefordert wird. Die Summe von 500-1000 Hotfixes je Monat ist relativ hoch.

Das derzeitige integrierte Verfahren von KVS, PRUEV und RM-Tool-Host unterstützt allerdings noch nicht alle PRUEV-Auftragsarten. Deshalb entspricht die obige PRUEV-Anzahl nur ca. 60-70% der Gesamtzahl. Nichtregistrierte KVS-Packages werden *native*, d.h. ohne jegliche Überprüfung bzw. Registrierung durch das RM in die Produktion gebracht.

Aufgrund der hohen Anzahl der momentan noch nicht über den durch das RM definierten Release-Prozess eingeführten Softwareänderungen wird in Abschnitt 4.4.1 die definitive Umstellung auf den durch das RM definierten Prozess gefordert. Defizite entstehen im Host-Bereich hauptsächlich aufgrund der direkten Überführung von Software von der Entwicklung in die Produktion. Das ist im nachfolgenden Abschnitt beschrieben.

Fehlendes Qualitätsmanagement
Wie im Bereich zS fehlt auch hier ein zentrales Qualitätsmanagement. Die Qualität der erfassten Daten ist noch nicht ausreichend, weil das RM hier erst in der Einführung ist. Eine stichprobenartige Untersuchung der Einträge der Incident-Nummern in PRUEV hat Unregelmäßigkeiten ergeben. Teilweise werden offensichtliche Fantasienummern (z.B. „1234567“ oder „0000001“) eingetragen. Für diese gibt es keinen entsprechenden *Incidentrecord*. Bei manchen Einträgen wird eine vorhandene Incident-Nummer durch Verwendung in mehreren PRUEV-Aufträgen wiederholt wiederverwendet. Es erfolgt hier momentan noch keine Überprüfung bzw. automatisch Verknüpfung der Daten.

Sowohl bei Hotfixes als auch bei Hotreleases werden Host-Komponenten direkt von der Entwicklungsebene in die Produktionsebene verbracht. Das Fehlen von vorherigen Tests in einer produktionsnahen Umgebung hat Iterationsschleifen, die im dezentralen Berich in der IT- oder QS-Phase auftreten, über die Produktivumgebung zur Folge. Bei einer internen Untersuchung im Jahr 2003 wurde festgestellt, dass im Zeitraum März bis April von 16 600 Übergaben per KVS 5 600 innerhalb kurzer Zeit wiederholt wurden. Das entspricht einer Nachlieferungsquote von ca. 34%. Im Vergleich dazu erfolgen diese Iterationen im dezentralen Bereich bzw. bei zS durch Nachlieferungen in den Testumgebungen der IT- und QS-Phase.

In der gleichen Studie wurde bei einer Nachbetrachtung eines großen Projektes festgestellt, dass durch umfangreiche Tests in der IT- und QS-Phase alleine bei diesem Projekt 344 *Incidents* in der Testphase aufgetreten sind. Diese konnten dann vor der Produktionseinführung behoben werden. Bis jetzt erfolgen diese Tests lediglich bei Standard-Releases und Sonderversorgungen in der IT- und QS-Phase in der GPL. Dieser Aspekt wird auch in Abbildung 3.6.4 auf Seite 125 verdeutlicht. Auch hier werden, bezogen auf die geringe Anzahl von AWS, eine relative hohe Anzahl der Fehler in der IT-Phase gefunden.

Nach einer Auswertung der täglich in die Produktion übergebenen KVS-Packages liegt der Anteil der darin als kritisch eingestuften Komponenten bei deutlich über 50%. Vor diesem Hintergrund ist es bedenklich, dass diese KVS-Packages nicht vorab einer Abnahme analog dem dezentralen Bereich vorgenommen werden. Deshalb wird auch hier ein zentrales Qualitätsmanagement gefordert, das die Art und den Umfang der durchzuführenden Tests vor Produktionseinführung festlegt (siehe Abschnitt 4.4.2).

4.4 Ableitung von Anforderungen an das zu konzipierende, plattformübergreifende Release-Management

Durch die im vorherigen Abschnitt durch die Analyse des derzeitigen RM gefundenen Defizite werden die Anforderungen an das integrierte RM abgeleitet. Diese Anforderungen werden nachfolgend erläutert. Aufgrund der neuen Orientierung des RM an ITIL wird die Umstellung des derzeitigen SWE-orientierten Release-Verfahrens zu einem ITSM-orientierten Release-Verfahren nötig (siehe Abschnitt 4.4.1). Daraus ergeben sich auch weiterhin eine einheitliche Festlegung der Prozessgrenzen (siehe Abschnitt 4.4.1.1), eine andere Verbindung zum Change-Management (siehe Abschnitt 4.4.1.2) und die Einführung eines Konfigurationsmanagements analog ITIL (siehe Abschnitt 4.4.1.3. Aufgrund der im vorigen Abschnitt beschriebenen Defizite wird die Einführung eines zentralen Qualitätsmanagements (siehe Abschnitt 4.4.2) gefordert. Zur besseren Unterstützung des plattformübergreifenden Release-Steuerungsprozess wird das RM durch ein plattformübergreifendes Umgebungsmanagement (siehe Abschnitt 4.4.3), durch neue, unterstützende Werkzeuge und Standards unterstützt. Diese Anforderungen ergeben sich aus einer im Vergleich zum plattformspezifischen RM gesteigerten Komplexitäts des integrierten RM.

4.4.1 Orientierung des RM an ITIL

Das gesamte RM wird an ITIL orientiert. Daraus resultiert eine komplette Umstellung des bisherigen am SWE ausgerichteten RM zu einem RM nach ITSM. Aufgrund der verschiedenen Ausprägungen des bestehenden RM im Bereich C/S, zS und Host ergeben sich hierfür unterschiedliche Konsequenzen.

Im dezentralen Bereich gibt es bereits ein RM, während es sich im Host-Umfeld noch im Pilotstadium befindet und im Bereich zS überwiegend noch nicht existiert. Deshalb ist die Einführung der Rolle des plattformübergreifenden Release-Managers nötig. Weiterhin werden, wie nachfolgend beschrieben, die Prozessgrenzen einheitlich festgelegt. Daraus resultiert auch die an ITIL orientierte Verbindung zum Change-Management und die Einführung des Konfigurationsmanagements.

Für den Host-Bereich resultiert daraus die definitive Einführung des momentan pilothaft stattfindenen RM. Im Bereich zS müssen hierfür ebenfalls die notwendigen Voraussetzungen durch die Einführung des RM-zS für alle Arten von Releases gelegt werden. Diese ITIL-Orientierung erfolgt nicht nur aufgrund der in Abschnitt 2.4.2 zitierten Vorteile, sondern auch, weil bestehende Prozesse der HVBInfo wie etwa Incident-, Change- und Problemmanagement bereits an ITIL ausgerichtet sind. Dadurch kann das neue Release-Management ohne wesentliche Schnittstellenprobleme in die bestehende Prozessinfrastruktur integriert werden. Für diese Prozessintegration sind nachfolgende Aspekte relevant.

4.4.1.1 Einheitliche Festlegung der Prozessgrenzen

Für den neuen plattformübergreifenden Prozess werden die Prozessgrenzen neu definiert. Insbesondere das Ende der Softwarerealisierungsphase wird für alle Plattformen einheitlich festgelegt. Der Softwareentwicklungsprozess endet dann entsprechend der Beschreibung des vorgestellen Lebenszyklusmodells (vgl. Abbildung 2.3 auf Seite 10) mit dem Unit-Test. Abbildung 4.7 zeigt die Prozessgrenzen der bestehenden und des neuen Release-Prozesses.

Nach der Entwicklung erfolgt die technische Übergabe der geänderten Software an der Theke. Anschliessend beginnt die Integrationsphase. Sie bildet zusammen mit dem erfolgten Integrationstest eine eigene Phase. In ITIL erfolgen die hierfür relevanten Tätigkeiten wie Release-Zusammenstellung und -Konfiguration in der Entwicklungsumgebung (vgl. Abbildung 2.61 auf Seite 47). Hier wird jedoch vom ITIL-Vorgehen abgewichen. Die für die Integrationsaufgaben nötige, produktionsähnliche Hardware befindet sich bei der HVBInfo.

Diese neue Integrationsphase wird im Vergleich zur derzeitigen IT-Phase vom nachfolgenden Abnahmeprozess zeitlich entkoppelt. Dargestellt ist das in der Abbildung durch die zweite gestrichelte Linie. Gründe hierfür sind zum einen die häufigen Iterationen dieser Phase im derzeitigen dezentralen Prozess. Bei Fehlern einzelner Produkte in der IT-Phase werden alle anderen fehlerfreien Produkte blockiert. Zum anderen dauert die Integration bei zentralen Serverprodukten bei bestehenden Produkten zwischen ein und sechs Tagen, bei komplexen

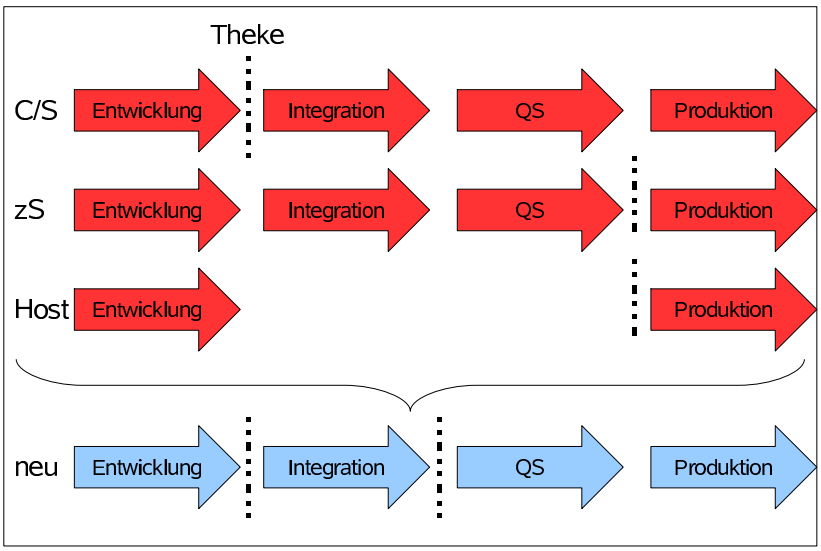


Abbildung 4.7: Bestehende und neue Prozessgrenzen

oder neuen Anwendungen kann die Integration mehrere Wochen dauern. Eine übergreifende Synchronisation der plattformspezifischen IT-Phasen ist somit schwer zu realisieren.

Die Integrationsphase ist somit nicht mehr im Fokus des eigentlichen Release-Steuerungsprozesses. Die Bereitstellung der nötigen Integrationsumgebungen erfolgt durch das Umgebungsmanagement (siehe Abschnitt 4.4.3) und die Testaktivitäten erfolgen unter Steuerung des Qualitätsmanagements (siehe Abschnitt 4.4.2). Am Ende dieser Phase hat jede Release-Einheit den Status der Freigabe für die QS- bzw. Abnahmephase. Ohne diese Freigabe dürfen mit diesem Produkt keine weiteren Prozessphasen durchlaufen werden. Dokumentiert wird dieser Status durch einen Eintrag des Produktes in die *DSL* (siehe Abschnitt 4.4.1.3.2). Durch die zeitliche Entkopplung der Integrationsphase kann dieser Status jedoch frühzeitig erworben werden. Eine Statusvergabe „auf Halde“ ist möglich. Die Produkte werden dann in „Pufferlagern“ zwischengespeichert. Im dezentralen Bereich wird an dieser Umsetzung durch das Projekt ReQuest bereits gearbeitet. Das Pufferlager wird dort als Hochregallager bezeichnet. Aus diesem wird ähnlich einem Fertigungsprozess ein Release als Bündel von $1..n$ Produkten kommisioniert. Im neuen RM wird hierfür die DSL eingeführt.

Der Release-Steuerungsprozess beginnt analog dem Release-Prozess in ITIL mit der Festlegung des Release-Umfanges. In Zukunft wird, ausgehend von genehmigten Änderungen, der Release-Inhalt festgelegt. Hierfür ist auch im Vergleich zum heutigen Prozess eine andere Verknüpfung mit dem Change-Management nötig. Das wird in nachfolgenden Abschnitt beschrieben. Durch diese Änderung hat das RM die Möglichkeit einer proaktiven Planung. Das RM reagiert dann nicht mehr auf Produkte, welche über die Mauer bzw. Theke geschoben werden, sondern legt proaktiv den Release-Inhalt fest.

Die weitere QS-Phase bzw. Abnahmephase entspricht der ITIL-Abnahmephase. Jedes einzelne Produkt muss die entspreichende Freigabe in der vorherigen Integrationsphase erreicht haben. Aus all diesen freigegebenen Produkten kann dann das Release zusammengestellt werden. In dieser Phase erfolgen dann operative Tests, Akzeptanztests und Geschäftsprozesstests. Unterstützt wird diese Phase durch das Qualitäts- und Konfigurationsmanagement.

4.4.1.2 Verknüpfung mit dem Change-Management

Die Verknüpfung mit dem Change-Prozess erfolgt analog ITIL. So wie im Bereich zS bereits gehandhabt, wird durch den Projektleiter bzw. PV oder AWSV für jede fachliche Programmänderung ein Change (*RfC*) eröffnet. Im dezentralen Bereich und im Host-Bereich geschieht das anstelle der bisherigen Einmeldung. Allerdings wird zukünftig der Change vor Beginn der Entwicklungsaufgaben eingestellt. In diesem Change wird zur Planungsunterstützung des RM der Fertigstellungstermin, d.h. das voraussichtliche Ende der neuen Integrationsphase eingetragen. Der Release-Umfang wird dann aus der Menge der Change-Anträge zusammen-

gestellt. Das RM bündelt hierfür die entsprechenden Change-Anträge zu Releases und plant aufgrund der eingetragenen Daten die Abnahmephase und Roll-out-Termine.

Durch diese Änderung im Change-Prozess ergeben sich auch für die Rollen PV bzw. AWSV und EM andere Tätigkeiten. Der PV bzw. AWSV ist zukünftig für die Beantragung der Softwareänderung über einen Change zuständig. Das EM wird bereits in den Genehmigungsprozess involviert. Dadurch kann das EM zukünfigt seine Aktivitäten, wie etwa die Auswahl von Pilotarbeitsplätzen oder Schulungsaktivitäten, früher planen.

Die für die technischen Subprozesse, wie etwa Verteilung oder Installation, nötigen Spezifikationen werden in der *CMDB* (siehe Abschnitt 4.4.1.3) angegeben. Hierfür ist auch eine Verknüpfung mit der derzeitigen Beauftragung (PRUEV) (siehe Abschnitt 4.4.1.3.1) und der CMDB nötig. Das Konfigurationsmanagement verwaltet die Einträge der CMDB. Relevant sind hier z.B. Installationsanleitungen etc. Diese Einträge ersetzten das bisherige Produktblatt.

Der Change-Prozess und auch die Beauftragung werden durch das Konfigurationmanagement unterstützt. Hiermit sind dann z.B. Angaben zu abhängigen Anwendungen ersichtlich oder technische Merkmale zur Installationsprozedur vermerkt. Durch diese neue Verknüpfung des Change-Prozesses werden oben beschriebene Intransparenzen vermieden. Durch dieses Verfahren steigt allerdings der Genehmigungsaufwand beim Change-Management. Allerdings wird die bisherige Einmeldung zu einem Release unnötig, da das RM für die Zuordnung der Changes zu einem Release zuständig ist. Die Changes für Softwareänderungen werden zusätzlich eingeteilt in korrigierende, adaptive oder perfektible Änderungen (vgl. Abschnitt 2.1.1.7 auf Seite 11). Durch diese zusätzliche Klassifikation kann der KVP unterstützt werden, durch Einleitung gegeigneter Maßnahmen, wenn z.B. die Anzahl der korrigierenden Changes eines AWS sehr hoch ist.

4.4.1.3 Einführung des Konfigurationsmanagements

Im Rahmen der Ausrichtung des neuen RM an ITIL wird die Einführung des Konfigurationsmanagements nötig. Die Unterstützung des Konfigurationsmanagements ist sowohl für das RM als auch für das Change-Management nötig. Das Change-Management kann dadurch die Auswirkungen von Changes auf abhängige Einheiten überprüfen. Das RM bzw. Qualitätsmanagement kann die Testaktivitäten besser steuern. Somit müssen z.B. nicht mehr prinzipiell und wie bisher alle PV im dezentralen Bereich zum Testen erscheinen, sondern es werden gezielt für diejenigen Produkte Tests veranlasst, bei denen Abhängigkeiten bestehen. Der in Abschnitt 2.1.1.7 beschriebene *ripple effect* kann so ausgeschlossen werden. Des Weiteren kann das RM die Auswirkung beim Entfernen von Produkten aus einem Release-Bündel abschätzen. So braucht im Fehlerfall der Abnahmeprozess nicht mehr bis zum Erfolgsfall wiederholt werden, sondern der Release-Durchlauf wird termingerecht fortgesetzt und das fehlerhafte Produkt kann aus dem Release entfernt werden.

Die Festlegung des Inhaltes von Kundensets ist eine Teilaktivität des Konfigurationsmanagements. Aus diesem Grund wird neben der Rolle des Konfigurationsmanagers auch ein Kundensetmanager für diese Aktivität eingeführt. Er ist dann der zentrale Ansprechpartner des RM oder des Lizenzmanagements zu Fragen der bei den Kunden eingesetzten Software.

Durch die Einführung des Konfigurationsmanagements werden auch das Umgebungsmanagement und die Integration unterstützt. So kennen das Umgebungsmanagement und auch die Verschaler vorab die Zuordnung der Software zur benötigten Hardware und nicht erst bei der physischen Softwareübergabe. Die derzeit definierten PRK (vgl. Abschnitt 3.5.2.1.1 auf Seite 104) und andere erwarteten Begleitdokumente entsprechen CI und werden ebenfalls vom Konfigurationsmanagement in der CMDB verwaltet. Wie nachfolgend beschrieben, muss im Rahmen der Einführung des Konfigurationsmanagements sowohl eine CMDB als auch eine DSL eingeführt werden.

4.4.1.3.1 CMDB Die durch das Konfigurationsmanagement verwaltete CMDB enthält alle notwendigen Informationen zu CI. Das RM benötigt hiervon eine geeignete Strukturierung der Release-Einheiten, die derzeit im *Repository* nicht abgebildet ist. Des Weiteren ist die Zuordnung der jeweiligen Release-Einheit zur benötigten Hardware notwendig. Auch das wird derzeit im *Repository* nicht ausreichend abgebildet. Die genaue Spezifikation der CMDB wird hier nicht weiter detailliert. Das ist die Aufgabe weiterer Arbeiten. Ziel soll hier sein, die derzeit bestehenden Informationsquellen (*Repository*, KVS, HyperDB etc.) effizient zu verknüpfen.

4.4.1.3.2 DSL Die DSL dient analog ITIL der sicheren Aufbewahrung von autorisierter Software. Die genaue Spezifikation der DSL wird hier nicht weiter detailliert. Eine DSL kann hier als Abstraktion der physischen Aufbewahrungsorte eingeführt werden. Im Host-Bereich eignet sich vermutlich das KVS-System als DSL oder im Bereich C/S die HyperDB. Im Bereich zS muss evtl. eine eigene Datenbank hierfür eingeführt werden. Allerdings ist die Untersuchung der Verwendbarkeit der bestehenden Datenspeicher als DSL bzw. ihre Einführung die Aufgabe weiterführender Arbeiten.

4.4.2 Qualitätsmanagement

Die Rolle des Qualitätsmanagements (QM) wird weiter ausgebaut. Derzeit nimmt das RM vor allem im dezentralen Bereich zusätzlich Qualitätsmanagementaufgaben wahr. Im Bereich zentraler Server und im Host-Bereich ist das QM weniger stark ausgeprägt. Aus diesem Grund ist ein zentrales QM nötig. Seine Aufgaben im Release-Prozess sind die Festlegung und Überprüfung von Test- und Testinhalten in der Integrations- und in der Abnahmephase.

Der Nachweis einer erfolgten Testdurchführung wird derzeit nicht nachdrücklich gefordert (vgl. Abschnitt vorher 4.1.2). Der RM fordert hier lediglich das Setzten des Status in Stargate. Aus diesem Grund ist das QM zuständig für die Festlegung der Art und des Umfangs der Tests und auch der Kontrolle der Einhaltung der festgelegten Kriterien. Der Nachweis der Testdurchführung kann der RM in der Abnahmephase anhand von Anwesenheitslisten kontrollieren. Auch in der Integrationsphase ist eine Dokumentation und Auswertung des Testverlaufs nötig. Eine Möglichkeit zur automatischen Verknüpfung des Ergebnisses der Testdurchführung mit dem entsprechenden Status im Release-Prozess besteht z.B. durch den Einsatz des *Testdirectors*. Der Testdirector ist eine Testfalldatenbank der Firma Mercury [Merc 06]. Er wird bei der HVBSystems als Testfalldatenbank eingesetzt und ermöglicht die Definition von Testfällen, die Durchführung von Tests, die Verwaltung von Testergebnissen und die Verwaltung und Auswertung von Fehlern. Der Testdirector ist bei der HVBSystems etabliert und wird bereits heute im Rahmen der Abnahmen benutzt. Die Untersuchung zur Eignung des Testdirektors für die Überwachung der Tests sowohl der Integrations- als auch der Abnahmephase ist die Aufgabe weiterführender Arbeiten.

Zur operativen Unterstützung des QM wird das unabhängige Testteam eingeführt. Es führt nach Beauftragung durch das QM, des PV oder des Kunden Tests aller Art durch, insbesondere auch Tests von Geschäftsprozessen. Die Beauftragung des Testteams kann über das bestehende Beauftragungswerkzeug (PRUEV) erfolgen. Diese muss für diese Anforderung allerdings erweitert werden. Diese Erweiterung ist nicht Bestandteil dieser Arbeit.

4.4.3 Umgebungsmanagement

Zur Unterstützung des integrierten RM wird ein plattformübergreifendes Umgebungsmanagement eingeführt. Das Umgebungsmanagement arbeitet zudem eng mit dem QM und Konfigurationsmanagement zusammen. Vertreten wird es durch die neue Rolle des plattformübergreifenden Umgebungsmanagers. Die Aufgaben des Umgebungsmanagers entsprechenden den bereits bestehenden im dezentralen Bereich. Der Umgebungsmanager ist der zentrale Ansprechpartner des RM für den Status von Testumgebungen. Für die Integrationsphase koordiniert er die Bereitstellung der entsprechenden Integrationsumgebungen, für die Abnahmephase die entsprechenden Abnahmeumgebungen. Die jeweiligen Umgebungsspezifikationen werden in der CMDB dokumentiert.

4.4.3.1 Geregelte Beauftragung

Das im Host-Bereich etablierte PRUEV-Verfahren zur standardisierten Kommunikation und Dokumentation des Übergabeprozesses wird auf den dezentralen Bereich und den Bereich zS erweitert. Die Beauftragung entspricht einem Werkzeug zur Workflow-Unterstützung. Im Rahmen der Beauftragung werden zukünftig im dezentralen Bereich z.B. Verschalungstermine vereinbart, im Bereich zS die Umgebungsbereitstellung für die Integration veranlasst oder im Host-Bereich Batch-Abläufe festgelegt. Für den Bereich zS ist die Beauftragung durch PRUEV bereits spezifiziert, aber noch nicht implementiert. Für den dezentralen Bereich muss

dieses noch erfolgen. Das ist wiederum die Aufgabe weiterführender Arbeiten. Somit kann z.B. der Integrator über eine Beauftragung beim UV die Bereitstellung der entsprechenden Integrationsumgebung veranlassen. Ebenso haben Softwareentwickler die Möglichkeit, für Integrationsaufgaben und -tests eine produktionsnahe Entwicklungsumgebung zu beantragen. Das RM beauftragt über dieses standardisierte Verfahren den Aufbau der QS-U. Diese Beauftragung regelt somit den Kommunikations- und Informationsfluss und trägt so zu effizienteren Abläufen bei.

4.4.3.2 Virtualisierte Umgebungen

Im dezentralen Bereich ist durch das Projekt ReQuest bereits die Virtualisierung insbesondere von Integrationsumgebungen geplant. Diese virtuellen Umgebungen sollen den Entwicklern die Möglichkeit einer frühzeitigen Integration in produktionsnahen Umfeld ermöglichen. Virtualisierte Umgebungen sollen auch im Bereich zS eingesetzt werden. Hierdurch wird besonders der Engpass entschärft, der durch das gleichzeitige Verwenden derselben Hardware in der IT- und QS-Phase entsteht.

4.4.4 Anforderungen aufgrund der höheren Komplexität des integrierten Release-Managements

In diesem Abschnitt werden Anforderungen definiert, die sich nicht aufgrund der im vorherigen Abschnitt beschriebenen Defizite des plattformspezifischen RM ableiten lassen. Die nachfolgend festgelegten Anforderungen ergeben sich aufgrund einer im Vergleich zum plattformspezifischen RM höheren Komplexität des integrierten RM. Aufgrund dieser höheren Komplexität ist wie nachfolgend beschrieben eine Anpassung der Taktung, eine bessere Werkzeugunterstützung, die Einführung plattformübergreifender Rollen oder auch die Einführung von Standards nötig.

4.4.4.1 Anpassung der Taktung

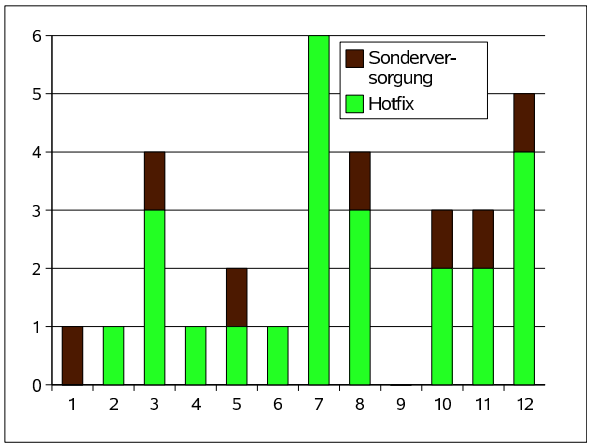


Abbildung 4.8: Anzahl der dezentralen Hotfixes und Sonderversorgungen je Monat im Jahr 2005

Die bisherige zeitplangetriebene Release-Planung ist auch weiterhin sinnvoll. Im dezentralen Bereich wird die Hardware für Abnahmen aller Kunden eingesetzt. Die Reservierung im Rahmen der Jahresplanung erleichtert die Einteilung und Abstimmung von Releases für alle Kunden.

Allerdings sollte die Taktung in Zukunft kundenspezifisch angepasst werden. Aus Abbildung 4.8 ist ersichtlich, dass im dezentralen Bereich für das Kundenset KVZ z.B. neben den Standard-Releases jeden Monat Releases in Form von Hotfixes oder Sonderversorgungen durchgeführt wurden. Diese können bei einer höheren Taktung, z.B. zwölf pro Jahr, im Rahmen von monatlichen Standard-Releases mit abgewickelt werden. Des Weiteren reduzieren sich die derzeitigen Aufwandspitzen bei den Verschalern, wenn anstatt von derzeit durchschnittlich fünf Standard-Release-Terminen in Zukunft zwölf durchgeführt werden.

Bei einer höheren Taktung ist auch die Integration der bisherigen Releases im Host-Bereich einfacher. Hier muss die Release-Frequenz bei diesem Kunden beispielsweise reduziert werden. Die bisher wöchentlich durchgeführten Hotreleases sollen sukzessive auf eine monatliche Taktung reduziert werden.

Im Bereich zS können im Rahmen dieser monatlichen Taktung ebenfalls die derzeitigen Corereleases und monatlichen Securityfixes mit integriert werden. Applikations-Releases, deren Produktionseinführung bisher

am Ende des jeweiligen Entwicklungsprojektes stattfand, werden möglichst auch in diese Standard-Releases integriert.

4.4.4.2 Werkzeugunterstützung

Für das integrierte RM müssen neue Werkzeuge eingeführt bzw. bestehende Werkzeuge geändert werden. Die detaillierte Beschreibung der Spezifikation dieser Werkzeuge ist kein Bestandteil dieser Arbeit. Zur Unterstützung des neuen RM ist ein Werkzeug zur Planungsunterstützung, zur Unterstützung des Release-Durchlaufes und für das Umgebungsmanagement nötig.

4.4.4.2.1 Werkzeug zur Planungsunterstützung Ein neues Werkzeug zur Planungsunterstützung soll den bisherigen Ablauf der Planung ändern. Mithilfe dieses Werkzeuges können vom RM sowohl plattformspezifische als auch plattformübergreifende Releases eingeplant werden. Diese Planung soll dann die Anforderungen beteiligter Stellen (Integration, Umgebungsmanagement) berücksichtigen. So kann z.B. ein Release durch Bündelung verschiedener RfC eingeplant werden. Das Umgebungsmanagement erhält dann automtisch einen Auftrag zur Bereitstellung der entsprechenden Umgebung, falls diese zur Verfügung steht. Falls sich die benötigte Umgebung in einer Wartungsphase befindet, wird durch das Werkzeug automatisch ein geeigneter Release-Termin vorgeschlagen.

Durch dieses Planungswerkzeug wird die derzeitige Planung mit Excel und Project abgelöst. Sie ist für die plattformübergreifende Planung nicht mehr handhabbar, weil alleine das Berücksichtigen aller zS zu einer deutlichen Steigerung der zu managenden Abnahmeumgebungen führt. Das kann in Excel nicht mehr übersichtlich dargestellt werden.

4.4.4.2.2 Werkzeug zur Unterstützung des Release-Durchlaufes Die bisherigen Werkzeuge zur Unterstützung des Release-Durchlaufes (Stargate und RM-Tool-Host) sind für ein plattformübergreifendes RM ungeeignet. Aus diesem Grund ist die Einführung eines geeigneten Werkzeuges nötig. Dieses wird auch hier nicht weiter spezifiziert. Das neue Werkzeug soll das RM in der Steuerung und Kontrolle des Release-Durchlaufes unterstützen. Weiterhin sollen flexible und kundenspezifische Release-Durchläufe damit handhabbar werden. Durch die Verwendung eines geeigneten Werkzeuges fallen die momentanen Tätigkeiten der Listenführung weg.

4.4.4.2.3 Werkzeug zur Unterstützung des Umgebungsmanagements Für das plattformübergreifende Umgebungsmanagement ist ebenfalls ein eigenes Werkzeug nötig.

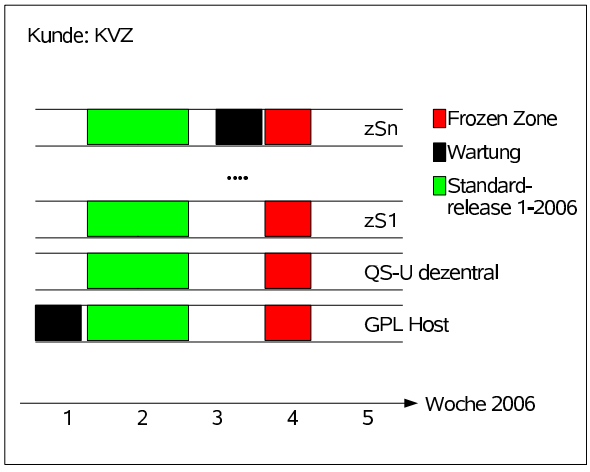


Abbildung 4.9: Skizze Darstellung toolunterstützte Umgebungsbelegung

Die jeweiligen Belegungszeiten der entsprechenden Umgebung bzw. Hardware durch Wartung, Aufbau, Datenbereitstellung oder Testphasen soll dadurch dargestellt und dokumentiert werden. Dieses Umgebungsmanagementtool unterstützt das RM bei der Planung der Abnahmephasen, indem es dort die entsprechenden Verfügbarkeiten einsehen kann. Abbildung 4.9 skizziert die mögliche Darstellung der Umgebungsbelegung. Voraussetzung ist auch hier die Unterstützung durch das Konfigurationsmanagement. Dadurch können die jeweilig für einen Kunden bzw. für ein Release benötigten Umgebungen zugeordnet werden. Für die Einplanung eines neuen Releases, z.B. eines ad-hoc auftretenden Emergencyfixes, kann der RM dann z.B. durch *Drag-and-Drop* für den entsprechenden RfC die benötigte Umgebungen reservieren. Über eine entsprechende Integration des Workflowtools würden die erforderlichen Prozessbeteiligten automatisch informiert werden.

4.4.4.3 Einführung von plattformübergreifenden Rollen

Zur Reduzierung des Koordinations- und Kommunikationsaufwandes werden derzeit bestehende Rollen plattformspezifisch ausgelegt. Abbildung 4.10 zeigt auf der linken Seite die Koordinations- und Kommunikati-

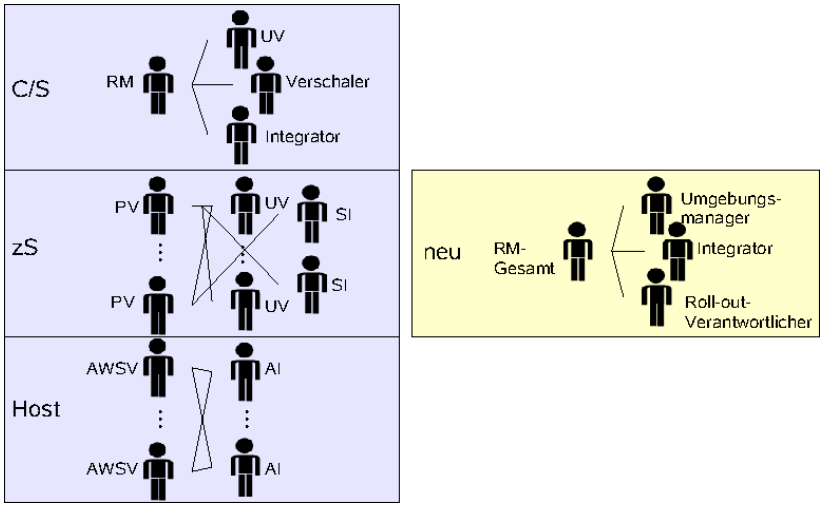


Abbildung 4.10: Gegenüberstellung bestehender und neuer Kommunikationsbeziehungen

onsbeziehungen beim derzeitigen RM. Aus Übersichtsgründen sind hier nur einige Rollen exemplarisch dargestellt. Auf der rechten Seite sind die zukünftigen plattformübergreifenden Rollen dargestellt. Durch diese plattformübergreifenden Ansprechpartner vereinfachen sich zukünftig die Kommunikations- und Koordinationsaufgaben für das RM. Als plattformübergreifend werden die Rollen Umgebungsmanager, Integrator und des Roll-out-Verantwortlicher eingeführt. Sie dienen dem RM oder auch anderen am RM beteiligten Stellen als zentrale Ansprechpartner. So ist der Umgebungsmanager z.B. für Fragen zu allen Umgebungen zuständig. Diese kann er natürlich an die weiterhin bestehenden Umgebungsverantwortlichen weiterleiten. Der Integrator koordiniert die plattformspezifischen Integrationsaufgaben, die auch weiterhin durch den Systemintegrator oder Anwendungsintegrator erfolgen. Der Roll-out-Verantwortliche ist zuständig für die plattformspezifischen Aktivitäten des Softwareversorgers oder Einsatzverantwortlichen. Anstelle von z.B. derzeit n Ansprechpartnern bei den Umgebungsverantwortlichen zu den n zentralen Servern hat das RM dann nur noch einen. Bei den Integratoren und Roll-out-Verantwortlichen gilt diese Reduzierung von Kommunikationsbeziehungen analog.

4.4.4.4 Einführung von Standards

Für die Release-Prozesse werden Standards verbindlich eingeführt. Das betrifft vor allem technische Standards bei den Deployment-Tätigkeiten, aber auch die Einführung einheitlicher Begriffe.

Durch die Einführung des Paketierungsstandards *MSI* von Microsoft können z.B. im dezentralen Bereich die Verschaler bei ihren Aufgaben entlasten werden. Im Bereich zS muss weiterhin überprüft werden, ob das Staging-Verfahren oder die Installation und Verteilung durch Tivoli der zukünftige Standard wird. Eine Bereitstellung standardisierter Entwicklungsumgebungen, analog dem HyperPC als Standardarbeitsplatz, hilft weiterhin Arbeitsaufwände bei z.B. Installationstätigkeiten zu reduzieren. Durch die Festlegung verbindlicher Betreiberstandards wird die Grundlage für weitere Automatisierung gelegt. Verdeutlicht werden kann das am Beispiel des Package- und Deployment-Standards KVS im Host-Bereich. Dort sind die Deployment-Aktivitäten bereits heute weitgehend automatisiert.

Im Rahmen der Standardisierung werden auch die verwendeten Begriffe, z.B. bei den Release-Arten, vereinheitlicht. Dadurch verbessert sich das Verständnis der am RM beteiligten Mitarbeiter füreinander.

4.5 Zusammenfassung

In diesem Kapitel wurden die bestehenden plattformspezifischen Prozesse analysiert. Die derzeitigen Release-Prozesse haben ähnliche Abläufe in den Phasen Entwicklung - IT - QS und Roll-out. Unterschiede bestehen in den Zuständigkeiten und den Schnittstellen zwischen Entwicklerorganisation und dem Release-Management. Im dezentralen Bereich startet der Release-Prozess unter Steuerung des RM-C/S nach der Entwicklungsphase. Ebenso bei den zentralen Servern im Windows-Umfeld für Core-Produkte und Securitypatches. Bei den zentralen Servern für Windows-Applikationen und im UNIX-Bereich ist kein RM definiert. Der Prozess wird vom jeweiligen Softwareentwicklungsprojektleiter bzw. Produktverantwortlichen gesteuert. Die offizielle Softwareübergabe erfolgt aus Projektleitersicht nach der QS-Phase. Im Host-Bereich ist das RM derzeit in der Einführung. Aus diesem Grund wird ein kleiner Teil der Releases wie im dezentralen Bereich vom RM-Host gesteuert, der andere Teil analog im Bereich zS vom Projektleiter.

Aufgrund der verschiedenen Ausprägungen der bestehenden Release-Prozesse resultieren als Anforderungen die Einführung eines plattformübergreifenden Release-Managers zur Steuerung und Kontrolle des neuen Prozesses und die Festlegung einheitlicher Prozessschnittstellen. Die IT-Phase wird im plattformübergreifenden Prozess von den restlichen Phasen zeitlich entkoppelt. Grund hierfür sind die unterschiedlich langen Integrationszeiten, die von wenigen Stunden bei bekannten Produkten im dezentralen Bereich bis zu mehreren Wochen bei der Einführung neuer zS-Anwendungen reichen können.

Ähnlich sind sich die bestehenden Prozesse im Verständnis des RM als beigleitete Abnahme- und Einführungsphase, wie sie in Abschnitt RM als Bestandteil des SWE (2.2.6.8) beschrieben wurde. Die Hauptaktivitäten sind die verschiedenen Testaktivitäten in den entweder vom RM beauftragten oder vom Projektleiter beantragten Testumgebungen und die Koordination von Deployment-, Schulungs- bzw. Trainingsaktivitäten. Zukünftig wird nur noch der RM für die Koordination dieser weiterhin bestehenden Abnahmephase verantwortlich sein. Unterstützt wird er hierbei durch die neue Rolle des plattformübergreifenden Umgebungsmanagers und Integrators. Diese neuen Rollen koordinieren die jeweils plattformspezifischen Aufgaben. Die Testumfänge und -inhalte werden von der ebenfalls neu ausgerichteten Rolle des Qualitätsmanagers festgelegt. Er definiert zusammen mit dem Kunden die notwendigen Abnahmekriterien geänderter Software. Weiterhin legt er die Art und den Umfang von Tests fest. Er kann im Notfall auch das gänzliche Weglassen von Testaktivitäten entscheiden, z.B. bei als fatal eingestuften Sicherheits-Updates.

Die bestehende SWE-Ausrichtung der Release-Prozesse wird zu einer an ITIL orientierten ITSM-Ausrichtung geändert. Dazu ist zum einen die Einbindung des Change-Managements bei der Einführung von Softwareänderungen nötig, zum anderen wird das Konfigurationsmanagement zur Unterstützung des Change- und Release-Managements neu eingeführt. Durch die Einbindung des Change-Managements unterliegen in Zukunft alle Softwareänderungen dem Genehmigungsprozess durch das Change-Management. Derzeit wird das lediglich bei Applikationsreleases im zS Umfeld so gehandhabt. Durch diese Genehmigungsprozedur kann der RM zukünftig den definitiven Release-Umfang wesentlich früher als heute anhand der beantragten Changes festlegen. Dadurch können die entsprechenden Ressourcen (Integratoren, Verschaler, Umgebungen etc.) rechtzeitig reserviert, beauftragt und informiert werden. Als Folge reduzieren sich die derzeit bei diesen Ressourcen auftretenden Auslastungsspitzen. Das Konfigurationsmanagement wirkt hierbei unterstützend, indem das Change-Management die Auswirkungen mithilfe der Informationen in der CMDB beurteilen kann. Weiterhin hat das Umgebungsmanagement die Informationen zu Umgebungsspezifikationen, die für eine Software benötigten werden.

Zur Unterstützung des gesamten Release-Steuerungsprozesses ist eine Werkzeugunterstützung durch ein entsprechendes Werkzeug nötig. Auch das Umgebungsmanagement benötigt ein Werkzeug für Einträge zum jeweiligen Status (Wartung, reserviert, in Aufbau etc.) aller im Release-Prozess benötigten Umgebungen. Das Konfigurationsmanagement muss die Inhalte der CMDB definierten und das Release-Management die der DSL.

Die in diesem Kapitel festgelegten Anforderungen an den plattformübergreifenden Prozess werden im nachfolgenden Kapitel zu seiner Definition aufgegriffen. Die Einführung des Konfigurationsmanagements wird hier im Rahmen des neuen Prozesses gefordert, aber nicht weiter beschrieben. Ebenso erfolgen keine näheren Spezifikationen der geforderten Werkzeuge und unterstützenden Datenbanken.

5 Plattformübergreifendes Release-Management

Inhaltsverzeichnis

5.1 Beteiligte Rollen und ihre Aufgaben 150
5.2 Release-Aktivitäten 154
5.2.1 Festlegen der Releasepolicy und des Release-Plans 154
5.2.1.1 Festlegen der Releasepolicy 155
5.2.1.2 Festlegen des Release-Plans 158
5.2.2 Entwurf, Release-Zusammenstellung und -Konfiguration 159
5.2.3 Testen und Abnahme des Releases 159
5.2.3.1 Geschäftsprozesse testen 160
5.2.3.2 Testauftrag 160
5.2.4 Planung des Einsatzes 160
5.2.5 Kommunikation, Vorbereitung und Schulung 160
5.2.6 Release-Verteilung und Installation 161
5.2.7 Beschreibung des Release-Steuerungsprozesses 161
5.2.7.1 Change-Zuordnung 162
5.2.7.2 Integrationsphase 162
5.2.7.3 QS-Phase 163
5.2.7.4 Pilot 164
5.2.7.5 Roll-out 165
5.2.8 Bewertung des Release-Durchlaufes 165
5.3 Einordnung in das Lebenszyklusmodell 167
5.4 Vorteile durch das integrierte Release-Management 168
5.5 Zusammenfassung 168

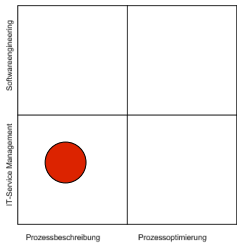


Abbildung 5.1: Einordnung des neuen Prozesses

In diesem Kapitel wird das erarbeitete Konzept des integrierten Release-Managements dargestellt. Unter Beachtung der im vorherigen Kapitel definierten Anforderungen und des ITIL-Release-Managements als weiterer Grundlage wurde das vorliegende Konzept entwickelt. Durch dieses neue Release-Management können zukünftig sowohl plattformübergreifende als auch plattformspezifische Releases bei der HVB-Info gesteuert werden. Die Gliederung in diesem Abschnitt folgt wiederum der ITIL-Gliederung.

Nachfolgend werden die am plattformübergreifenden Release-Management beteiligten Rollen mit ihren Aufgaben beschrieben. Der geänderte Release-Steuerprozess ist in Abschnitt 5.2.7 dargestellt. Abbildung 5.2 skizziert das Zusammenspiel des plattformübergreifenden RM mit dem Change-, Konfigurations- und Umgebungsmanagement und der Softwareentwicklung im neuen Gesamtprozess. Neu im Vergleich zum bestehenden RM ist das Konfigurationsmanagement analog ITIL. Dadurch verschiebt sich der Fokus des RM zukünftig von der SWE-Sichtweise zur ITSM-Sichtweise und ist damit wie in Abbildung 5.1 dargestellt links unten eingeordnet. Die Schnittstelle zum Change-Management wurde analog dem Change-Management in ITIL (vgl. Abbildung 2.58 auf Seite 44) angepasst. Im Vergleich zu ITIL ist die Abbildung 5.2.7 um die Softwareentwicklung und das Umgebungsmanagement ergänzt. In der ITIL-Abbildung ist das nicht dargestellt, weil ITIL, wie in Abschnitt 2.4.2 auf Seite 62 bereits dargelegt wurde, keine Anleitung zur Implementation der jeweiligen Prozesse liefert. Für die Implementation des RM sind die Einheiten Softwareentwicklung und Umgebungsmanagement allerdings nötig.

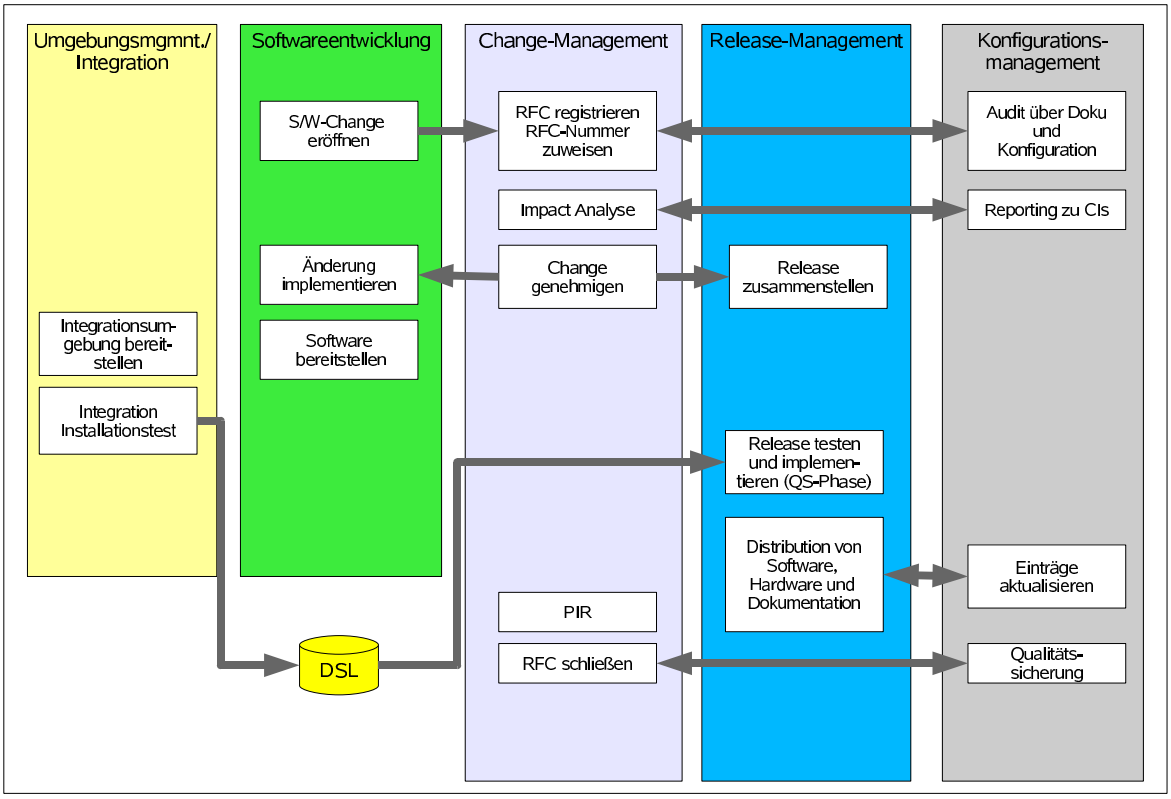


Abbildung 5.2: Übersicht RM mit Change-, Konfiguration-, Umgebungsmanagement und Softwareentwicklung im Gesamtprozess

5.1 Beteiligte Rollen und ihre Aufgaben

In diesem Abschnitt werden die beteiligten Rollen beschrieben. Tabelle 5.1 stellt die jeweiligen Aufgaben dieser Rollen dar. Der Release-Manager koordiniert und steuert den gesamten Release-Prozess. Hierfür kommuniziert und interagiert er mit folgenden Rollen. Die unterstrichenen Rollen sind hierbei neue Rollen bzw. Rollen mit verändertem Aufgabengebiet. Die relevanten Rollen sind: Auftraggeber, Change-Manager, Einführungsmanager (EM), Endanwender, Entwickler, Fachverantwortlicher, Integrator, Konfigurationsmanager, Kunde, Kundensetmanager, Lizenzmanager, Produktverantwortlicher (PV) bzw. Anwendungssystemverantwortlicher (AWSV), Qualitätsmanager, Projektleiter und Teilprojektleiter, Roll-out-Verantwortlicher , Tester, Umgebungsmanager (UM) und die so genannte *Security*. Aus ITIL sind jedoch lediglich die Rollen Release-Manager, Change- und Konfigurations-Manager abgeleitet. ITIL, wie in Abschnitt 2.4.2 zitiert, beschreibt Prozesse, lässt aber Raum für die eigene Implementierung dieser Prozesse. Aus diesem Grund ergibt sich die Notwendigkeit der verbleibenden Rollen für die Implementierung des Prozesses. Diese Rollen sind weitestgehend aus den bestehenden Rollen abgeleitet. Veränderungen im Profil der einzelnen Rollen sind in der Tabelle (5.1) beschrieben. Das Release-Management-Gremium (RMG) wird hier ebenfalls als Rolle aufgeführt. Es entspricht aber einem Forum aus am Prozess beteiligter Rollen, das je nach Bedarf vom Release-Manager einberufen werden kann.

Rolle	Aufgaben	Bemerkungen
Auftraggeber	• beauftragt bzw. genehmigt Softwareentwicklung	wird nach Genehmigung i.d.R. vom Fachverantwortlicher vertreten; unveränderte Rolle und Aufgabengebiet

Rolle	Aufgaben	Bemerkungen
Change-Manager	• registriert Changes • genehmigt Changes	der Change-Manager ist zukünftig auch für die Genehmigung von Softwareänderungen zuständig; dieser Change erfolgt dann je Softwareänderungen anstelle der bisherigen Einmeldung
Einführungsmanager (EM)	• koordiniert die Pilotierung • koordiniert Schulungsaktivitäten des Endanwenders • genehmigt Changes	neu hier ist die Genehmigung von Changes; dadurch wird das EM frühzeitig von bevorstehenden Änderungen informiert und kann seine Aktivitäten dementsprechend steuern; diese Information erfolgte früher im Rahmen der Einmeldung
Endanwender	• arbeitet mit dem eingeführten System	unverändert
Entwickler	• realisiert die in den Anforderungsdokumenten beschriebenen Softwaresysteme	unverändert
Fachverantwortlicher	• vertritt Auftraggeber und den späteren Anwender • überprüft Berücksichtigung der fachlichen und organisatorischen Anforderungen • fachliche Abnahme geänderter oder neuer Software	unverändert
Integrator	• definiert Anforderungen an die Zielumgebung • integriert übernommene Software zu Versorgungseinheiten • koordiniert die Integrationsphase • testet die Integration • testet die Installation	zusätzlich zu den bestehenden plattformspezifischen Integratoren gibt es den plattformübergreifenden; er koordiniert die jeweiligen plattformspezifischen System- und Anwendungsintegratoren und ist der zentrale Ansprechpartner des RM (vgl. Anforderung in Abschnitt 4.4.4.3)
Konfigurationsmanager	• Aufgaben analog ITIL • unterstützt RM durch Informationen zu Release-Einheiten • unterstützt Change-Management	diese Rolle wird neu eingeführt (vgl. Anforderung in Abschnitt 4.4.1.3)
Kunde	• beauftragt und genehmigt Softwareänderung • arbeitet mit den eingeführten Systemen	wird durch Auftraggeber, Endanwender und Fachverantwortlichen vertreten; unverändert

Rolle	Aufgaben	Bemerkungen
Kundenset-manager	• definiert Inhalt von Kundensets • genehmigt Changes bei neuen Produkten	neue Rolle, gehört organisatorisch zum Konfigurationsmanagement (siehe Anforderung in Abschnitt 4.4.1.3)
Lizenz-manager	• genehmigt Changes bei neuen Produkten	Teilnahme am Genehmigungsprozess dient der Information des Lizenzmanagements über Einführung neuer Produkte; anstelle des bisherigen Listenversandes durch den RM
Projekt-leiter	• träg Verantwortung für Entwicklungsprojekt • eröffnet Change vor Entwicklungsbeginn	kann bei kleineren Entwicklungsaufgaben auch Entwickler bzw. PV sein, bei großen wird er durch Teilprojektleiter unterstützt; der Change muss zukünftig verbindlich vor Beginn der Entwicklungstätigkeiten erfolgen, heute ist der Zeitpunkt der Change-Eröffnung nicht festgeschrieben; siehe auch Anforderung in Abschnitt 4.4.1.2
Produkt-verant-wortli-cher (PV)	• übernimmt das System zur Betreuung und Wartung	Aufgaben analog Projektleiter, wenn Wartungsprojekt; unverändert
Qualitäts-manager (QM)	• definiert und prüft Tests und Testinhalte in Integrations-, QS- und Pilotphase • definiert Qualitätskennzahlen	gab es bisher nur im dezentralen Bereich, jetzt für alle Plattformen; siehe Anforderung in Abschnitt 4.4.2
Release-Manager	• genehmigt Changes • plant Releases • legt Release-Umfang fest • koordiniert QS-Phase • koordiniert Roll-out-Phase	jetzt plattformübergreifend; neu ist die Genehmigung von Changes anstelle der Einmeldung und die Release-Planung anhand der genehmigten Changes
Release-Management Gremium	• stimmt übergreifende Aspekte ab (z.B. bei Ressourcen-Planung) • wird bei Bedarf vom Release-Manager einberufen	die Zusammensetzung ist individuell, abhängig von Problemstellung
Roll-out-Verant-wortlicher	• erstellt Verteilungsplan • informiert *Serviceline* über bevorstehenden Roll-out • verteilt und installiert Releases in der Zielumgebung • betreibt PÜ • dokumentiert und meldet Störungen	fungiert als zentraler Ansprechpartner des RM zu Fragen zum Roll-out,, vgl. Anforderung in Abschnitt 4.4.4.3

Rolle	Aufgaben	Bemerkungen
Security	• legt Sicherheitsrichtlinien fest • überprüft Einhaltung der Securityrichtlinien	unverändert
Tester	• führt auf Auftrag Tests durch	neu aufgrund der Anforderung nach einer unabhängigen Qualitätssicherung (siehe Abschnitt 4.4.2)
Umgebungsmanager (UM)	• stellt Umgebungen bereit • konfiguriert die Zielumgebung • betreibt diese Umgebungen	Aufgabe wie bisher, allerdings plattformübergreifender UM zur Koordination der plattformspezifischen UV und als zentraler Ansprechpartner des RM, vgl. Anforderung in Abschnitt 4.4.4.3

Tabelle 5.1: Rollen und Aufgaben beim integrierten Gesamtprozess

Im Vergleich zu den bisherigen Rollen gibt es sowohl neue als auch Rollen mit veränderten oder neuen Aufgaben. Die Aufgaben des EM werden um die Genehmigung von Changes erweitert. Indem er bereits zu einem frühen Zeitpunkt von der geplanten Softwareänderung erfährt, kann er frühzeitig seine Aufgaben planen. Er koordiniert auch weiterhin alle Aktivitäten zur Pilotierung und Schulung und bildet somit die Schnittstelle zwischen RM und Kunden. Für die Pilotierung wählt er z.B. eine geeignete Pilotfiliale oder einen geeigneten Pilotarbeitsplatz aus. Dieser Prozess kann abgeschwächt werden, indem der EM lediglich von jedem Change in Kenntnis gesetzt wird, anstatt ihn genehmigen zu müssen. Das kann u.U. auch AWS-spezifisch oder projektabhängig gehandhabt werden.

Neue Rollen sind der Konfigurations- und Kundensetmanager. Der Konfigurationsmanager ist zuständig für das Konfigurationsmanagement analog ITIL (vgl. Abschnitt 2.3.4). Er definiert und registriert CI und dokumentiert Änderungen an diesen. Mit den Einträgen in der CMDB unterstützt er sowohl das Change- als auch das Release-Management. Das Change-Management kann damit die Auswirkung von Änderungen einfacher beurteilen. Das RM hat Informationen zu Abhängigkeiten zwischen CI und zu den benötigten Umgebungsvoraussetzungen für Abnahmeumgebungen. Der Kundensetmanager definiert den Inhalt von Kundensets. Damit kann durch die Überprüfung der Zuordnung der Software bzw. ihrer Änderung zu einem Kunden das falsche Verteilen von Software vermieden werden. Der Kundensetmanager legt den Inhalt von Kundensets fest und ist deshalb im Change-Prozess auch bei der Genehmigung von Einführung neuer Software beteiligt. Der Kundensetmanager kann organisatorisch beim Konfigurationsmanagement angesiedelt sein.

Die Rolle des Qualitätsmanagers gibt es zwar heute schon. Im dezentralen Bereich übernimmt das Release-Management bereits Qualitätsmanagementaufgaben. Im neuen Kontext ist der QM zuständig für das gesamte Qualitätsmanagement in der Integrations- und QS-Phase über alle Plattformen (vgl. Anforderung in Abschnitt 4.4.2). Er legt die Testinhalte und den Testumfang in diesen Phasen fest und kontrolliert ggf. auch deren Durchführung. Weiterhin definiert er Qualitätskennzahlen, mit denen sowohl die Qualität der Produkte als auch die Qualität der Prozesse festgestellt werden kann. In diesem Zusammenhang wird auch ein unabhängiges Testteam eingeführt.

Die Tester des Testteams führen Tests sowohl in der Integrations-, QS- als auch in der Pilotphase durch. Sie können dafür von verschiedenen Stellen beauftragt werden. Der Kunde beauftragt es z.B. für spezielle Lasttests, der Projektleiter für Integrationstests oder der Integrator für Installationstests. Durch das Testteam ist die Anforderung (vgl. Abschnitt 4.4.2) nach einer unabhängigen Qualitätssicherung erfüllt.

Die Rolle des Systemintegrators und Anwendungsintegrators wurde zusammengefasst als Integrator. Dieser koordiniert die plattformspezifischen Integrationsaufgaben. Der UM und der Roll-out-Verantwortliche agieren ebenfalls plattformübergreifend (vgl. Anforderung in Abschnitt 4.4.4.3). Der UM koordiniert in Absprache mit dem RM oder Projektleiter die Bereitstellung der jeweils plattformspezifischen Integrations- oder QS-Umgebung. Für die plattformspezifischen Aufgaben gibt es dafür jeweils den plattformspezifischen Umgebungsmanager. Der Roll-out-Verantwortliche ist der Ansprechpartner des RM in allen Angelegenheiten bzgl.

der Softwareverteilung und -installation in der Produktivumgebung. Er koordiniert hierfür die plattformspezifischen Roll-out-Verantwortlichen bzw. Softwareversorger.

Das RMG besteht aus am Release-Prozess beteiligten Rollen. Es wird nach Bedarf vom RM einberufen. Die jeweilige Zusammensetzung ist dabei individuell und abhängig von der jeweiligen Fragestellung. Es tritt vor allem bei übergreifenden Fragestellungen, z.B. bei der Abstimmung der Ressourcen oder Inhalten der Releasepolicy oder bei Eskalationen zusammen.

5.2 Release-Aktivitäten

Wie in Abschnitt 4.4.1 gefordert, werden zur vollständigen Migration der derzeitigen Release-Prozesse die Release-Aktivitäten an ITIL ausgerichtet. Dadurch entstehen klare Schnittstellen zu den bisher etablierten ITIL-Prozessen wie etwa dem Change- oder Incident-Management. Der RM ist zuständig für die Festlegung der Release-Policy, die Release-Planung und die Steuerung des Release-Prozesses (siehe Abbildung 5.3). Die Integrationsphase wird zukünftig nicht mehr vom RM, sondern vom Integrator gesteuert. Im Release-

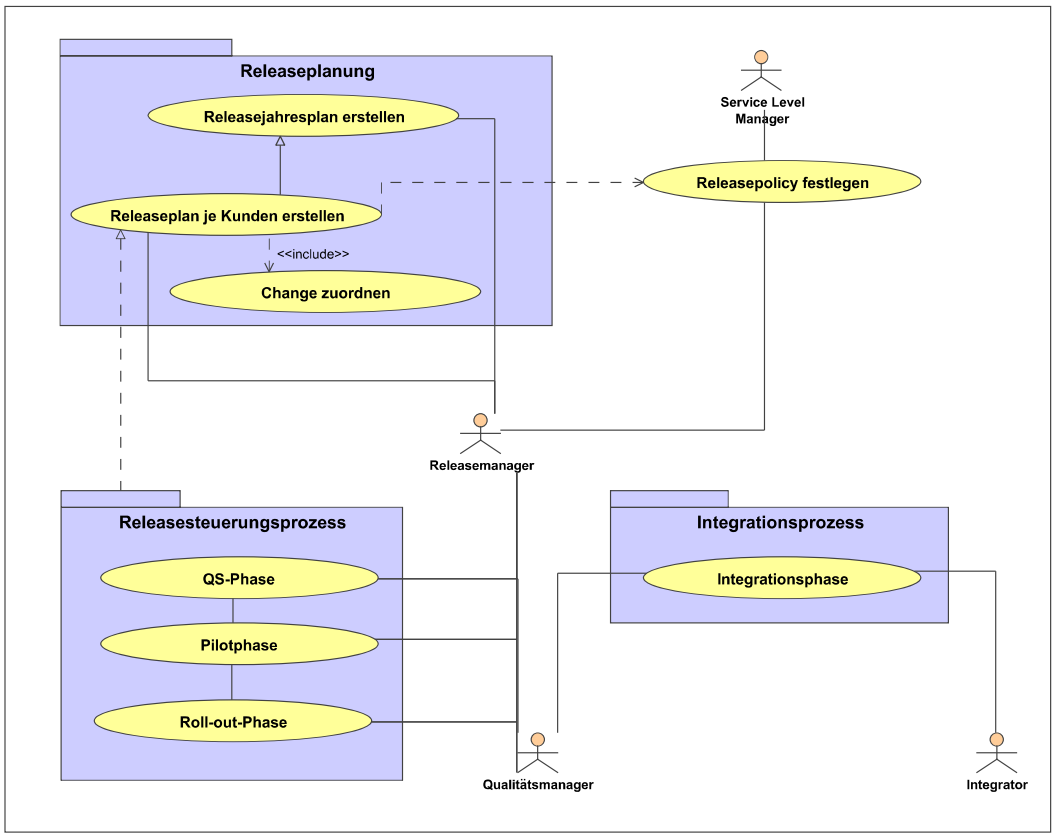


Abbildung 5.3: Übersicht der Release-Aktivitäten

Steuerungsprozess koordiniert und steuert der RM die QS-, Pilot- und Roll-out-Phase. Die in Abbildung 5.3 dargestellten Aufgabengebiete des RM werden nachfolgend detaillierter beschrieben. Die Gliederung dieses Abschnittes richtet sich wieder nach den Hauptgliederungspunkten von ITIL aus Abschnitt 2.3.4.3.

5.2.1 Festlegen der Releasepolicy und des Release-Plans

Das Festlegen der Releasepolicy erfolgt auch weiterhin in Kooperation des Release-Managements mit dem Service Level-Management und dem Kunden. Die dafür nötigen Aktivitäten sind im folgenden Abschnitt dargestellt.

5.2.1.1 Festlegen der Releasepolicy

Die nach ITIL relevanten Inhalte einer Releasepolicy, wie etwa Festlegung der Release-Einheiten oder der Dokumentationsrichtlinien, werden nachfolgend beschrieben.

5.2.1.1.1 Festlegung der Release-Einheiten Aufgrund der Anforderung zur Einführung eines Konfigurationsmanagements inkl. der unterstützenden CMDB (siehe Abschnitt 4.4.1.3.1) werden die Release-Einheiten jetzt individuell definiert.

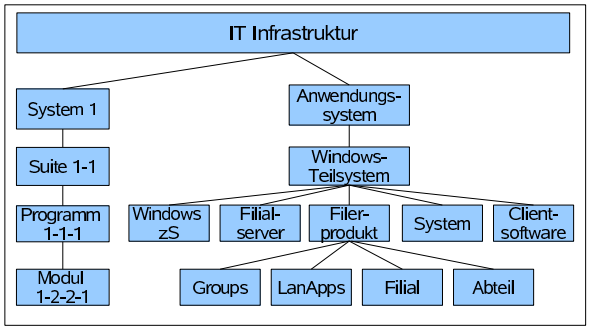


Abbildung 5.4: Beispiel Release-Einheiten im Windows-Bereich

Bei den bisherigen Festlegungen gibt es keine Übereinstimmung zwischen den organisatorisch und technisch gehandhabten Release-Einheiten. Jetzt entsprechen die technisch geänderten und übergebenen Komponententypen denjenigen im organisatorischen Ablauf. Im Change-Antrag bzw. detaillierter in der Beauftragung werden die geänderten Komponenten vom Projektleiter bzw. PV angegeben. Für die dezentrale Plattform sind das z.B. die geänderten Komponenten der Teilsysteme, im Host-Bereich KVS-Packages und im Bereich der zentralen Server, je nach Änderungsart Anwendungssysteme oder deren Komponenten.

Die Eintragungen im bestehenden *Repository* reichen hierfür nicht aus. Deshalb liefert das Konfigurationsmanagement die nötigen releaserelevanten Informationen. Im Windows-Bereich erfolgt dann die weitere Unterteilung der Teilsysteme wie in Abbildung 5.4 dargestellt. Sie zeigt die heutige im Windows-Bereich bei den Verschalern gebräuchliche Einteilung.

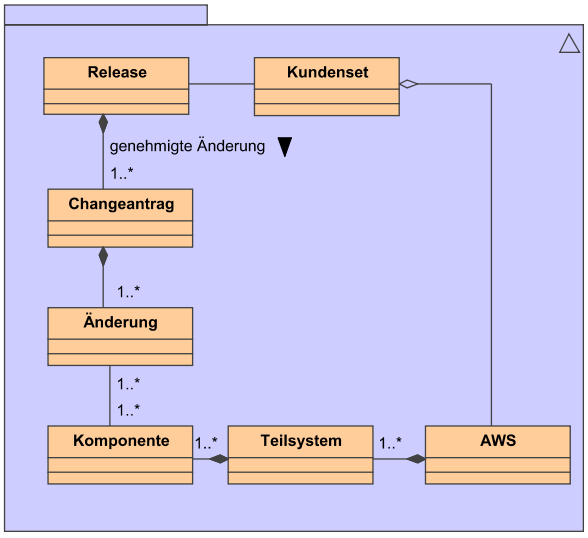


Abbildung 5.5: Zusammenhang zwischen Release, Änderung, Change und Komponenten

Abbildung 5.5 zeigt den neuen Zusammenhang zwischen Release und Änderungsantrag (Change, RfC) bzw. Änderung. Er ergibt sich aus der Anforderung nach der neuen Ausrichtung der Verbindung des Change- und Release-Managements (siehe Abschnitt 4.4.1.2). Ein Release entspricht jetzt der Bündelung aller genehmigten $(1..n)$ Änderungen. Jede Änderung entspricht- wie derzeit im Host-Bereich- einer fachlogisch zusammengehörigen Änderung. Diese wird entweder als Ganzes verteilt und installiert oder kann als Ganzes ohne negative Einflüsse wieder zurückgenommen werden.

Die Änderung bezieht sich auf mindestens ein Teilsystem. Sie kann aber auch die Änderung mehrerer Teilsysteme verschiedener Plattformen beinhalten. Jedes Teilsystem ist wiederum ein Bestandteil eines AWS. Ein Release wird weiterhin kundenspezifisch gehandhabt. Aus diesem Grund bleibt die Aggregation der AWS zu einem Kundenset bestehen.

5.2.1.1.2 Release-Bezeichnung und -Nummerierung Aufgrund der Anforderung zur Einführung von Standards (siehe Abschnitt 4.4.4.4) und der Orientierung an ITIL werden die Begriffe zur Release-Bezeichnung vereinheitlicht. Derzeit beinhalten sie verschiedene Aussagen, wie z.B. Release-Durchlauf oder Release-Inhalt (siehe Abschnitt 3.2.2.3). Durch die Standardisierung wird das Verständnis der am Release-Prozess beteiligten Mitarbeiter untereinander verbessert. Da nicht mehr für jeden Release-Aspekt, wie etwa Release-Durchlauf etc. ein eigener Begriff gefunden werden muss, ergibt sich auch eine Flexibilität. Diese kann auch im zukünftigen, RM-unterstützenden Tool abgebildet werden. Wo heute in Stargate nur zwei

Release-Arten (Standard-Release und Hotfix) als Workflow abgebildet sind, werden zukünftig, unabhängig von der Wahl eines Release-Begriffes und seiner Bedeutung, flexible Abläufe möglich sein. Aus diesem Grund reichen in Zukunft die drei Release-Begriffe **Standard-Release, Sonderversorgung und Hotfix** aus. Die Kunden können natürlich weiterhin eigene Bezeichnungen für ihre Releases festlegen. Die Klassifiktion dieser Release-Arten ergibt sich wie folgt:

Standard-Release: Standard-Releases sind regelmäßig durchgeführte, vom RM vorab geplante Releases, z.B. auf monatlicher Basis. Sie können Änderungen aller Plattformen beinhalten. Im ITIL-Sinne entsprechen sie Major-Releases.

Sonderversorgung: Sonderversorgungen sind vorab planbare Releases, die terminlich nicht über Standard-Releases gehandhabt werden können. Sie können ebenfalls Änderungen aller Plattformen betreffen und sind i.d.R. weniger umfangreich bzw. aufwendig als Standard-Releases. Aus diesem Grund entsprechen sie im ITIL-Sinne den Minor-Releases.

Hotfix: Hotfixes (ITIL Emergencyreleases) sind ebenfalls für alle Plattformen möglich. Sie treten ad-hoc auf und dienen der Behebung schwerwiegender Fehler in der Produktion bzw. werden auf expliziten Kundenwunsch durchgeführt. Der Ablauf von Emergencyreleases kann einen schnelleren bzw. verkürzten Release-Steuerungsprozess haben. Diese Beschleunigung wird durch das Weglassen einzelner Tests bzw. das Auslassen der kompletten Abnahmen in der QS- oder Pilotphase erreicht. Der jeweilige Testumfang bei Emergencyreleases wird vom QM festgelegt.

Die bestehende Release-Bezeichnung und -nummerierung mit der Benennung des betroffenen Kundensets, der Release-Art und einer laufenden Nummer verbunden mit der Jahreszahl, wird beibehalten. Das erste Standard-Release im Jahr 2006 für das Kundenset KVZ heißt somit $\langle Standard - Release - 1 - 2006\rangle - \langle KVZ\rangle$. Sonderversorgungen werden beispielsweise $\langle Sonderversorgung - 1 - 2006\rangle$ -$\langle KVZ\rangle$ genannt und Hotfixes $\langle Hotfix - 1 - 1 - 2006\rangle$ -$\langle KVZ\rangle$, falls es sich um den ersten Hotfix zum ersten Standard-Release des Kundensets KVZ handelt.

5.2.1.1.3 Definition von Major- und Minor-Releases und einer Regel für Emergencyfixes

Standard-Releases sind Major-Releases. Es sollen möglichst alle Änderungen über Standard-Releases in die Produktion eingeführt werden. Bei ihnen kann der gesamte Ablauf rechtzeitig vorab geplant werden und damit auch der Einsatz der benötigten Ressourcen. Aus diesem Grund sind sie in der Durchführung effizienter als etwa ungeplant auftretende Emergencyfixes. Ausnahmen bilden Sonderversorgungen, welche z.B. aufgrund gesetzlicher Vorschriften andere Termine benötigen. Sie können sowohl Major- als auch Minor-Releases sein. Hotfixes entsprechen Emergencyfixes. Im Vergleich zu heute sollen sie nur noch durchgeführt werden, wenn tatsächlich dringende Fehler behoben werden müssen oder auf expliziten Kundenwunsch. Eine zu hohe Anzahl von Hotfixes gefährdet die Termine der vorab festgelegten Standard-Releases und Sonderversorgungen und kann schlimmstenfalls zu Inkompatibilitäten aufgrund von sich selbst überholenden Releases führen (vgl. Abschnitt 3.3).

5.2.1.1.4 Häufigkeit von Major- und Minor-Releases

Standard-Releases als Major-Releases werden anhand der in den SLA spezifizierten Frequenz durchgeführt. Diese wird aufgrund der Anforderungen aus Abschnitt 4.4.4 kundenspezifisch angepasst. Beim Kundenset KVZ kann das z.B. eine monatliche Taktung anstatt der bisherigen fünf Standard-Releases im Jahr bedeuten. Die Anzahl der Sonderversorgungen sollte so gering wie möglich sein, da sie die Termineinhaltung der bestehenden Standard-Releases aufgrund evtl. entstehender Ressourcenengpässe beeinflussen können. Wenn möglich, sollten vom RM die Standard-Release-Termine so verschoben werden, dass sie auch gleichzeitig die Sonderversorgung mit handhaben können.

5.2.1.1.5 Identifikation von geschäftskritischen Zeiten, in denen Implementationen vermieden werden sollen

Geschäftskritische Zeiten werden weiterhin vom Kunden in Form von Frozen Zones bestimmt.

5.2.1.1.6 Erwartete Begleitdokumentation für jeden Release-Typ

Aufgrund der in Abschnitt 4.4.1 geforderten Orientierung des RM an ITIL ist für jede durchzuführende Programmänderung ein genehmigter

Change nötig. Weitere erwartete Begleitdokumente (Installationsanleitung, Benutzerhandbücher etc.) entsprechen *CI*. Sie sollen zukünftig vom Konfigurationsmanager in der CMDB verwaltet werden. Die Einführung des Konfigurationsmanagements wird in dieser Diplomarbeit zwar gefordert, die konkrete Durchführung ist jedoch die Aufgabe weiterführender Arbeiten. Der Umfang der Begleitdokumente wird analog den bisherigen Festlegungen in den PRK vorab zwischen den beteiligten Rollen, d.h. QM, RM, Konfigurationsmanagement, Integration, UM und Kunde festgelegt.

5.2.1.1.7 Richtlinien, wie und wo Releases dokumentiert werden sollen Ein Release als Bündelung von genehmigten Änderungen entspricht selbst wieder einem CI. Die Dokumentation des Releases an sich erfolgt deshalb mit Unterstützung des Konfigurationsmanagements. Nachdem für den eigentlichen Release-Steuerungsprozess eine Werkzeugunterstützung durch ein geeignetes Workflowtool nötig ist (vgl. Anforderung in Abschnitt 4.4.4.2.2), können die entsprechenden Steuer- und Statusinformationen auch innerhalb dieses Werkzeuges dokumentiert werden. Es besteht dann eine definierte Schnittstelle zum Konfigurationsmanagement. Die Spezifikation und Umsetzung eines neuen Werkzeuges zur Unterstützung des RM wird im Rahmen eines neu gestarteten Projektes bei der HVBInfo durchgeführt werden.

5.2.1.1.8 Regel für Erstellen und Testen von Back-out-Plänen Jede genehmigte Änderung muss für sich genommen wieder zurückgenommen werden können. Im Host-Bereich wird das bereits über KVS automatisiert praktiziert. Sowohl für dezentrale als auch zentrale Server Komponenten muss der Back-out-Plan spätestens bis zum Ende der Integrationsphase nachgewiesen werden. Der Integrator ist für die Kontrolle des Nachweises zuständig. In Abschnitt 4.4.4.4 wird die Einführung von Standards, insbesondere auch Deployment-Standards ähnlich dem Vorbild im Host-Bereich, gefordert. Damit können zukünftig auch im Bereich C/S und zS die Back-out-Tätigkeiten standardisiert und somit auch automatisiert werden.

5.2.1.1.9 Zuständigkeiten des RM Die Aufgaben des RM sind in Abbildung 5.3 dargestellt. Das RM definiert zusammen mit dem Service Level-Manager und dem Kunden die Releasepolicy. In der Releasepolicy werden auch Festlegungen bzgl. der Häufigkeit von Standard-Releases definiert. Auf Basis dieser Festlegungen erstellt der RM die jeweiligen Release-Jahrespläne, unterstützt durch das in Abschnitt 4.4.4.2.1 geforderte Werkzeug zur Planungsunterstützung. Durchzuführende Programmänderungen werden zukünftig analog des Verfahrens in ITIL (vgl. Anforderung 4.4.1) vor Start der Entwicklungsaktivitäten vom jeweiligen Projektleiter bei Neuentwicklungen oder PV bei Wartungsvorhaben per Change beantragt. Der RM überprüft diesen Change auf Durchführbarkeit und ordnet den jeweiligen Change-Antrag einem vorab festgelegten Standard-Release zu. Lediglich Sonderversorgungen und Hotfixes bilden hierfür eine Ausnahme. Sie werden individuell je nach Dringlichkeit vom RM gehandhabt und deswegen evtl. außerhalb der Standard-Release-Termine abgewickelt. Das RM ist zuständig für den jeweiligen Release-Steuerungsprozess. Dieser Prozess beinhaltet die Steuerung und Kontrolle der QS-, Pilot- und Roll-out-Phase. Nach jedem erfolgten Roll-out erstellt der RM eine Release-Nachbetrachtung (siehe Abschnitt 5.2.8). Diese Release-Nachbetrachtung entspricht in ITIL dem *Post Implementation Review* (PIR) . Dadurch wird die erfolgreiche Einführung des Releases evaluiert und es werden wie bisher Kennzahlen zum RM erstellt. Mithilfe der Release-Nachbetrachtung können Verbesserungsvorschläge für den KVP erarbeitet werden.

Die bisherige IT-Phase mit dem Schwerpunkt der Integrationsaufgaben liegen nicht mehr im Fokus des Release-Steuerungsprozesses (siehe Anforderung in Abschnitt 4.4.1.1). Alle Integrationsaufgaben müssen bis zu Beginn der QS-Phase abgeschlossen sein. Der so erreichte Qualitätsstand der jeweiligen Komponenten wird durch den Eintrag der Komponente in der DSL dokumentiert. Die konkrete Einführung der DSL ist die Aufgabe weiterführender Projekte.

5.2.1.1.10 Dokumentation der exakten DSL-Konfiguration Eine DSL muss in weiterführenden Projekten eingeführt werden. Die exakte DSL-Konfiguration wird mit dem Konfigurationsmanagement abgestimmt. Aufgrund der technologischen Unterschiede der eingesetzten Komponenten kann die DSL auch nur aus einer Zusammenfassung der benötigten Meta-Informationen bestehen. Die physischen Objekte werden dann in entsprechend festgelegten Speicherorten aufbewahrt. Im Host-Bereich kann das z.B. weiterhin KVS bedeuten.

5.2.1.2 Festlegen des Release-Plans

Der jeweilige Release-Plan ergibt sich aus der in der Releasepolicy definierten Anzahl der Standard-Releases und aus den vorab festgelegten Sonderversorgungsterminen. Anhand dieser Vorgaben wird vom RM ein Jahresplan mit den entsprechenden Terminen für den Start der QS-, Pilot- und Roll-out-Phase definiert. Die Integrationsphase und etwaige Iterationen in dieser Phase werden aufgrund der großen Bandbreite der plattformspezifischen Integrationszeiten nicht mehr in diesen Jahresplan integriert (vgl. Anforderung in Abschnitt 4.4.1.1). Die jeweilige Fertigstellung der Integration bis zum Startpunkt der QS-Phase muss vom jeweiligen Projektleiter bzw. PV selbst sichergestellt sein.

Der RM ordnet diesen fest definierten Terminen die jeweils beantragten Softwareänderungen (Changes) zu. Dieses Verfahren wird anstelle der bisherigen Einmeldung eingeführt. Sollten sich im Entwicklungs- und Integrationsverlauf von Komponenten Verzögerungen ergeben, werden diese einem anderen Release zugeordnet, möglichst aber dem nächsten stattfindenden Standard-Release. Der Release-Durchlauf nimmt somit keine Rücksicht mehr auf Verzögerungen bei der Bereitstellung einzelner Komponenten. Dadurch verkürzt sich dieser Durchlauf im Vergleich zum bisherigen Vorgehen, vor allem im dezentralen Bereich (vgl. Abschnitt 4.1).

Abbildung 5.6 zeigt beispielhaft den Release-Plan für zukünftige Standard-Releases über alle Plattformen. Meilensteine im Release-Plan sind Beginn bzw. Ende der Vorbereitungsaktivitäten der QS-Abnahmen, der QS-Abnahmen, der Pilotphase und der Roll-out-Phase. Die Dauer der Vorbereitung der QS-Phase ist abhän-

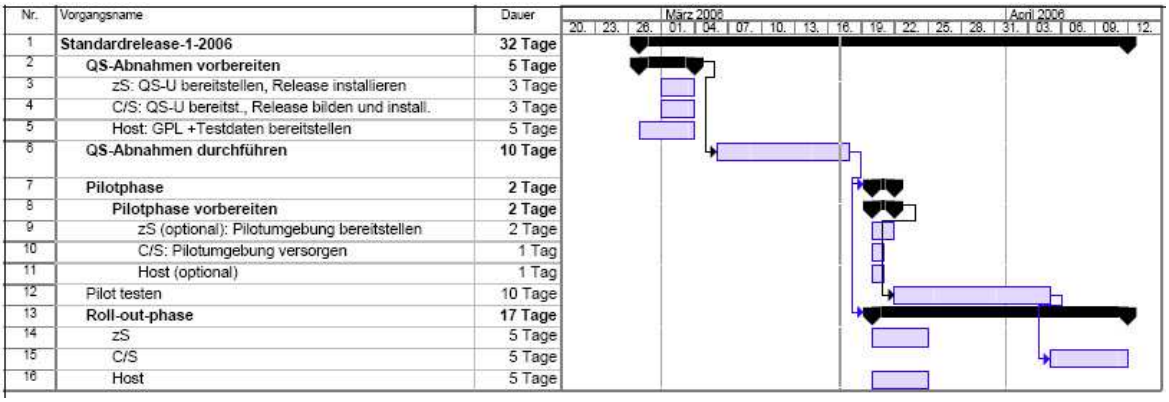


Abbildung 5.6: Beispiel eines plattformübergreifenden Release-Plans

gig von der Art und vom Umfang der genehmigten Changes und der dadurch benötigten Plattformen. Die jeweils für die Abnahmephase notwendigen Plattformen bzw. notwendige Hardware sind jetzt aber durch die Unterstützung des UM durch das Konfigurationsmanagement vorab bekannt und damit planbar. Die Dauer der QS-Abnahmephase kann vom RM in Absprache mit dem QM nach Bedarf variiert werden. Die Durchführung der Pilotphase im Bereich zS und im Host-Bereich ist weiterhin optional und wird individuell festgelegt. Die Abwicklung des eigentlichen Roll-outs kann plattform- und release-spezifisch erfolgen und wird im RMG definiert. Der Plan für die jeweilige Roll-out-Phase wird vom Roll-out-Verantwortlichen detailiert. Er übernimmt die Koordination der plattformspezifischen Roll-out-Tätigkeiten. Diese werden wie bisher vom Softwareversorger und Einsatzverantwortlichen durchgeführt.

5.2.1.2.1 QS-Phase Die QS-Phase ist unterteilt in Vorbereitungsaktivitäten und in die eigentliche Abnahmephase. Die Vorbereitungsaktivitäten beinhalten den Aufbau der jeweils notwendigen QS-U, die Integration und Installation des Releases in die QS-U und die Bereitstellung von Testdaten. In der Regel erfolgt die Datenhaltung hauptsächlich auf dem Host, selten auch bei zentralen Servern. Aus diesem Grund erfolgen in deren Vorbereitungsphase auch die Bereitstellung der entsprechenden Testdaten. Wenn diese Vorbereitungsaktivitäten abgeschlossen sind, beginnt die QS-Abnahmephase. In dieser erfolgen alle Testaktivitäten.

5.2.1.2.2 Pilotphase Die Pilotphase erfolgt im dezentralen Bereich wie bisher in der vom EM festgelegten Pilotfiliale. Im Bereich zS und Host ist diese Phase weiterhin optional und vom jeweiligen Projekt abhängig. Sie wird deshalb nur bei Bedarf eingeplant.

5.2.1.2.3 Roll-out-Phase Der Verlauf des eigentlichen Roll-outs kann für jede Plattform und für jedes Releases individuell erfolgen. Das wird vom RM entsprechend den Anforderungen der Kunden und aufgrund technischer Anforderungen festgelegt. Unterstützt wird er bei dieser Festlegung durch das RMG. Der Roll-out kann z.B. zeitgleich über alle Plattformen erfolgen oder zeitversetzt.

Bei einem zeitversetzten Roll-out, wenn z.B. der Roll-out im Host-Bereich erfolgt, während bei den anderen Plattformen noch Pilottests stattfinden, muss im ungünstigsten Fall beim Auftreten von Fehlern in der Pilotphase auch der Host zurückgesetzt werden. Beim Roll-out in die Fläche im dezentralen Bereich müssen bei der Roll-out-Planung auch die Verteilzeiten berücksichtigt werden. Änderungen im Bereich zS und C/S werden nach der Pilotphase ausgerollt. Der Roll-out-Verantwortliche erstellt hierfür die konkreten Roll-out-Abläufe. Der dargestellte Release-Plan stellt lediglich einen Beispielentwurf dar. Dieser kann sich verkürzen, wenn z.B. Aktivitäten am Wochenende oder in der Nacht stattfinden können. Das kann beispielsweise durch die in Abschnitt 4.4.4.4 geforderte Standardisierung und dem damit verbundenen Automatisierungspotenzial bei z.B. Installationsvorgängen oder automatisierten Tests erreicht werden.

5.2.2 Entwurf, Release-Zusammenstellung und -Konfiguration

Der zuständige Projektleiter bzw. PV entwirft die jeweilige Release-Einheit. Die Zusammenstellung eines konkreten Releases als Bündelung genehmigter, einzelner Release-Einheiten erfolgt vom RM. Er ordnet dafür, wie in ITIL vorgesehen (vgl. Anforderung 4.4.1), die Changes den jeweils vorab geplanten Releases zu. Bei Bedarf wird diese Zuordnung mit den anderen Prozessbeteiligten abgestimmt. Das Release für den konkreten Release-Steuerungsprozess ergibt sich dann aus dieser Change-Zuordnung. Die jeweiligen Release-Einheiten werden dafür aus der *DSL* entnommen und durch den Integrator zu einem Release gebündelt. Im dezentralen Bereich wird hierfür weiterhin das Gesamt-Release durch entsprechende Verschalungsaktivitäten gebildet (vgl. Abschnitt 3.4.2.2). Bei zS und beim Host steht bereits am Ende der Integration die verteilbare Release-Einheit zur Verfügung.

Falls vorher genehmigte Changes aufgrund von Verzögerungen oder Fehlern noch nicht in der DSL sind, werden sie vom Release-Manager aus dem ursprünglich geplanten Release herausgenommen und einem anderen Release zugeordnet. Im Vergleich zum bestehenden Vorgehen entstehen dadurch im Ablauf keine Verzögerungen mehr. Der Integrator und UM koordinieren und steuern die plattformspezifische Konfiguration der Hardware und der Software wie gehabt.

5.2.3 Testen und Abnahme des Releases

Die Koordinierung der Testaktivitäten sowohl in der QS- als auch in der Pilotphase erfolgt wie gehabt durch den RM. Wenn die Testumgebungen für die Abnahmen bereitstehen, lädt der RM die entsprechenden Beteiligten per Mail zu den Abnahmen ein. Der jeweilige Status der Testumgebung wird wie bisher über die ASU-Ampel im Intranet publiziert. Im Vergleich zum bisherigen Vorgehen, wo alle PV mit geänderten und unveränderten Produkten im Kundenset zum Testen eingeladen wurden, können jetzt durch die Unterstützung des RM durch das Konfigurationsmanagement gezielt diejenigen Personen eingeladen werden, bei deren Komponenten sich zum einen Änderungen ergaben und zum anderen Abhängigkeiten zu den geänderten bestehen. Anstelle der bisherigen Testverantwortlichen (PV, Fachbereich etc.) kann alternativ ein Tester des in Abschnitt 4.4.2 geforderten Testteams mit den Tests beauftragt werden (siehe nachfolgenden Abschnitt).

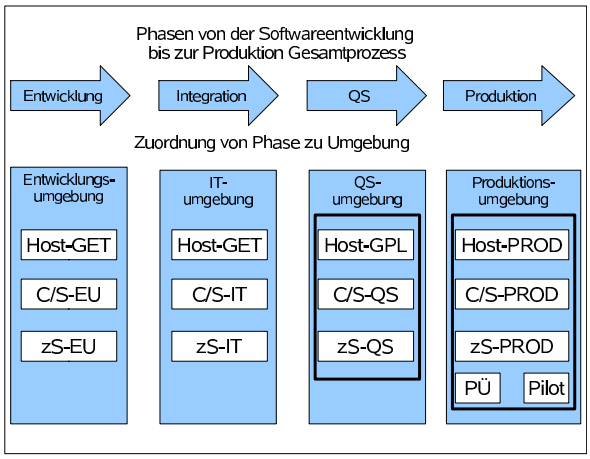


Abbildung 5.7: Übersicht der Prozessphasen und benötigten Umgebungen

Die jeweilige Umgebung für die QS-Phase ist synchronisiert. Abbildung 5.7 zeigt die benötigten Umgebungen in der jeweiligen Phase. Die eingezeichnete Umrandung in der QS-Phase bedeutet eine dem Produktionsumfeld ähnliche und abgeschlossene Umgebung. Das bisherige Vorgehen des Umswitchens von Servern von der IT- zur QS-Phase im Bereich zS kann jetzt durch die neu eingeführte zeitliche Entkoppelung von IT- und QS-Phase zu Engpässen führen. Zur Vermeidung dieses Engpasses sollte auch hier möglichst viel Hardware durch virtuelle Umgebungen ersetzt werden (vgl. Anforderung in Abschnitt 4.4.3.2). Im dezentralen Bereich wird das teilweise bereits praktiziert. Dadurch reduziert sich neben dem finanziellen Aufwand für entsprechende Hardware auch der Platzbedarf.

5.2.3.1 Geschäftsprozesse testen

Durch die synchronisierte Bereitstellung aller Umgebungen inkl. Datenbestände können nun komplette Geschäftsprozesse getestet werden. Der QM definiert zusammen mit dem Kunden die Testfälle für die zu testenden Geschäftsprozesse. Das Testteam wird über den Testauftrag zur Durchführung dieser Geschäftsprozesstests beauftragt. Die Tester des Testteams sind auch für die Dokumentation der Testdurchführung zuständig.

5.2.3.2 Testauftrag

In das bestehende Auftrags-Tool (PRUEV) wird der Testauftrag als neuer Auftragstyp integriert (vgl. Anforderung in Abschnitt 4.4.2). Mit diesem Auftragstyp hat jeder Testverantwortliche die Möglichkeit, eigene Tests durch Tester des Testteams durchführen zu lassen. Der Testauftrag spezifiziert die zur Testdurchführung nötigen Informationen wie die Testfallbeschreibung oder die geschätzte Dauer der Tests.

5.2.4 Planung des Einsatzes

Der Roll-out-Verantwortliche plant den Einsatz des Releases in der Zielumgebung. Hierfür detailliert er die vom RM vorgegebenen Release-Pläne um weitere Informationen wie etwa die konkreten Installationsorte verbunden mit den Installationsterminen. Die detaillierte Roll-out-Planung erfolgt aufgrund der Technologieabhängigkeit des Roll-outs weiterhin plattformspezifisch. Die Veröffentlichung dieser Planung erfolgt wie gehabt im Intranet.

5.2.5 Kommunikation, Vorbereitung und Schulung

Das RM nutzt weiterhin die etablierten Kommunikationsmittel Mail und Intranet zur Bekanntmachung der releaserelevanten Informationen. Ein weiteres Kommunikationsmedium bildet das in Abschnitt 4.4.4.2 geforderte neue Werkzeug zur Workflow-Unterstützung (anstelle von Stargate und RM-Tool-Host). Die Spezifikation dieses Tool liegt nicht im Rahmen dieser Diplomarbeit.

Das EM ist weiterhin für Kommunikations-, Vorbereitungs- und Schulungaufgaben auf Endbenutzerseite zuständig. Aus diesem Grund ist auch das EM bereits bei der Beantragung des Changes involviert. Durch diese Beteiligung an der Change-Genehmigung kann das EM schon vorab den nötigen Vorbereitungs- und Schulungsaufwand abschätzen und abstimmen.

5.2.6 Release-Verteilung und Installation

Die entsprechenden logistischen Prozesse zur Verteilung und Installation des Releases werden vom Roll-out-Verantwortlichen gesteuert. Hier werden auch weiterhin aufgrund der Technologieabhängigkeit die jeweiligen plattformspezifischen Prozesse (siehe Abschnitt 3.4.2.6, 3.5.2.6 und 3.6.2.6) durchgeführt.

5.2.7 Beschreibung des Release-Steuerungsprozesses

Abbildung 5.8 zeigt die wichtigsten Meilensteine des Release-Steuerungsprozesses mit den jeweiligen Subaktivitäten. Auslöser des jeweiligen Release-Steuerungsprozess ist der Termin aus der jeweiligen Release-Planung. Voraussetzung für Programmänderungen sind genehmigte Changes, welche vom RM dem jeweiligen Release zugeordnet sind. Jede geänderte Softwarekomponente muss vor Beginn der QS-Phase den Integrationsprozess erfolgreich durchlaufen haben. Nachfolgend werden die Subaktivitäten Change-Zuordnung, Integrations-, QS-,Pilot- und Roll-out-Phase detaillierter beschrieben.

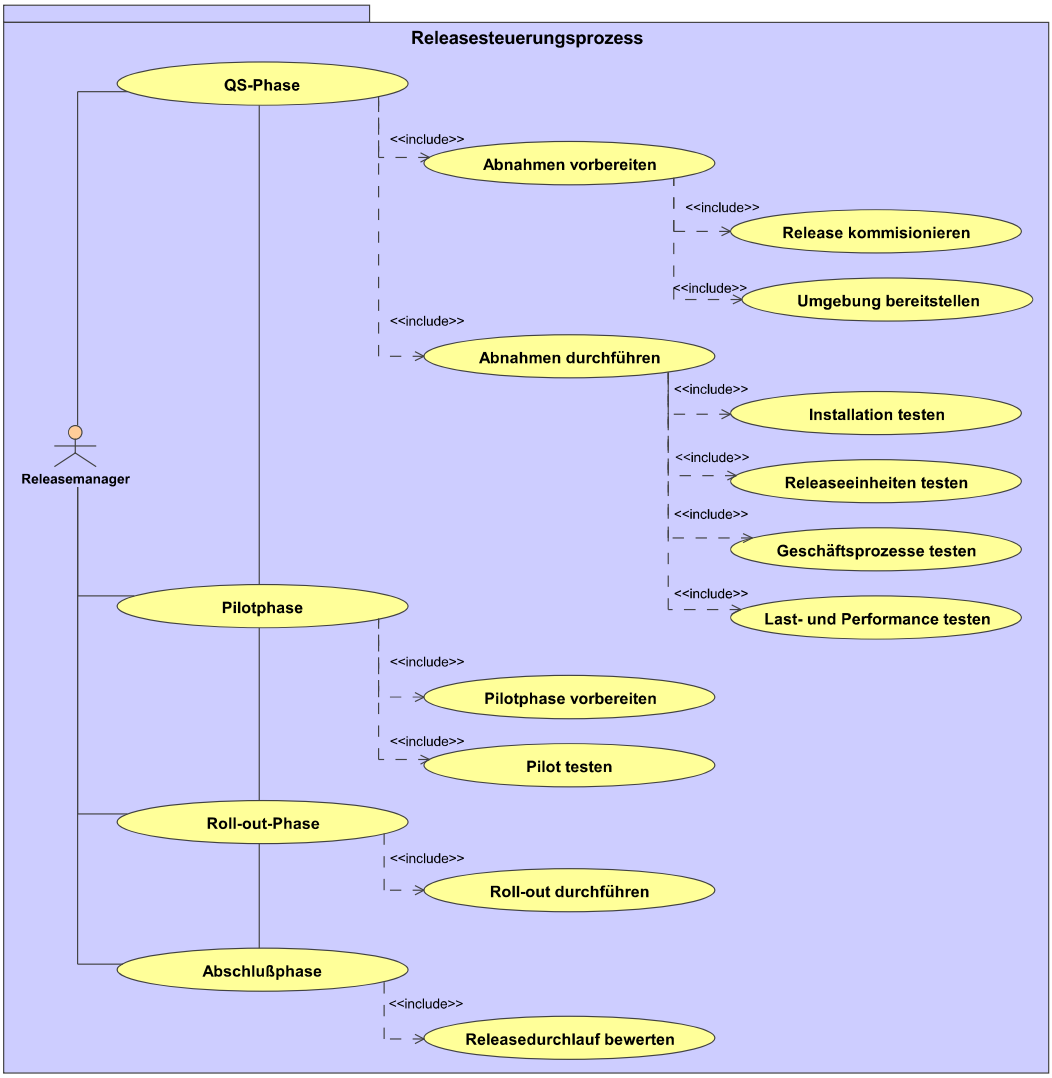


Abbildung 5.8: Darstellung der Phasen im Release-Steuerungsprozess

Beim konkreten Ablauf eines Release-Steuerungsprozesses können Probleme auftreten. Zur Lösung von Problemen ist sowohl eine funktionale Eskalation als auch eine hierarchische Eskalation durch den Release-Manager möglich. Bei der funktionalen Eskalation werden weitere Spezialisten, bei der hierarchischen Es-

kalationen die Vorgesetzten der am Prozess beteiligten Mitarbeiter einbezogen. Bei Bedarf kann der Release-Manager auch das RMG einberufen. Es setzt sich je nach Bedarf aus weiteren Spezialisten oder vorgesetzten Personen zusammen. Eskalationen können überall im Release-Steuerungsprozess auftreten. Auf eine konkrete Darstellung im nachfolgenden Ablauf wird deshalb verzichtet.

5.2.7.1 Change-Zuordnung

Anstelle der bisherigen Einmeldung eröffnet der Projektleiter bzw. PV für jede Softwareänderung einen Change. Die Eröffnung des Changes erfolgt jetzt vor dem eigentlichen Start der Entwicklungsaktivtäten und nicht wie bisher gegen Ende der Softwareentwicklung. Der jeweilige Auftraggeber bzw. Kunde ist für die inhaltliche Genehmigung des Changes verantwortlich. Abbildung 5.9 verdeutlicht den daraus resultierenden

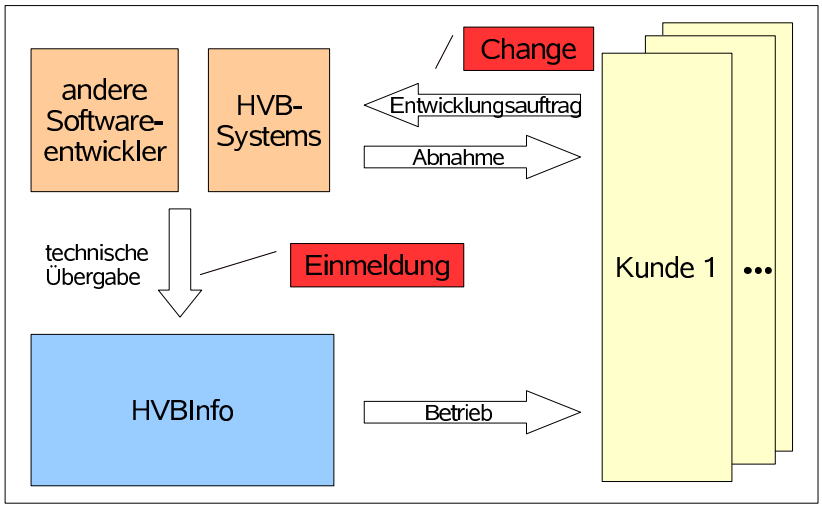


Abbildung 5.9: Schnittstellen der HVBInfo in Verbindung mit Einmeldung bzw. Change

Vorteil. Die bisherige Einmeldung erfolgte an der Schnittstelle zwischen Softwareentwicklung und der technischen Übergabe für den Betrieb bei der HVBInfo und damit relativ spät im Prozess. Im Vergleich dazu resultiert aus dem geforderten ITIL-orientierten Vorgehen durch die Verknüpfung mit dem Change-Prozess, dass die Informationen für das RM früher zur Verfügung stehen. Dadurch verbessert sich die Ressourcenplanung.

Der Change beinhaltet aufgrund der Anforderung in Abschnitt 4.4.1.2 eine Klassifikation der Änderung in korrigierend, adaptiv oder perfektibel. Im Change erfolgt auch die Angabe des erwarteten Fertigstellungstermins. Dieser ist die Grundlage für die terminliche Genehmigung des Changes durch das RM und EM. Im Genehmigungsverfahren können je nach Art der Programmänderung die Integratoren, UV oder Roll-out-Verantwortlichen entweder als weitere Genehmiger bzw. rein informativ beteiligt werden.

Durch diese Einbeziehung des RM und EM können sie ihre Planung frühzeitig abstimmen. Die Integratoren und UV erhalten ebenfalls Kenntnis der durchzuführenden Änderungen. Mit der Unterstützung des Konfigurationsmanagements können sie nun die jeweils zur Softwareänderung benötigte Umgebung identifizieren.

Durch die geforderte Definition des Inhaltes des Kundensets durch den Kundensetmanager kann auch direkt durch den Change-Prozess die Berechtigung des Änderungsantrages evaluiert werden. Das Lizenzmanagement erhält hierbei die nötigen Informationen bei Neueinführungen.

5.2.7.2 Integrationsphase

Die bisherige Phase der Integration, in der sowohl eine Software-Software-Integration als auch die Integration der Software in die entsprechende Hardware erfolgte, ist nun von der QS-Phase zeitlich entkoppelt. Durch diese in Abschnitt 4.4.1.1 geforderte Entkopplung der Phase IT und QS blockieren Iterationen in dieser Phase nicht mehr die geplante Release-Auslieferung.

Neben den Integratoren, die Integrationsaufgaben durchführen und UV, welche nach Auftrag der jeweiligen Projektleiter bzw. PV die Integrationsumgebung bereitstellen, überwacht das Qualitätsmanagement die Integrationsphase. Das QM ist für die Erstellung von Kennzahlen, wie sie heute im dezentralen Bereich bereits ermittelt werden, zuständig. Wichtige Qualitätskennzahlen sind hier die Anzahl der Iterationen, Fehlerfälle und Fehlerart etc.

Mithilfe dieser Kennzahlen können im Rahmen des KVP Verbesserungsmaßnahmen eingeleitet werden. Falls z.B. sehr häufig Integrationsfehler auftreten, können Maßnahmen zur Vermeidung von Iterationen entwickelt werden. Laut Crosby [Cros 96] entspricht jede Art der Iteration einer Nacharbeit und das Ziel soll hier das Reduzieren von Nacharbeit durch geeignete Verbesserungsmaßnahmen sein. Eine Maßnahme zur Verbesserung der Qualität in der Integrationsphase wird bereits heute durch das Projekt ReQuest durch die Bereitstellung von virtuellen Integrationsumgebungen für die Entwicklerarbeitsplätze geplant. Dadurch sollen Integrationsaufgaben schon in der Entwicklung durchgeführt und getestet werden können und damit auch Integrationsfehler eher erkannt werden. In der Integrationsphase finden wie bisher auch weiterhin Verschalungsaufgaben statt.

5.2.7.2.1 Verschalung Die Integratoren verschalen die jeweiligen Softwarekomponenten. Die Aufgaben entsprechen auch hier den plattformspezifischen bereits beschriebenen Tätigkeiten. Durch bereits geplante und teilweise durchgeführte Standardisierungsmaßnahmen, wie etwa Festlegung von *MSI* als Standard im Windows-Bereich (siehe Abschnitt 3.4.2.2), können auch hier Qualitätsverbesserungen erzielt werden.

Die heute auftretenden Back-log-Arbeiten bzw. Auslastungsspitzen werden durch die Einbeziehung der Integratoren in den Change-Prozess vermieden. Die Integratoren haben so die Möglichkeit, vor Entwicklungsbeginn Installationsstandards festzulegen und die Entwickler ggf. bei der Umsetzung zu unterstützen.

5.2.7.2.2 DSL Die verlangte zeitliche Entzerrung von IT- und QS-Phase wird durch die Einführung der DSL unterstützt. Das durch das Projekt ReQuest angedachte so genannte *Hochregallager* für den dezentralen Bereich entspricht im Wesentlichen einer DSL. Nach erfolgreicher Integration wird jede Komponente in der DSL gelagert. In der nachfolgenden QS-Phase wird das jeweilige Release aus Komponenten der DSL zusammengestellt. Die DSL muss hierfür aufgrund technologiebedingter Unterschiede nicht einem einzigen Speicherort entsprechen. Sie kann auch verteilt sein. Der jeweilige Status jeder Komponente wird in der DSL gespeichert und es muss sichergestellt werden, dass die nachfolgenden Verteilaktivitäten nur Komponenten verteilen, welche den Status „für Installation freigegeben “haben.

5.2.7.3 QS-Phase

Der eigentliche Release-Steuerprozess beginnt in der QS-Phase. Der Startzeitpunkt ergibt sich aus dem jeweiligen Release-Plan. Abbildung 5.10 zeigt die Übersicht der Aktivitäten vor dem Start der Abnahmen.

Der RM muss durch die Beauftragung des UM sicherstellen, dass die benötigte Umgebung bereitgestellt wird. Der UM erhält die entsprechenden Umgebungsspezifika aus der Summe der Changes zusammen mit den Umgebungsspezifikationen aus der CMDB. Er baut daraufhin die Umgebung auf und stellt die entsprechenden Testdaten bereit. Die Datenhaltung erfolgt hauptsächlich auf dem Host bzw. zS-Umgebungen. Nach dem Aufbau und der Testdateneinspielung überprüft er die Funktionsfähigkeit der Umgebung anhand vorher definierter Tests. Der RM veröffentlicht diesen Status wie gehabt über die ASU-Ampel.

Die Integratoren werden rechtzeitig mit der Erstellung der Versorgungspakete beauftragt. Auch sie haben die nötigen Informationen bereits aus dem Change und den CMDB-Einträgen erhalten. Die jeweiligen Komponenten entnehmen sie aus der DSL. Mit diesen erstellen sie wie gehabt die plattformspezifischen Versorgungspakete (siehe Abschnitt 3.4.2.2 und 3.6.2.2). Diese werden in die entsprechend vom UM bereitgestellte Umgebung integriert und gleichzeitig wird die Installation getestet.

Nach der erfolgten Installation des Releases auf der entsprechenden Hardware beginnt die QS-Abnahme. Abbildung 5.11 zeigt die wichtigen Tätigkeiten in diesem Abschnitt. Der RM lädt die entsprechenden Testverantwortlichen per Mail zu den Abnahmen ein. Diese führen ihre jeweiligen Tests durch und dokumentieren das Testergebnis. Der Fachbereich nimmt die jeweilige fachliche Änderung ab und dokumentiert dadurch die Freigabe für die Produktion. Die Tester des Testteams führen beantragte (Geschäftsprozess-)Tests durch. Der RM

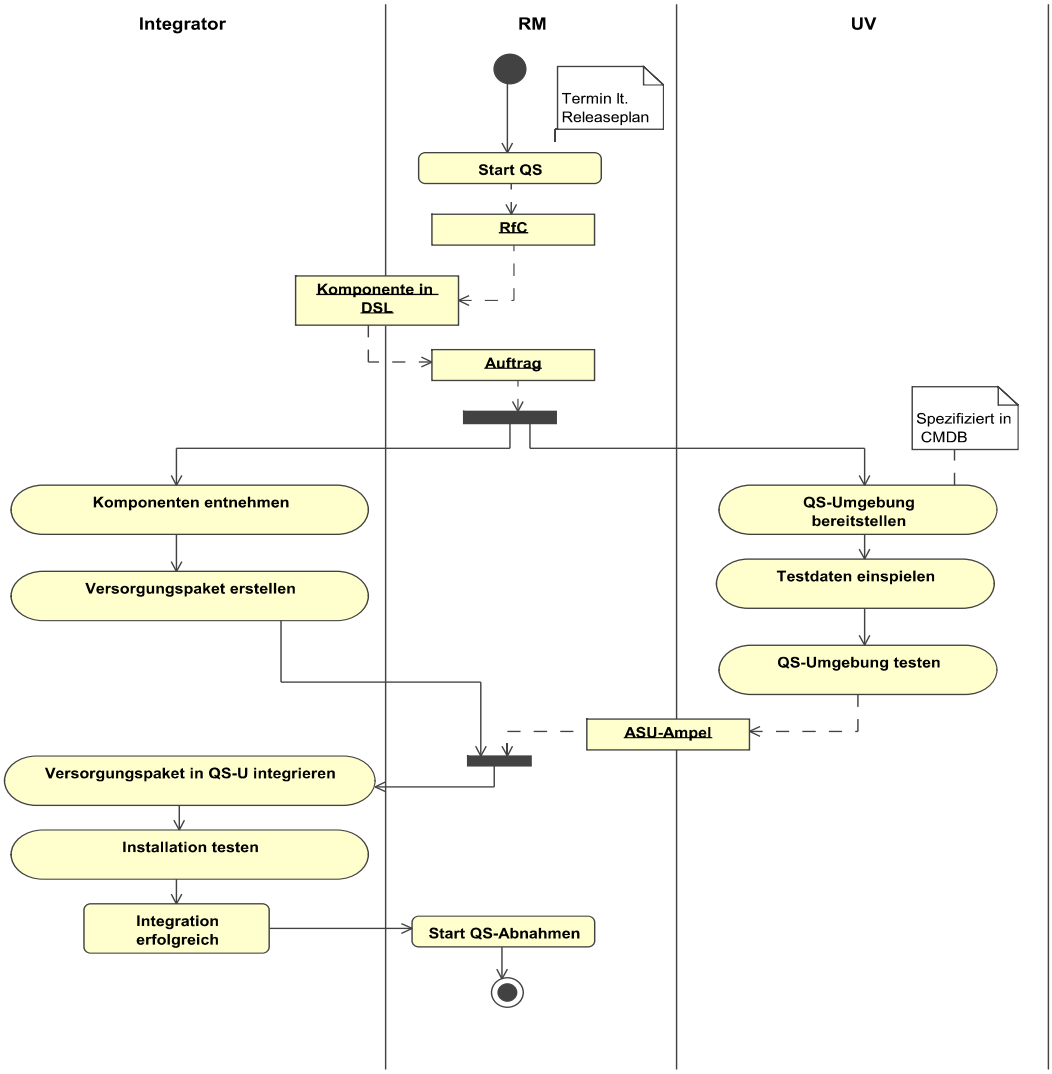


Abbildung 5.10: Aktivitäten in der QS-Phase bis zum Start der Abnahme

verfolgt während der Abnahmen den Verlauf. Er überprüft den jeweiligen Teststatus und klärt bei gescheiterten Tests zusammen mit den Verantwortlichen das weitere Vorgehen. Nach dem Abschluss der Testaktivitäten gibt der RM das Release für die nachfolgende Pilot- bzw. Roll-out-Phase frei. Im Falle des Scheiterns einzelner Tests entscheidet der RM zusammen mit dem RMG das weitere Vorgehen. Handlungsalternativen bestehen im Entfernen der fehlerhaften Komponenten aus dem Release, dem Beibehalten und Verteilen der fehlerhaften Komponente mit einem *Known Error*) oder dem kompletten Stop des Releases mit entsprechender Eskalation.

5.2.7.4 Pilot

Abbildung 5.12 zeigt die Aktivitäten des RM, EM und Roll-out-Verantwortlichen in der Pilotphase. Der RM stimmt den Umfang und den Ablauf der Pilotphase mit dem EM ab. Dieser bestimmt die jeweiligen Pilotarbeitsplätze. Auf diesen wird das Release durch die Roll-out-Verantwortlichen installiert. Der EM koordiniert die jeweilis durchzuführenden Schulungs- und Einführungsmaßnahmen. Die Endbenutzer überprüfen die Funktionalität der neuen Softwareauslieferung. Falls das Release vom Kunden in der Pilotphase freigegeben wird, erfolgt nach dem Ende der Pilotphase der Roll-out.

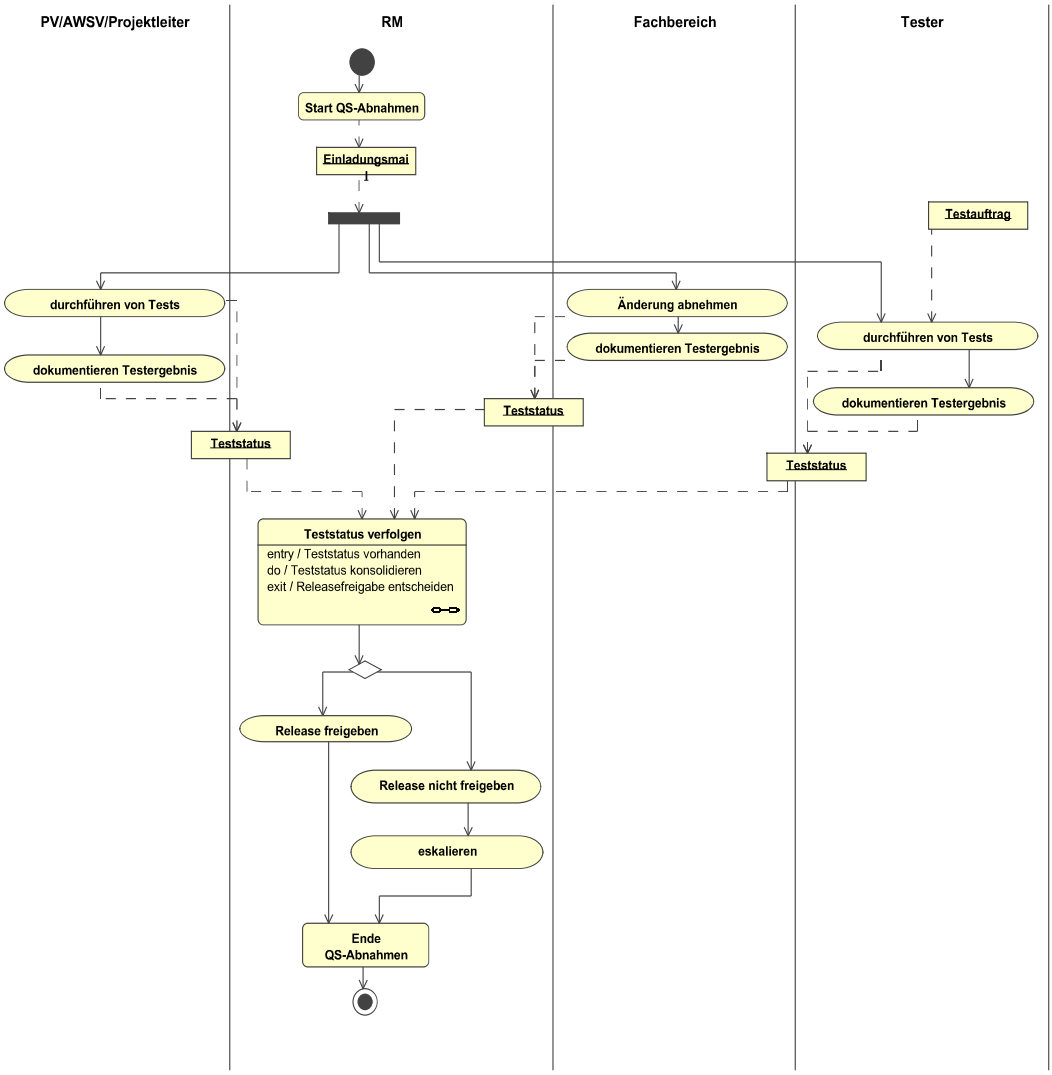


Abbildung 5.11: Aktivitäten in der QS-Phase während der Abnahmen

5.2.7.5 Roll-out

Beim eigentlichen Roll-out erfolgen weiterhin die plattformspezifischen Roll-out-Tätigkeiten. Die jeweiligen Release-Empfänger bzw. Zielumgebungen sind in der geforderten CMDB hinterlegt. Die CMDB enthält hierfür Meta-Informationen aus den bereits bestehenden Verteil- bzw. Installationsdatenbanken wie z.B. der HyperDB im dezentralen Bereich (siehe Abschnitt 3.4.2.6). Durch diese Unterstützung können in Zukunft Fehler wie z.B. das Installieren von für Einzelplätze freigegebenen Softwarekomponenten in der Fläche vermieden werden.

5.2.8 Bewertung des Release-Durchlaufes

Nach Abschluss der Roll-out-Phase eines Release-Durchlaufes wird der Release-Durchlauf analysiert und bewertet. Die Notwendigkeit der Analyse und Bewertung des Release-Durchlaufes ergibt sich aus der Anforderung zur Orientierung des RM an ITIL aus Abschnitt 4.4.1. In ITIL wird diese Nachbetrachtung *Post Implementaion Review (PIR)* genannt. Hierfür erstellt der RM analog im dezentralen Bereich Kennzahlen zu quantitativen und qualitativen Aspekten. Input für diese Auswertung liefern Daten, die im Release-Steuerungsprozess

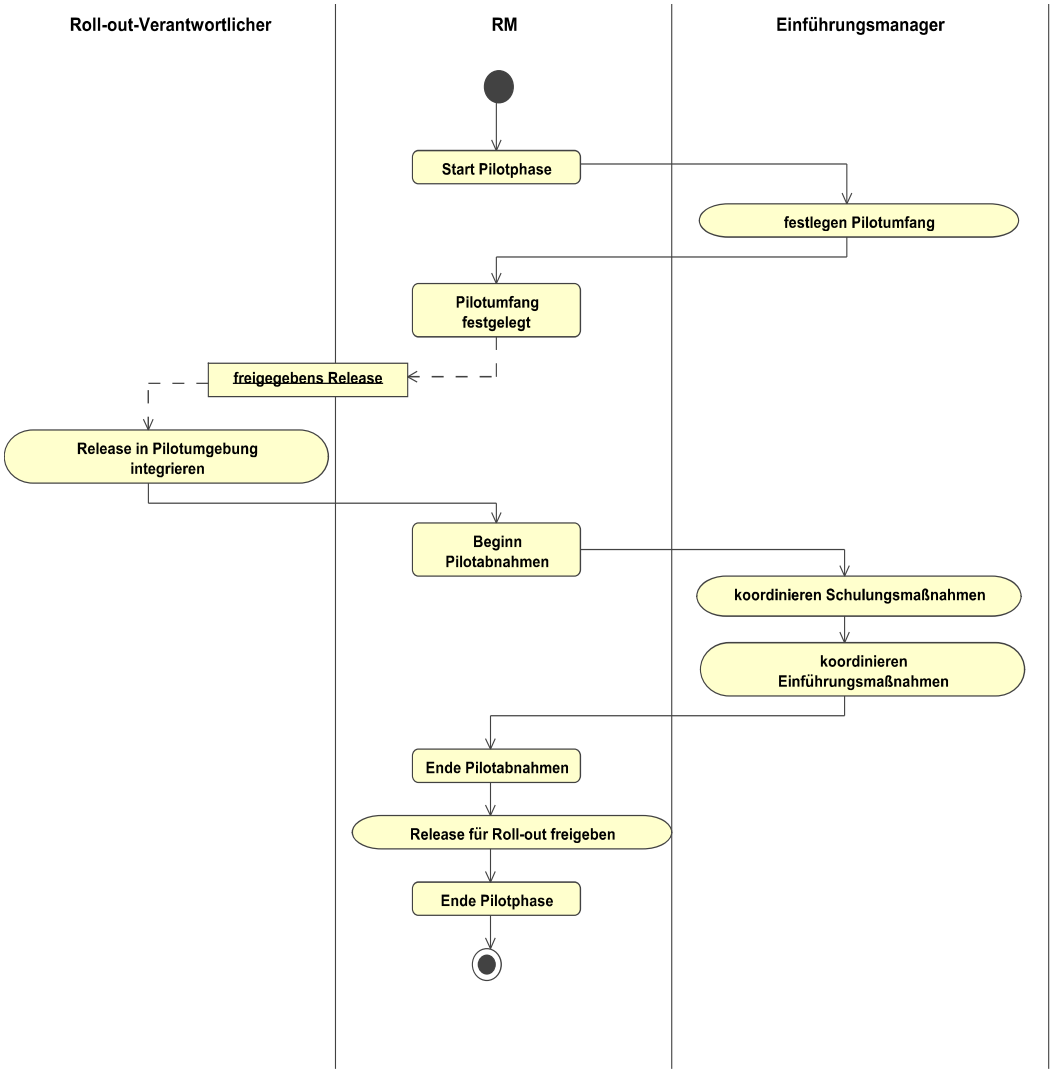


Abbildung 5.12: Aktivitäten in der Pilotphase

gesammelt wurden. Relevante Kennzahlen für diese Auswertung sind (vgl. ITIL [OGC04, S.229f] oder auch CobIT [Cob01]):

- Termintreue: kann anhand der festgelegten Terminen der Meilensteine des Release-Planes evaluiert werden
- Fehleranzahl:
  - Anzahl, Art (korrigierend, adaptiv oder perfektibel) und Gewichtung (minor, major) von Fehlern in der QS-Abnahme- und Pilotphase
  - Anzahl Fehler bei Flächen-Roll-out
  - Anzahl abgelehnter Änderungen in QS-Abnahme oder Pilotphase
  - Anzahl *Incidents* nach Roll-out
- Anzahl der durch das jeweilige Release neuen, geänderten oder gelöschten Komponenten
- Anzahl und Häufigkeit der Änderungen je Plattform und je Komponente
- Anzahl Emergencyreleases

- Umfrageergebnis zu Release-Prozess, Schulungs- und Einführungsmaßnahmen
- Anzahl der *Training Calls* bei der Serviceline
- Anzahl der Komponenten, die automatisch installierbar sind
- Anzahl der Abweichungen von der Standardkonfiguration
- Anzahl genehmigter versus abgelehnter Change-Anträge

Diese Kennzahlen dienen als Grundlage zur Einleitung von Verbesserungsmaßnahmen innerhalb des KVP. Sie können jederzeit durch andere Kennzahlen erweitert werden. Durch die Ermittlung und Auswertung dieser Kennzahlen kann zum einen die Qualität der Produkte und zum anderen die Qualität des Prozesses bewertet werden. Beispielsweise kann eine hohe Änderungshäufigkeit eines Anwendungssystems z.B. durch einen ungenügend geschulten Softwareentwickler, aus einer veralteten Anwendung, die ständig an neue Anforderungen angepasst werden muss oder aus Fehlern in der Systemanalyse entstehen. Erst durch Erfassen dieser Fehlerzahl können Verbesserungsmaßnahmen wie etwa Schulung der Softwareentwickler, Überarbeitung des Systems oder anderer Methoden bei der Systemanalyse eingeleitet werden.

5.3 Einordnung in das Lebenszyklusmodell

Abbildung 5.3 zeigt die Einordnung des neuen, plattformübergreifenden Prozesses in das Lebenszyklusmodell. Die Phase Integration ist durch die zeitliche Entkopplung der bisherigen IT-Phase von der QS-Phase nicht mehr komplett relevant. Integrationsaufgaben fallen im neuen Release-Verfahren lediglich im Rahmen der Vorbereitung der QS-Phase an. Hier werden im dezentralen Bereich die Softwarekomponenten aus der DSL zu einem einzigen Release-Bündel integriert. Des Weiteren findet in der QS- bzw. Pilotphase eine SW/HW-Integration statt, indem die geänderte Software in der QS-Phase in die produktionsnahe Hardware bzw. in der Pilotphase in die produktive Hardware integriert wird.

Der Fokus der QS- und Pilotphase liegt auf dem Akzeptanztest durch die Endanwender und auf operativen Tests, wie etwa Last- und Performanztests. Der Release-Steuerungsprozess ist Bestandteil des neuen, integrierten Prozesses. Durch die Beteiligung an der Genehmigung von Softwareänderungen durch Changes ist das RM zukünftig in der Lage, Änderungen proaktiv zu steuern anstatt wie bisher reaktiv abzuwickeln.

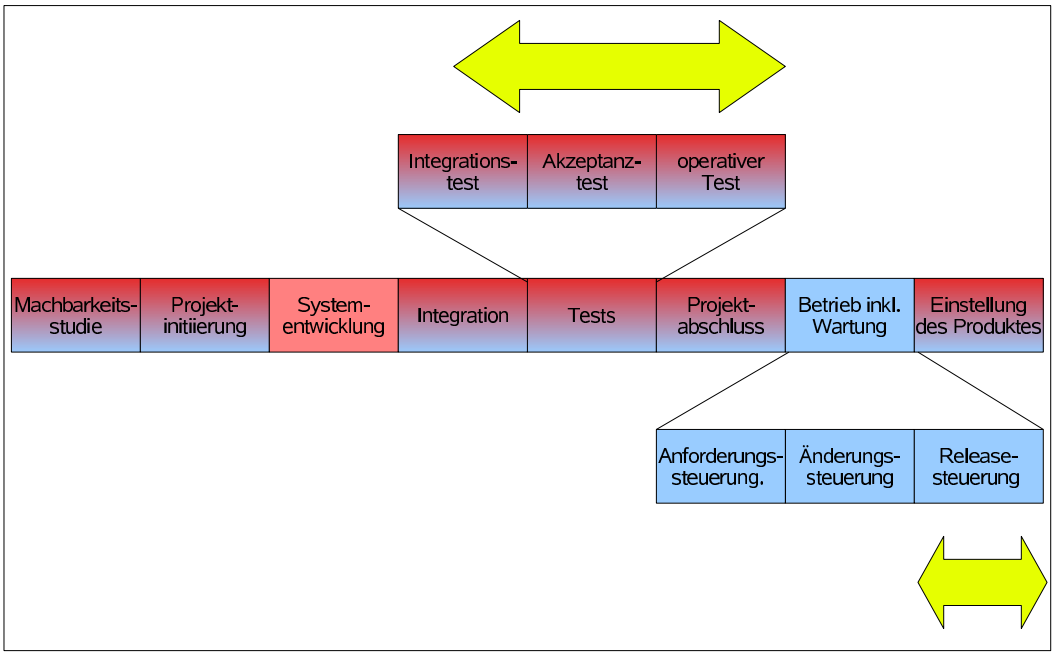


Abbildung 5.13: Einordnung des neuen RM in das Lebenszyklusmodell

5.4 Vorteile durch das integrierte Release-Management

Im Vergleich zum bisherigen Vorgehen ergeben sich durch die Integration der plattformspezifischen Prozesse zu einem einzigen plattformübergreifenden Prozess Vorteile durch Qualitätsverbesserung und die Steigerung der Effizienz. Die Qualität der von der Softwareentwicklung übernommenen Komponenten kann durch ein einheitliches Vorgehen im Abnahmeprozess verbessert werden. Durch die vom QM vorgeschriebenen Tests in der QS-Phase werden alle geänderten Komponenten standardisierten Tests unterworfen. Von Änderungen abhängige Komponenten können durch die Unterstützung des Konfigurationsmanagements einfach identifiziert werden. Dadurch kann die Auswirkung von Änderungen auf andere Einheiten vorab bestimmt und der Testumfang entsprechend festgelegt werden.

Die plattformübergreifende Synchronisation in einer produktionsnahen Umgebung ermöglicht das Testen kompletter Geschäftsprozesse. Das war bisher nicht möglich. Dadurch können Geschäftsprozesse anstatt wie bisher in der Pilotphase bereits in der QS-Phase getestet werden. Die Qualität der Testdokumentation wird durch eine zentrale, durch das QM gesteuerte Dokumentation verbessert. Bisher erfolgt die Testdokumentation vor allem im Bereich zS und Host dezentral. Eine zentrale Kontrolle durch das QM vereinfacht auch die Analyse für mögliche Verbesserungen.

Die bisherigen Tests in der IT- bzw. QS-Phase wurden durch den jeweiligen PV, zum Teil auch durch den Entwickler selbst durchgeführt. Durch die Einführung unabhängiger Tester ergibt sich ein Vier-Augen-Prinzip. Der Entwickler ist dann nicht mehr gleichzeitig der Tester. Deshalb können die Testaktivitäten auch früher starten (vgl. W-Modell [Spil 03]). Durch das unabhängige Testteam ergeben sich u.U. auch finanzielle Vorteile durch Personalkosteneinsparungen und durch die Senkung des vorgeschriebenen Eigenmittelausstattungsbedarfes. Der Eigenmittelausstattungsbedarf wird von der Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) anhand bestimmter Kriterien festgelegt [BaFin]. Durch ein unabhängiges Testteam reduziert sich das so genannte *operational Risk*, was evtl. zu einer Senkung dieser vorgeschriebenen Rücklagen führt.

In der produktionsnahen Abnahmeumgebung können in Zukunft schon vorab ausgedehnte Abnahmetests durch die Kunden erfolgen. Diese Abnahmeumgebung kann ggf. auch der Softwareentwicklung zur Vorführung von Softwareprototypen zur Verfügung gestellt werden. Dadurch haben sie die Möglichkeit, Anforderungen des Kunden an neue Software zu validieren. Durch den Einsatz virtualisierter Umgebungen ergeben sich neben Kosteneinsparungen auch Einsparungen durch einen reduzierten Platzaufwand und geringere Lizenzkosten für Betriebssystemsoftware.

Durch den einheitlichen, technologieunabhängigen Prozess vereinfacht sich in Zukunft die Planung. Bisher aufwändige Abstimmungen, wie etwa zur Reservierung von zentralen Servern für Abnahmen, erfolgen nun zentral koordiniert über das RM. Eine geforderte Tool-Unterstützung des Umgebungsmanagements steigert auch hier die Effizienz. Die Produktionsübergabe von der Softwareentwicklung ist standardisiert für Produkte aller Plattformen. Durch die Standardisierung kann evtl. auch Optimierungspotenzial durch weitere Automatisierung, z.B. bei den Deployment-Aktivitäten im Bereich zS, entstehen.

Der Prozess ist in Zukunft nicht mehr an der Technologie ausgerichtet. Ein einziger einheitlicher Prozess reduziert die Komplexität und Verflechtungen der Aufgaben durch eindeutige Schnittstellen und Standards. Anstatt wie bisher abzuwarten, was im Rahmen einer Einmeldung über die Theke geschoben wird, plant das RM in Zukunft proaktiv im Rahmen des neuen Release-Steuerungsprozesses die Einführung von Softwareänderungen in die produktive IT-Infrastruktur seiner Kunden. Dadurch verbessert sich zukünftig die Auslastung der für den Release-Prozess benötigten Ressourcen.

5.5 Zusammenfassung

In diesem Kapitel wurde das entwickelte Konzept zur Integration des Release-Managements bei der HVBInfo dargestellt. Die Grundlage für dieses Konzept bildeten die in Abschnitt 4.4 erarbeiteten Anforderungen an das integrierte Release-Management. Das Konzept umfasst die Beschreibung der am Release-Prozess beteiligten Rollen und ihre Aufgaben und die Beschreibung des Release-Steuerungsprozesses. Das neue plattformübergreifende Release-Management wurde am ITIL-Release-Management orientiert. Diese neue Ausrichtung verändert vor allem den Release-Steuerungsprozess. Zukünftig wird für jede durchzuführende Softwareänderung

vor Beginn der Entwicklungstätigkeiten ein Change benötigt. In diesem Change wird der Fertigstellungstermin der Softwareentwicklung festgelegt. Die Release-Planung ist im integrierten RM sowohl zeitplan- als auch änderungsgetrieben. Die zeitplangetriebene Release-Planung umfasst wie bisher die vorherige Reservierung der Ressourcen (Testumgebungen, technische Mitarbeiter). Die Taktung dieser Zeitpläne erfolgt kundenspezifisch. Die änderungsgetriebene Release-Planung befasst sich mit der Zuordnung der genehmigten Changes zu diesen reservierten Zeiten. Diese können bei Bedarf flexibel verändert werden. Durch dieses Vorgehen kann die Ressourcenplanung effizient erfolgen.

Die aufgrund der erfolgten Analyse geforderte ITIL-Orientierung macht auch die Einführung eines Konfigurationsmanagements nötig. Es unterstützt dann sowohl das Change- als auch das Release-Management. Auswirkungen durch Änderungen können so zukünftig durch erkennbare Abhängigkeiten vorab bewertet werden. Der Umfang von Testaktivitäten bei Release-Abnahmen kann ebenfalls vorzeitig festgelegt werden.

Aufgrund der erarbeiteten Anforderungen in Abschnitt 4.4 ist die Integrationsphase kein Bestandteil des Release-Prozesses mehr. Sie wird zeitlich von der QS-Phase entkoppelt und vom Integrator gesteuert und vom Qualitätsmanagement überwacht. Der Release-Durchlauf verkürzt sich dadurch wesentlich, da heutige Iterationen in der Integrationsphase nicht mehr berücksichtigt werden müssen. Die Software wird nach der Integrationsphase bis zum Beginn des Release-Durchlaufs in der DSL gespeichert. Von dort wird die Software für ein Release-Bündel zusammengestellt, in der QS-Phase getestet und abgenommen und anschließend ausgerollt.

Das in dieser Arbeit entwickelte plattformübergreifende Release-Management ermöglicht die effiziente Bereitstellung produktionsnaher Abnahmeumgebungen. Ein plattformübergreifender Umgebungsmanager unterstützt hier das Release-Management. In dieser Abnahmeumgebung können komplette Geschäftsprozesse abgenommen werden. Für die Kunden verbessert sich durch das neue RM sowohl die Qualität des Release-Prozesses als auch die Qualität der eingeführten Softwareänderungen.

6 Zusammenfassung

In dieser Diplomarbeit wurde ein Konzpt zur Integration des plattformspezifischen Release-Managements bei der HVBInfo erstellt. Die HVBInfo betreibt die IT-Infrastruktur von verschiedenen Unternehmen innerhalb der HVB Group.

Zur Bearbeitung dieser Aufgabenstellung wurden zunächst in Kapitel 2 verwandte Arbeiten zum Thema Release-Management (RM) vorgestellt. Nachdem es prinzipiell zwei grundverschiedene Sichtweisen auf das RM gibt, nämlich aus der Perspektive des Softwareengineering (SWE) und des IT-Servicemanagements (ITSM), wurden in Abschnitt 2.1 ab Seite 8 ein kanonisches Lebenszyklusmodell von Software und ein hierauf basierendes Klassifikationsschema eingeführt. Das Lebenszyklusmodell von Software beinhaltet die Phasen Machbarkeitsstudie, Projektinitiierung, Systementwicklung, Integration, Integrationstest, Akzeptanztest, operativer Test, Projektabschluss, Betrieb inklusive Wartung und Einstellung des Produkte.

Bei jeder vorgestellten verwandten Arbeit zum Thema RM und bei der Beschreibung des derzeitigen RM bei der HVBInfo und dem Konzept zum neuen integrierten RM bei der HVBInfo wurde die Phasenabdeckung mit dem Lebenszyklusmodell untersucht bzw. dargestellt. Falls der jeweilige Schwerpunkt im Bereich der Systementwicklung lag, erfolgte eine Zuordnung zum SWE. Bei einem Schwerpunkt im operativen IT-Betrieb erfolgte die Zuordnung zum ITSM. Als weiteres Einteilungsmerkmal wurde gewählt, ob die dargestellte verwandte Arbeit zur Beschreibung oder zur Bewertung von Prozessen dient. Nach diesen vier Klassifikationsmerkmalen wurden alle untersuchten Beiträge zum RM anhand des Klassifikationsschemas eingeordnet (siehe auch Abbildung 6.1).

Mithilfe der Beschreibung dieser verwandten Arbeiten zum Thema RM konnte dargestellt werden, dass die Sichtweise auf das RM uneinheitlich ist. Das RM im SWE-Sinne beschäftigt sich im Wesentlichen mit der Erstellung eines Releases. Ein Release entspricht hier einer bestimmten Version eines einzigen Softwaresystems eines Herstellers. Es konnte gezeigt werden, dass selbst innerhalb des SWE die Auffassungen vom RM unterschiedlich sind. So reichen die Aufgaben innerhalb des RM beim SWE von der Erstellung von Software und / oder deren Konfiguration und Versionierung über die Verfügbarmachung dieser Software bis zur Unterstützung der Einführung der Software beim Kunden.

Die Aufgaben des RM im ITSM dagegen betreffen hauptsächlich die kontrollierte und gesteuerte Einführung von (Software-)Änderungen in eine produktive IT-Infrastruktur. Im ITSM wird auch die Einführung von Hardwareänderungen adressiert, allerdings lag das nicht im Fokus dieser Diplomarbeit und wurde deswegen nicht weiter betrachtet. Aus diesem Grund sind die Aufgaben des RM hier im Rahmen der Release-Steuerung die Festlegung des Release-Umfangs aus allen $1..n$ geplanten und genehmigten Softwareänderungen, die Zusammenstellung eines Releases aus diesen Änderungen, Qualitätssicherungsaufgaben und die Verteilung des Releases in die produktive IT-Infrastruktur. Die Softwareänderungen betreffen hier oft unterschiedliche Software, teils sogar von verschiedenen Herstellern.

Diese vorgestellten, verwandten Arbeiten dienten zur Bearbeitung der Aufgabe dieser Arbeit. Sie wurden zur Beschreibung des bei der HVBInfo bestehenden RM in Kapitel 3 und für dessen Analyse in Kapitel 4 verwendet. Weiterhin dienten sie als Grundlage zur Erstellung des Konzeptes des integrierten RM in Kapitel 5.

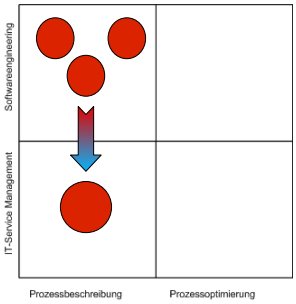


Abbildung 6.1: Verlauf Einordnung

Anhand des Vergleiches der vorgestellten verwandten Arbeiten mit der Darstellung des RM bei der HVBInfo wurde das dort bestehende RM dem SWE zugeordnet. Die HVBInfo ist eine reine Betreiberorganisation ohne nennenswerte eigene Softwareentwickung. Aus diesem Grund wurde im Rahmen dieser Arbeit festgelegt, dass sich der Prozess zukünftig von einem SWE-Verfahren zum ITSM-Verfahren ändert. Diese Verschiebung ist auch aus Abbildung 6.1 ersichtlich.

Als Basis für die Konzeption des neuen RM wurde die IT Infrastructure Library (ITIL) aus Abschnitt 2.3.4 ausgewählt. Die gewählte Ausrichtung des neuen RM am ITIL-RM hat mehrere Vorteile (siehe auch Abschnitt 2.4.2). ITIL ist ein internationaler De-facto-Standard im ITSM und basiert auf praktischer Erfahrung von Nutzern aus verschiedenen Branchen und legt die Basis für eine Zertifizierung nach dem kommenden Standard ISO 20000. Dadurch können die Transparenz der IT-Prozesse, ihre Qualität und auch Reaktionszeiten verbessert werden. Des Weiteren hat die HVBInfo schon die Prozesse Incident-, Change- und Problem-Management an ITIL ausgerichtet. Durch die neue Ausrichtung des RM an ITIL ist es zukünftig besser in die bestehende Prozesslandschaft integriert. Aufgrund der besseren Integration des RM entstehen weniger Reibungspunkte an den Schnittstellen sowie bessere Kommunikations- und Koordinationsbeziehungen. Durch diese konzipierte ITIL-Orientierung des RM ergibt sich ein standardisiertes Release-Verfahren.

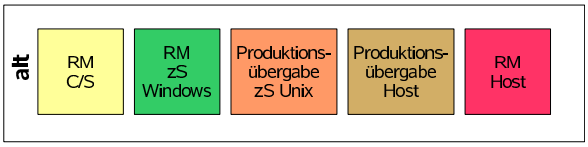


Abbildung 6.2: Ausgangslage: heterogenes Release-Management

Ab Kapitel 3 wurde das derzeit bestehende RM bei der HVBInfo beschrieben. Abbildung 6.2 verdeutlicht die heterogene Release-Prozesslandschaft als Ausgangssituation. Das wesentliche Merkmal des bestehenden RM ist die Technologieorientierung der betrachteten Release-Prozesse. Die drei betrachteten Kategorien bei den Technologien sind im dezentralen Bereich bei den Kunden installierte Windows-Client-Serverarchitekturen (Fat-Clients) (C/S), im zentralen Server Bereich (zS) zentral bei der HVBInfo installierte Windows-, UNIX oder Linux-Server und im Host-Bereich Server auf der Basis von IBM Mainframes (Host). Zu diesen drei Bereichen C/S, zS und Host gibt es bestehende Release-Prozesse. Im dezentralen Bereich wird dieser durch das RM gesteuert. Im Host-Bereich ist ein Release-Prozess unter Steuerung des RM gerade in Einführung. Deswegen wird ein Teil von Softwareänderungen über das RM, der andere Teil im Rahmen von Softwareentwicklungsprojekten durch den Projektleiter eingeführt. Im Bereich zS werden Betriebssysteme und betriebssystemnahe Software im Windows-Bereich duch das RM, im Bereich von Windows-Applikationen und im UNIX- und Linuxumfeld durch den jeweiligen Projektleiter von Softwareentwicklungsprojekten gesteuert. Dieser Projektleiter ist oft organisatorisch bei der HVBSystems angesiedelt. Der Prozess zur Einführung von Änderung in die Produktivumgebung eines Kunden wird dort Prozess zur Produktionsübergabe genannt.

Durch diese Heterogenität ergeben sich verschiedene Nachteile (siehe auch ab Kapitel 4). Die jeweiligen Release-Prozesse sind eigenständig und kaum miteinander verknüpft. Dadurch entstehen hohe Aufwände bei der Kommunikation und Koordination, falls die anderen Technologiebereiche doch benötigt werden. So sind derzeit im Rahmen der Einführung von Softwareänderungen z.B. keine Abnahmen von Geschäftsprozessen der Kunden möglich, weil jeder Technologiebereich für sich betrachtet wird und es keine technologieübergreifenden Prozesse gibt. Innerhalb der jeweiligen Release-Prozesse haben sich Release-Begriffe mit unterschiedlicher Bedeutung entwickelt. Diese verschiedenen Begriffswelten verringern das Verständnis der Mitarbeiter untereinander. Das RM ist im Vergleich zum RM bei ITIL relativ spät in den Prozess der Einführung von Softwareänderungen eingebunden worden. Aus diesem Grund erfolgt die Planung der benötigten Ressourcen kurzfristig oder ist manchmal gar nicht möglich. Viele Prozessbeteiligten arbeiten deshalb reaktiv, wodurch sich Mehrarbeit und Auslastungsspitzen ergeben.

Im Rahmen dieser technologieorientierten Release-Prozesse haben sich verschiedene Release-Arten etabliert. Im dezentralen Bereich werden möglichst viele Änderungen verschiedener Produkte in so genannten Standard-Releases gebündelt. Diese werden dann gemeinsam in die Produktivumgebung eingeführt. Beim Host dagegen wird beim Hotfix-Prozess jede Änderung für sich gehandhabt. Bei den zentralen Servern werden Applikations-Releases, d.h. Änderungen an einer Applikation, über genehmigte Changes eingeführt. Die bestehenden Release-Arten Standard-Release, Sonderversorung oder -Release, Hotfix, Hotrelease, Securityfix oder Change

umfassen verschiedene Merkmale wie etwa Release-Durchlauf, Umfang der Tests und der damit verbundenen Testumgebungen, Release-Inhalt oder Dringlichkeit der Durchführung.

Die Release-Planung durch das RM umfasst hauptsächlich die vorherige Reservierung der entsprechenden Testumgebungen für Installations-, Integrations-, Last-, Performanz- oder Abnahmetests. Diese Testumgebungen werden für die Testphasen IT und QS benötigt. In der IT-Phase erfolgen teilweise noch Integrationsaufgaben. Sie wird im dezentralen Bereich häufig aufgrund auftretender Fehler wiederholt, während sie im Host-Bereich bei Hotfixes und Hotreleases ganz entfällt. Der Umfang und Inhalt dieser eingeplanten Release-Durchläufe ist vorab nicht bekannt. Die Softwareentwickler geben erst im Rahmen der so genannten Einmeldephase ihre Änderungen dem RM bekannt. Der Zeitpunkt dieser Bekanntgabe liegt meist am Ende des Softwareentwicklungsprozesses. Im Vergleich zum Vorgehen in ITIL, wo das RM vor Beginn der Softwareentwicklungstätigkeiten beginnt, ist das relativ spät. Aus diesem Grund haben die am Prozess beteiligten Stellen, wie etwa Integratoren oder Umgebungsverantwortlichkeiten kaum die Möglichkeit zur Planung ihrer Ressourcen und arbeiten deshalb reaktiv. Das ist oft mit Mehrarbeit und Auslastungsspitzen bei den betroffenen Mitarbeitern verbunden.

Die Abwicklung der Einführung von Softwareänderungen wird im dezentralen Bereich und im Host-Bereich teilweise durch Werkzeuge auf Basis eines Versionierungssystems unterstützt. Die Belegung und Reservierung der entsprechenden Testumgebungen erfolgt ohne Werkzeugunterstützung. Vor allem im Bereich zS erfordert das einen hohen Koordinierungsaufwand. Die unzureichende Unterstützung durch geeignete Tools zeigt sich durch das Führen verschiedener (Excel-)Listen durch die Release-Manager.

Obwohl die Prozesse zur Einführung von Änderungen entweder vom Projektleiter der HVBSystems oder vom RM bei der HVBInfo gesteuert werden, haben alle den Fokus im SWE. Abdeckt werden bisher die Phasen In-

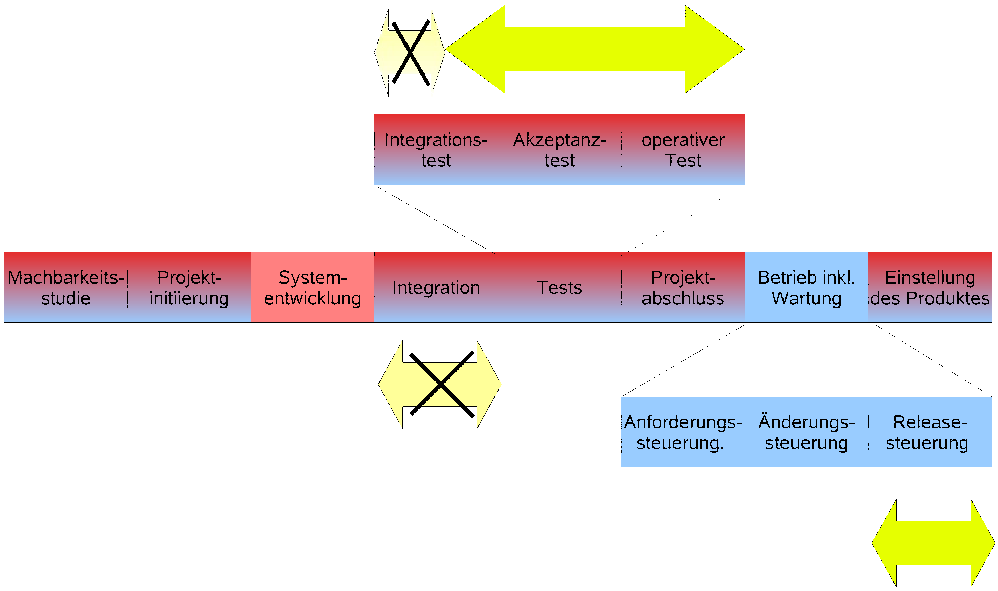


Abbildung 6.3: Änderungen im Lebenenszyklus

tegration, Integrationstest, Akzeptanztest und operative Tests des im Kapitel 2 vorgestellten Lebenszyklusmodells. Die Phase der Release-Steuerung, in der vorab der Release-Umfang festgelegt wird, mit der Verbindung zum Änderungssteuerungsprozess (ITIL Change-Management) ist nicht vorhanden. In dieser Arbeit wurde die Ausrichtung des RM an ITSM und speziell an ITIL als Anforderung abgeleitet. Diese neue Ausrichtung ist auch in Abbildung 6.3 dargestellt.

In dieser Arbeit wird empfohlen, dass die Integrationsphase zukünftig kein Bestandteil des RM mehr ist. Sie ist

eine eigenständige Phase, in der jeweils plattformspezifische Integrationsaufgaben anfallen. Integrationstests fallen aus diesem Grund im neuen Release-Prozess nur noch aufgrund der Integration von geänderter Software in die spezifische Testhardware an. Beibeihalten wurden die Phasen Akzeptanz- und operativer Test. Das RM koordiniert und steuert hierfür die Bereitstellung von produktionsähnlichen Abnahmeumgebungen und die Testaktivitäten. Als neue Phase wurde der Release-Steuerungsprozess unter Verantwortung des Release-Managers definiert.

Abbildung 6.4 verdeutlicht, dass dieser neue Release-Steuerungsprozess plattformunabhängig ist.

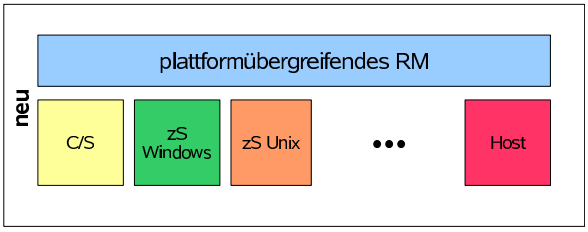


Abbildung 6.4: Plattformübergreifendes Release-Management

Aus diesem Grund ist er auch für die Einführung von in der Diplomarbeit nicht betrachteten Bereichen wie z.B. die Einführung von Änderungen an Speicher- oder Telekommunikationssystemen geeignet. Um eine Plattformunabhängigkeit zu erreichen, wurden hier im Rahmen des Konzeptes neue Rollen definiert. Neue Rollen im Rollenmodell sind der plattformübergreifende Release-Manager, Umgebungsmanager und Integrator. Weitere neue Rollen sind der Konfigurationsmanager und Kundensetmanager. Der Kundensetmanager definiert den Inhalt von Kundensets. Das Konfigurationsmanagement entspricht dem ITIL-Konfigurationsmanagement. Seine Einführung wird in dieser Arbeit zwar gefordert, allerdings nicht weiter betrachtet. Das ist die Aufgabe weiterer Projekte. Ebenfalls in weiterführenden Projekten müssen die in der Arbeit geforderten Tools zur Workflow-Unterstützung und Unterstützung des Umgebungsmanagements und die geforderten Datenbanken CMDB des Konfigurationsmanagements und die DSL des RM spezifiziert und eingeführt werden.

Das neue Release-Management ist im ITSM-Bereich angesiedelt und wurde an ITIL orientiert. Dadurch besteht zukünftig die Verknüpfung zum Change-Management und Konfigurationsmanagement. Im Rahmen des Change-Prozesses muss jetzt jede geplante Änderung genehmigt werden. Früher war das nicht immer notwendig. Das Release-Management ist dann in den Gehnehmigungsprozess durch die Bestätigung der geplanten Termine involviert. Der Release-Plan wird dann zukünftig proaktiv festgelegt. Er ergibt sich aus der geeigneten Bündelung der genehmigten Änderungen. Dadurch ist eine langfristige Planung der Ressourcen möglich. Durch die ITIL-Orientierung wird auch das Konfigurationsmanagement eingeführt. Es unterstützt das RM und am Release-Prozess beteiligte Mitarbeiter durch Informationen, mit denen z.B. die Auswirkung von geplanten Änderungen vorher festgelegt werden, was bisher nicht möglich ist. Der Umgebungsmanager hat Informationen für die Konfiguration der Testumgebungen, der Integrator hat Zugriff auf entsprechende Installationsanweisungen und der Release-Manager kann die entsprechenden Testaktivitäten besser koordinieren.

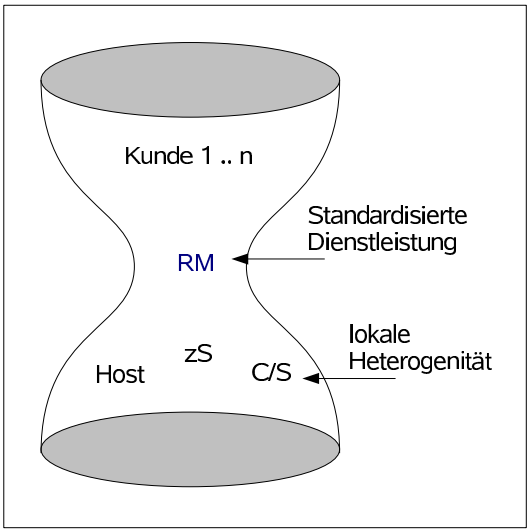


Abbildung 6.5: Release-Management als Schnittstelle nach Sanduhr-Modell

Das in der Arbeit konzipierte plattformübergreifende RM ist generisch angelegt. Wie in Abbildung 6.5 dargestellt, bildet das RM zukünftig eine standardisierte Dienstleistung für die Einführung von Änderungen an der IT-Infrastruktur. Diese Dienstleistung können alle Kunden in Anspruch nehmen. Das RM schattet hierbei die Heterogenität der benötigten Ressourcen ab. Die notwendigen Ressourcen bilden den unteren Teil der Sanduhr. Sie können beliebig geändert oder erweitert werden, solange der Hals der Sanduhr, das RM, beibehalten wird.

Durch die Standardisierung des Release-Prozesses ergibt sich eine Transparenz bezüglich seiner Beziehungen zu anderen Prozessen und erleichtert damit die Anpassung an geänderte Bedingungen. Durch die Standardisierung können bestehende Prozesse leichter optimiert und Kosten gesenkt werden [ZaBr 04].

Obwohl Releases aller Plattformen über den vorgeschlagenen Prozess abgedeckt werden können, sollte weiterhin

überprüft werden, ob anstelle der Fat-Clients des dezentralen Bereichs eine sukzessive Umstellung auf Thin-Clients vorgenommen wird. Vorteile durch den Einsatz von Thin-Clients ergeben sich laut Teichner u.a. durch reduzierte Betriebskosten, geringere Komplexität und einfachere Sicherheitskonzepte [Teic 06].

Des Weiteren sollten in Zukunft alle kritische Änderungen an der IT-Infrastruktur in der vorproduktiven Abnahmeumgebung getestet werden. In der Vergangenheit gab es beispielsweise Performanzprobleme im produktiven Umfeld, weil die Organisationseinheit Security einen *Trojaner* direkt im produktiven Umfeld zur Überprüfung der *Firewall* getestet hat. Solche Tests sollen zukünftig in der Abnahmeumgebung stattfinden. Das Ziel sollte hier sein, alle Änderungen- d.h. neben Software- auch Hardwareänderungen- über diesen einen Release-Prozess abzuwickeln.

Der standardisierte Release-Prozess legt die Grundlage für weitere Automatisierungsaktivitäten. Potenzial zur Automatisierung liegt im Bereich der Deployment- und Testaktivitäten. Bei den Deployment-Aktivitäten ist der Host-Bereich durch die Verwendung von KVS am weitesten fortgeschritten. Im dezentralen Bereich und Bereich zS muss untersucht werden, ob weitere Standardisierung und auch Automatisierung von Tätigkeiten möglich sind. Die in Abschnitt 2.3.9.1 auf Seite 56 beschriebenen Ansätze durch Softwarekonnektoren oder das *Software Dock* in Abschnitt 2.2.6.6 ab Seite 29 können hierfür erste Ansätze liefern. Das Rahmenwerk von Carzaniga et. al. aus Abschnitt 2.2.6.6.1 auf Seite 29 eignet sich ebenso zur weiteren Untersuchung der bestehenden Deployment-Prozesse. Die Basis zur Testautomation wurde bei der HVBInfo durch die Einführung des Automatisierungswerkzeugs *SilkTest* von Segue [Segu 06] bereits gelegt. Hiermit sollen möglichst viele Testfälle automatisiert durchgeführt werden. Dadurch kann sich der Durchlauf eines einzelnen Release-Prozesses in Zukunft reduzieren.

Mithilfe weiterer Standardisierung und Automatisierung soll dann das weitere Ziel ein flexibler Release-Prozess sein. Anstelle der bisher vorab getakteten Produktionseinführungen soll die Produktionseinführung abhängig vom jeweiligen Änderungsbedarf erfolgen. Ähnlich wie bei der von Thanheiser et. al. in Abschnitt 2.2.6.6.1 auf Seite 31 vorgestellten Softwaretankstelle oder dem bei Andersson referenzierten Push-Modell in Abschnitt 2.2.6.6.2 betanken sich dann die jeweiligen Zielumgebungen automatisch beim Vorliegen entsprechender Änderungen. Hierfür ist neben der Einführung der DSL die ständige Verfügbarkeit der Abnahmeumgebung nötig. Durch diese ständige Verfügbarkeit der entsprechenden Abnahmeumgebung kann jederzeit ein Release fertiggestellt und abgenommen werden. Erzielt werden kann diese ständige Nutzbarkeit durch den Einsatz weiterer virtueller Umgebungen. In der DSL werden dann diese fertiggestellten Releases abgelegt und von dort bei Bedarf in die Zielumgebung verteilt.

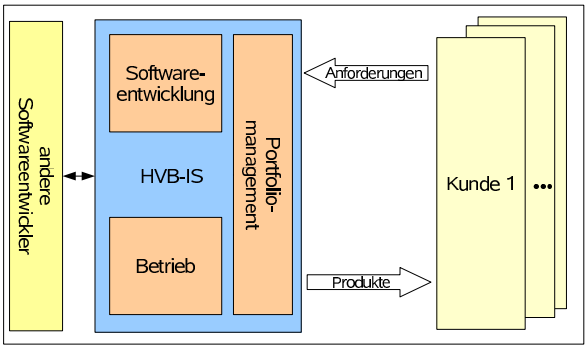


Abbildung 6.6: Schnittstelle zwischen Kunde und Leistungserbringer nach [ZBP 05]

Die laufenden Änderungen an den bestehenden Release-Prozessen durch Projekte wie AQUA und ReQuest wurden soweit als sinnvoll berücksichtigt. Weitere Anpassungen durch resultierende Änderungen aufgrund der Fusion der HVB mit der italienischen UniCredit und der Fusion der HVBInfo mit der HVBSystems (zukünftig als HVB Information Service (HVB-IS)) werden dennoch nötig sein. Vor allem die Fusion der HVBInfo mit der HVBSystems bietet die Chance, die in Abschnitt 3.2 von Zarnekow [ZBP 05] zitierte traditionelle Organisationsform in eine produktorientierte umzubauen (siehe Abbildung 6.6). Bei der bestehenden Organisationsform entsteht laut Zarnekow für den Kunden eine unnötige Komplexität aufgrund von Absprachen mit zwei Vertragspartnern (Softwareentwickler und Betreiber). Dadurch entstehen in der Praxis oft Probleme durch gegenseitige Problemzuweisung zwischen Entwicklung und Produktion. Es ist daher oft schwierig, einen tatsächlich Verantwortlichen zu finden. Diese zitierten Probleme können derzeit auch beobachtet werden. Durch die in der Abbildung dargestellte Produktorientierung kauft der Kunde zukünftig IT-Produkte. Diese beinhalten sowohl Entwicklungs- als auch Betriebsleistungen. Der Leistungsanbieter muss deshalb zur Unterstützung seiner Kunden Wissen über die Geschäftsprozesse der Kunden sammeln. Dieser Ansatz ist bei der HVBInfo bereits durch eine neue Abteilung (Inf1GP) im Entstehen. Die Schnittstelle zum Kunden bildet dann das Portfolio-Management. Die Entwicklung und der Produktionsbereich sind zukünftig dann gleichberechtigte Bereiche.

In dieser Arbeit wird vor allem deutlich, dass die Schnittstellen zwischen dem SWE und dem ITSM unklar sind. Das ist besonders am Release-Management erkennbar, dass zu beiden Disziplinen zugeordnet werden kann. Es gibt, wie in der Arbeit vorgestellt, zahlreiche Referenzmodelle zum SWE und auch einige im Bereich des ITSM. Allerdings gibt es kein übergreifendes, formalisiertes Referenzmodell, welches diese beiden Sichtweisen integriert. Das wurde auch von Brenner et. al. [BGN 06] am Beispiel des *Requirements Engineerings* verdeutlicht. Aufgrund dieses Fehlens eines übergreifenden Modells kommt es in der Praxis häufig zu Problemen, weil die Schnittstellen nicht klar geregelt sind oder die Koordination und Kommunikation vor allem im unternehmensübergreifenden Umfeld schwierig ist. Aus diesem Grund ist hier die Forschung gefragt, ein solches übergreifendes Referenzmodell zu entwickeln.

Weiterhin wird in dieser Arbeit deutlich, dass für die Einführung neuer oder geänderter Software in eine bestehende IT-Infrastruktur sehr viel Aufwand betrieben wird. Aufwände entstehen hier vor allem bei der Anpassung dieser Software an die lokalen Gegebenheiten und beim Testen. Zur Reduzierung dieser Aufwände im Deployment-Bereich sind zukünftig weiterführende Forschungsarbeiten zu den in der Arbeit bereits aufgeführten, wie etwa zur Arbeit von Bulej et. al. zu Softwarekonnektoren für das Deployment heterogener Komponenten in Abschnitt 2.3.9.1.1, nötig. Die Aufwandsreduzierung beim Testen läasst sich durch eine bessere Qualität der Softwareprodukte erreichen. Obwohl das SWE im Vergleich zum ITSM die ältere Disziplin ist, besteht hier im Vergleich zu anderen Industrieprodukten immer noch eine relativ geringe Qualität der erzeugten Software. Das ist daran erkennbar, dass sowohl die Abnehmer der Software als auch die Hersteller selbst automatisch mit dem Auftreten von Fehlern bei der Software rechnen. Im konkreten Fall der HVBInfo zeigt sich das an schon vorher fest eingeplanten Iterationsschleifen im Integrations- und Testablauf. Im Bereich der Softwareentwicklung verdeutlicht sich das am Beispiel von Jansen et. al. in Abschnitt 2.2.6.1. Sie schlagen ein Kommunikationsrahmenwerk vor, mit dessen Hilfe beim Softwarebenutzer entstandene Fehler automatisch an die Softwarehersteller zurückgemeldet werden können. Diese Institutionalisierung der geringen Softwarequalität durch ein Kommunikationsrahmenwerk ist sogar bei kommerziellen Softwareanbietern beobachtbar. Aus diesem Grund muss auch im SWE die Forschung zur Erzeugung besserer Softwarequalität weiter betrieben werden. Laut Crosby sollte anstatt eines akzeptablen Qualitätsniveau der angestrebte Qualitätsstandard immer Null-Fehler sein [Cros 96].

Abbildungsverzeichnis

1.1 Vorgehen bei der Diplomarbeit 3

2.1 kanonisches Lebenszyklusmodell nach [MD 93] 9
2.2 kanonischer Lebenszyklus im Kontext der Diplomarbeit 9
2.3 Aktivitäten innerhalb der Systemerstellung 10
2.4 Testarten innerhalb der Testphase 10
2.5 Betrieb und Wartung 11
2.6 Einordnung von SWE und ITSM in das Lebenszyklusmodell 13
2.7 Schema zur Klassifikation von RM 14
2.8 Einordnung V-Modell 15
2.9 Submodell Systemerstellung des V-Modells nach [V-Modell-c] 15
2.10 Überleitung in die Nutzung, [V-Modell] 16
2.11 Einordnung des V-Modells in das Lebenszyklusmodell 17
2.12 Einordnung V-Modell XT 17
2.13 Entscheidungspunkte und Projektdurchführungsstrategien im V-Modells XT nach [V-Modell XT] 17
2.14 Einordnung des V-Modells XT in das Lebenszyklusmodell 18
2.15 Einordnung von MSF 18
2.16 Rollen des MSF Teammodells 19
2.17 MSF Prozessmodell 20
2.18 Einordnung des MSF in das Lebenszyklusmodell 20
2.19 Einordnung RUP 21
2.20 Dimensionen von RUP aus [RUP] 21
2.21 Einordnung von RUP in das Lebenszyklusmodell 23
2.22 Einordnung CMMI 23
2.23 Einordnung von CMMI in das Lebenszyklusmodell 24
2.24 Einordnung der Prozesse 24
2.25 Softwarelogistikprozesse und Datenbanken, [Gree 00] 24
2.26 Customer Configuration Updating (CCU) aus [Jan05] 25
2.27 *Software Delivery Model* [Ball 05] 26
2.28 Einordnung Wartung 26
2.29 Häufigkeit der Anforderungen nach Kategorie (aus [SOSA 99]) 27
2.30 Einordnung des Modells von Stark et. al. in das Lebenszyklusmodell 28
2.31 Repository- und Weiterleitungs-Schema aus [JBBvN 05] 29
2.32 Einordnung RM bei Carzaniga et. al. 29
2.33 Deployment-Prozess aus [Carz 97, S. 3] 30
2.34 Framework für Deploymentprozesse nach [CFH+ 98] 30
2.35 Einordnung RM bei Andersson 31
2.36 Einordnung RM im KM 32
2.37 Phasen in Software- und Hardwareentwicklung, aus [CAPD 03, S. 7] 32
2.38 Einordnung in das Lebenszyklusmodell (Gillis et. al.) 33
2.39 Einordnung Modell von Balzert 33
2.40 Einordnung der Abnahme- und Einführungphase bei Balzert in das Lebenszyklusmodell . . . 34
2.41 Einordnung Modells von Steinweg 34
2.42 Einordnung des Modells von Steinweg in das Lebenszyklusmodell 35
2.43 Einordnung SWE-BOK 35
2.44 Ebenen von SWE-BOK 36

2.45 Einordnung von SWE-BOK in das Lebenszyklusmodell 37
2.46 Einordnung von MOF 37
2.47 MOF Quadranten und Reviews 37
2.48 Zusammenhang zwischen Change- und Release-Management bei MOF nach [Mof05] 38
2.49 Release-Prozess bei MOF nach [Mof05] 38
2.50 Einordnung von MOF und MSF in das Lebenszyklusmodell 39
2.51 Einordnung SMART 39
2.52 Einordnung von SMART in das Lebenszyklusmodell 39
2.53 Einordnung SERVIAM 40
2.54 Rollenmodell von EM^3 41
2.55 zeitlicher Prozessablauf des RM bei EM^3 41
2.56 Einordnung von EM^3 in das Lebenszyklusmodell 42
2.57 Einordnung von ITIL 43
2.58 ITIL Prozesse Release- , Change- und Konfigurations-Management nach [Olbr 04] 44
2.59 Software-Release-Einheiten nach [OGC04] 45
2.60 Beispiel: verschiedene Release-Typen in ITIL 46
2.61 Aktivitäten im Release-Prozess und betroffene Umgebungen nach [vKP 02] 47
2.62 Einordnung von ITIL in das Lebenszyklusmodell 49
2.63 Einordnung CobIT 50
2.64 Einordnung MITO 51
2.65 Prozessgruppen von MITO 51
2.66 Einordnung von MITO in das Lebenszyklusmodell 53
2.67 Einordnung SMCMM 53
2.68 Kontexteinbettung von Software Maintenance laut [AAD 04b] 53
2.69 Einordnung von SM^{CMM} in das Lebenszyklusmodell 54
2.70 Einordnung IT-SCMM 54
2.71 Diskrepanz zwischen wahrgenommener und gelieferter Servicequalität nach [Niva 00] 55
2.72 Einordnung des Modells von Niessink et. al. in das Lebenszyklusmodell 56
2.73 Einordnung der Prozesse 56
2.74 Einordnung des OMG Modells in das Lebenszyklusmodell 57
2.75 Phasen der Softwarewartung aus [CuGi 89] 58
2.76 Einordnung der Wartung nach Curth et. al. in das Lebenszyklusmodell 58
2.77 Übersicht Einordnung aller beschriebenen Modelle 59
2.78 Übersicht Einordnung einer Auswahl der beschriebenen Referenzmodelle in das Lebenszyklusmodell 59
2.79 ITIL Service Support und Delivery im Vergleich zu SOITM nach [HHB 05] 63

3.1 Schnittstellen zwischen HVBInfo, Entwicklungsorganisationen und Kunden 67
3.2 Gegenüberstellung Release-Einheit ITIL und HVBInfo 69
3.3 Zuordnung der Teilsysteme zu den Plattformen 71
3.4 Phasenmodell des Testprozesses innerhalb des Release-Prozess bei der HVBInfo 74
3.5 Schematische Darstellung der unterschiedlichen Release-Taktung je Plattform 75
3.6 Problem durch überholendes Release 76
3.7 Reduzierung der Incidents bei der Integration der Vereins- und Westbank 76
3.8 Einordnung RM dezentral 77
3.9 Übersicht Change- und Release-Management und Softwareentwicklung im Bereich C/S 80
3.11 Zusammenhang zwischen Release, Einmeldung und Produkt 80
3.10 Darstellung der wichtigsten Release-Aktivitäten im dezentralen Bereich 81
3.12 Auszug aus der Übersicht aller Release-Pläne der HVBInfo 84
3.13 Auszug aus einem Release-Plan der HVBInfo 85
3.14 Zuordnung von Prozessphasen zu Umgebungen im dezentralen Bereich 86
3.15 Logistikprozess bis zur Produktion 89
3.16 Ablauf des Release-Steuerungsprozesses bei der Planung 90
3.17 Ablauf des Release-Steuerungsprozesses bis zum Start der IT-Phase 91
3.18 Zustandsübergänge des Releases definiert durch das Release-Begleitblatt 92
3.19 Zustandsübergänge eines Produktes definiert durch das Produktbegleitblatt 93

3.20 Release-Inhalt durch die Einmeldung 94
3.21 Ablauf des Release-Steuerungsprozesses in der IT- und QS-Phase 96
3.22 Ablauf des Release-Steuerungsprozesses ab Start Pilot-Phase 97
3.23 Fehlerverlauf im Release-Prozess 98
3.24 Einordnung des dezentralen RM in das Lebenszyklusmodell 99
3.25 Einordnung RM zS 99
3.26 Übersicht Change-Management und Softwareentwicklung im Bereich zS 100
3.27 Übersicht der wichtigsten Phasen im Produktionsübergabeverfahren 103
3.28 Auszug aus den PRK der HVBSystems 105
3.29 Zusammenhang Realisierung, Verifikation & Validierung aus ZAD HVBSystems 106
3.30 Übersicht der wichtigsten Phasen im Produktionsübergabeprozess 107
3.32 Zuordnung von Prozessphasen zu Umgebungen im Bereich zS 107
3.31 Testprinzip im V-Modell aus [WSK 05] 108
3.33 Darstellung der Softwareversorgung mit Staging im Bereich zS 109
3.34 Darstellung der Aktivitäten bis zur Theke 111
3.35 Darstellung der Aktivitäten zur Produktionsübergabe 112
3.36 Abdeckung des Lebenszyklusmodell durch das ALADIN-Vorgehensmodell 113
3.37 Einordnung RM Host 113
3.38 Gegenüberstellung der Prozessabläufe im Zusammenspiel mit Release- und Change-Management, Softwareentwicklung und Integration 114
3.39 Zusammenhang zwischen AWS, KVS-Package und Komponente 115
3.40 Toolzuordnung im Host-Bereich 117
3.41 Auszug aus dem Host-Release-Plan für Standard-Releases 118
3.42 Ebenen des KVS 121
3.43 Übersicht Ablauf Weiterleitung durch KVS-Ebenen 122
3.44 Zusammenhang zwischen Einmeldung, AWS, KVS-Package und PRUEV 123
3.45 Gegenüberstellung der Prozessphasen und benötigten Umgebungen bei den verschiedenen Release-Prozessen 125
3.46 Fehlerverlauf Standard-Release Host in IT-(GPL)-Phase bei 4 AWS 125
3.47 Einordnung des RM-Host in das Lebenszyklusmodell 126
3.48 Übersicht der wichtigsten Merkmale aller Release-Arten 128

4.1 Gegenüberstellung der ITIL-Prozesse und der Prozesse der HVBInfo im dezentralen Bereich . 130
4.2 W-Modell aus [Spil 03] 134
4.3 Fehlertypen und Häufigkeit bei Releases 2005 (KVZ) 135
4.4 Gegenüberstellung der ITIL-Prozesse und der Prozesse der HVBInfo im Bereich zS 136
4.5 Gegenüberstellung der Prozessabläufe im Zusammenspiel mit Release- und Change-Management, Softwareentwicklung und Integration im Host-Bereich 138
4.6 Anzahl der PRUEVs in Hotrelease und Hotfix 139
4.7 Bestehende und neue Prozessgrenzen 141
4.8 Anzahl der dezentralen Hotfixes und Sonderversorgungen je Monat im Jahr 2005 144
4.9 Skizze Darstellung toolunterstützte Umgebungsbelegung 145
4.10 Gegenüberstellung bestehender und neuer Kommunikationsbeziehungen 146

5.1 Einordnung des neuen Prozesses 149
5.2 Übersicht RM mit Change-, Konfiguration-, Umgebungsmanagement und Softwareentwicklung im Gesamtprozess 150
5.3 Übersicht der Release-Aktivitäten 154
5.4 Beispiel Release-Einheiten im Windows-Bereich 155
5.5 Zusammenhang zwischen Release, Änderung, Change und Komponenten 155
5.6 Beispiel eines plattformübergreifenden Release-Plans 158
5.7 Übersicht der Prozessphasen und benötigten Umgebungen 160
5.8 Darstellung der Phasen im Release-Steuerungsprozess 161
5.9 Schnittstellen der HVBInfo in Verbindung mit Einmeldung bzw. Change 162
5.10 Aktivitäten in der QS-Phase bis zum Start der Abnahme 164
5.11 Aktivitäten in der QS-Phase während der Abnahmen 165

5.12 Aktivitäten in der Pilotphase 166
5.13 Einordnung des neuen RM in das Lebenszyklusmodell 167

6.1 Verlauf Einordnung 171
6.2 Ausgangslage: heterogenes Release-Management 171
6.3 Änderungen im Lebenenszyklus 172
6.4 Plattformübergreifendes Release-Management 173
6.5 Release-Management als Schnittstelle nach Sanduhr-Modell 173
6.6 Schnittstelle zwischen Kunde und Leistungserbringer nach [ZBP 05] 174

Tabellenverzeichnis

2.1 Schlüsselqualitätsziele und zugeordnete Rollen in MSF 19
2.2 Auszug der Taxonomiy der Anforderungsarten aus [SOSA 99] 27
2.3 Gegenüberstellung Anwendungsentwicklung vs. *Support & Maintenance* nach [Thom 05] . . 60

3.1 Release-Begriffe in der HVBInfo 72
3.2 Release-Begriffe in der HVBInfo im Vergleich zu ITIL 73
3.3 Rollen und Aufgaben im dezentralen Bereich 79
3.4 Übersicht der Testarten, Testphase im dezentralen Bereich 87
3.5 Übersicht Status Release-Begleitblatt zum Status im Release-Steuerungsprozess 93
3.6 Rollen und Aufgaben im Bereich zS 102
3.7 Kategorien und Auswirkungsklassifikation von Changes in Impact 103
3.8 Auszug aus den erwarteten Begleitdokumenten aus ALADIN 104
3.9 Übersicht der Testarten, Testumgebung und Tester im Bereich zS laut Testkonzept der HVB-Systems 108
3.10 Übersicht der Verteil- und Installationsarten im Bereich zS 110
3.11 Übersicht Einstufungskriterien eines Systems in die Risikoklasse 116
3.13 Übersicht der Testarten, Testumgebung und Tester im Host-Bereich laut Testkonzept der HVB-Systems 119
3.15 Übersicht der Verteil- und Installationsarten im Bereich Host 120

4.1 Informationsvergleich von Change und Einmeldung 131

5.1 Rollen und Aufgaben beim integrierten Gesamtprozess 153

Glossar

AQUA Abnahme Qualität - Projekt zur Einführung eines Release-Managements im Host-Bereich

AgeNT Anpassung der Entwicklungsmethodik im NT Umfeld - durch Einführung eines einheitlichen Versionierungs- und Konfigurierungssystems auf Basis von PVCS-Dimensions

AI Anwendungsintegrator

ALADIN Application Development Information Navigator

ASM Application Support and Maintenance

ASU-Ampel Abnahmeumgebungsstatus-Ampel - Aktueller Status der Abnahmeumgebungen wird mit Ampelfarben dargestellt

AWS Anwendungssystem

AWSV Anwendungssystemverantwortlicher

BaFin Bundesanstalt für Finanzdienstleistungsaufsicht

BSI Bundesamt für Sicherheit in der Informatik

C/S Client/Server

CCB Change Control Board

CCU Customer Configuration Updating

CI Configuration Item

CMDB Configuration Management Database

CMMI Capability Maturity Model Integration

CobIT Control Objectives for Information and Related Technology

Common Softwarebündel aus allgemein einsetzbarer Software

CORBA Common Object Request Broker Architecture

Core Softwarebündel aus Betriebssystem und betriebssystemnaher Software

CRM Customer Relationship Management

CSF critical success factor

Custom Softwarebündel aus kundenspezifischer Software

DHS Definitive Hardware Store

DSL Definitive Software Library

EFQM European Foundation for Quality Management

EI Einsatzverantwortlicher

EM Einführungsmanager

EM^3 Evolution and Maintenance Maturity Model

ENDEVOR/MVS Environment for Development and Operation

EU Entwicklungsumgebung

EV Einführungsverantwortlicher

Frozen Zone Vom Kunden festgelegter Zeitraum, in dem kein Roll-out stattfinden darf

FSW Fremdsoftware-Teilsystem

GET gemanagte Entwicklungsumgebung im Host-Bereich

GPL gemanagte Prelife-Umgebung - produktionsähnliche Abnahmeumgebung

HVBInfo HVB Informations-Verarbeitungs-GmbH

HyperDB Datenbank, mit der Lokationen, Rechner und Releases verwaltet werden

I Installation

IOC Initial Operational Capability

ISACA Information Systems Audit and Control Association

ISKB Intelligent Software Knowledge Base

IT-Phase Integrationstestphase

IT-U Integrationstestumgebung

IT-Umgebung Integrationstestumgebung

ITIL IT Infrastructure Library

ITSM IT-Servicemanagement

itSMF IT Service Management Forum

KM Konfigurationsmanagement

KPA Key Process Areas

KPI Key-Performance-Indikator

KVP kontinuierlicher Verbesserungsprozess

KVS Komponentenverwaltungsystem

KVZ Kundenset für Vertrieb und Zentrale

LCA Life-Cycle Architecture

LCO Life-Cycle Objective

MCB HVB Corporates & Markets

MITO Maturity Model for IT Operations

MOF Microsoft Operations Framework

MSF Microsoft Solutions Framework

MSI Microsoft Windows Installer

NDM Netview Datasupply Manager

OLA Operational Level Agreement

OMG Object Management Group

PARIS Product and Release Integration Suite - Datenbank zur Konfigurationsunterstützung

PDM Product Data Management

PIR Post Implementation Review

PKG Paketmanager

PM Projektmanagement

PMS Problem Monitoring System

PR Produkt-Release

PREP Produktionseinsatz und -planung

PRK Production Readiness Kriterien - Auflistung erwarteter Artefakte als Ergebnis des Softwareentwicklungsprozesses

PROD Produktivumgebung

PRUEV Produktions Übergabe / Übernahme Verfahren für Host-Anwendungen

PV Produktverantwortlicher

PÜ Produktionsübergabe

PÜ Produktionsübergabeumgebung, d.h. Pilotumgebung für Installationstests

QS Qualitätssicherung

QS-Phase Qualitätssicherungsphase

QS-U Qualitätssicherungsumgebung

QS-Umgebung Qualitätssicherungsumgebung

RAB Release Advisory Board

RE Requirements Engineering

ReQuest Release Quality established - Projekt im dezentralen RM mit Hauptfokus Einzelproduktversorgung und Virtualisierung von IT-U

RfC Request for Change

RM Release-Management

RM-C/S Release-Manager Client/Server

RM-Host Release-Manager im Host-Bereich

RMG Release-Management-Gremium

RPM Red Hat Package Manager zum (De-)Installieren und Aktualisieren von Software und Zusammenstellen von Softwarepaketen

RRR Release Readiness Review

RUP Rational Unified Process

SAQ Swiss Association for Quality

SB-Services Selbstbedienungs-Services

SCM Software Configuration Management

SCOR Supply-Chain Operation Reference

SE Systemerstellung

Securityfix (C/S) Release zur schnellen Behebung von Sicherheitsproblemen im laufenden Betrieb

SEI Software Engineering Institute der Carnegie Mellon University

SI Systemintegrationsumgebung

SI Systemintegrator

SIM-Portal Dokumentationstool im Intranet der HVBInfo basierend auf HiScout

SIM-Portal System Information Management-Portal; Instrument basierend auf *HiScout* zum dokumentieren und veröffentlichen von Dokumenten

SLA Serverice Level Agreement

SLA Service Level Agreement

SM^{CMM} Software Maintenance Capability Maturity Model

SMART Solution Management and Request Tracking

SMF Service Management Function

SOITM Serviceorientiertes IT-Management

Sonderversorgung C/S Release zu einem bestimmten Termin, außerhalb der Standard-Releases

SPICE Software Process Improvement Capability Determination

SPR Software Problem Reports

SRM Software Release-Management

Standard-Release (C/S) Release, dass Standardprozess durchläuft und dessen Anzahl vorab festgelegt wird

Stargate Tool zur Unterstützung des Release-Prozesses im Bereich C/S

SWE Softwareengineering

SWE-BOK Software Engineering Body of Knowledge

SWV Softwareversorger

UC Underpinning Contract

UM Umgebungsmanager

UML Unified Modelling Language

Updaterelease (C/S) Release, in dem nur veränderte Software gebündelt wird

UV Umgebungsverantwortlicher

V Versorgung bzw. Verteilung

Voll-Release (C/S) Release, in dem geänderte und unveränderte Software gebündelt wird

zS zentrale Server

Literaturverzeichnis

[AAD 04a] ABRAN, ALAIN, ALAIN APRIL und REINER R. DUMKE: *Assessment of Software Maintenance Capability: A Model and its Architecture*. In: *IASTED conference on Software Engineering*, Innsbruck, Austria, 2004. .

[AAD 04b] ABRAN, ALAIN, ALAIN APRIL und REINER R. DUMKE: SM^{CMM} *Model to Evaluate and Improve the Quality of Software Maintenance Process: Overview of the model.* In: *SPICE 2004 Conference on Process Assessment and Improvement, Critical Software SA*, Seiten 19–32, Lisbon (Portugal), 2004. The Spice User Group.

[AADHH 05] ABRAN, ALAIN, ALAIN APRIL, REINER R. DUMKE und JANE HUFFMAN HAYES: *Software Maintenance Maturity Model (SM^{mm}: the Software maintenance process model*. Journal of Software Maintenance and Evolution: Research and Practice, 17:197–223, 2005.

[AbZi 96] ABRAN, ALAIN und MOHAMED ZITOUNI: *A Model To Evaluate And Improve The Quality Of Software Maintenance Process*. In: *6th International Conference on Software Quality*, Ottawa: ASQ-Software Division, 1996. .

[Alc05] *Getting In-Control - Combining CobiT and ITIL for IT Governance and Process Excellence*. Alcyone Consulting, 2005. Whitepaper.

[Ande 00] ANDERSSON, JESPER: *A Deployment System for Pervasive Computing*. In: *ICSM '00: Proceedings of the International Conference on Software Maintenance (ICSM'00)*, Washington, DC, USA, 2000. IEEE Computer Society.

[asg] *Infrastructure Management - ASG-IMPACT Consolidated Service Desk Solution*. www.asg.com. Zugriff am 20.2.2006.

[BaFin] *Bundesanstalt für Finanzdienstleistungsaufsicht*. www.bafin.de. Zugriff am 19.1.2006.

[Ball 05] BALLINTIJN, GERCO: *A Case Study of the Release Management of a Health-care Information System*. In: *IEEE International Conference on Software Maintenance (ICSM2005)*, September 2005.

[Balz 96] BALZERT, HELMUT: *Lehrbuch der Software-Technik-Software-Entwicklung*. Spektrum Akademischer Verlag, 1996.

[BGN 06] BRENNER, MICHAEL, MARKUS GARSCHHAMMER und FRIEDERIKE NICKL: *Requirements Engineering und IT Service Management Ansatzpunkte einer integrierten Sichtweise*. In: *GIEditionLecture Notes in Informatics (LNI), 2006(P82), Modellierung 2006, 5165*. Gesellschaft für Informatik (GI), März 2006.

[BJv 04] BALLINTIJN, GERCO, REMY JANSEN und TIJS VAN DER STORM: *Deliver: A Fresh Look at Software Release and Deployment*. ERCIM News, 58, July 2004.

[BöKr 05] BÖHMANN, TILO und HELMUT KRCMAR: *Grundlagen und Entwicklungstrends im IT-Servicemanagement*. HMD - Praxis der Wirtschaftsinformatik, 237:7–21, 2005.

[BS15000] *The ISO 20000 (BS15000 / BS 15000) ITSM Standard*. http://www.bs15000.org.uk/. Zugriff am 19.12.2005.

[BSI05] *ITIL und Informationssicherheit - Möglichkeiten und Chancen des Zusammenwirkens von IT-Sicherheit und IT-Service Management*. Bundesamt für Sicherheit in der Informationstechnik, 2005.

[BuBu 05] BULEJ, LOBOMIR und TOMAS BURES: *Using Connectors for Deployment of Heterogeneous Applications in the Context of OMG D&C Specification*. In: *INTEROP-ESA 2005 conference, Geneva, Sitzerland*, 2005.

[CAPD 03] CRNKOVIC, IVICA, ULF ASKLUND und ANNITA PERSSON DAHLQVIST: *Implementing and Integrating Product Data Management and Software Configuration Management*. Artech House Publishers, 2003.

[Carz 97] CARZANIGA, ANTONIO: *A Characterization of the Software Deployment Process and a Survey of related Technologies*. Technischer Bericht, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 1997.

[CFH+ 98] CARZANIGA, ANTONIO, ALFONSO FUGGETTA, RICHARD S. HALL, DENNIS HEIMBIGNER, ANDRÉ VAN DER HOEK und ALEXANDER L. WOLF: *A Characterization Framework for Software Deployment Technologies*. Technischer Bericht CU-CS-857-98, University of Colorado, Department of Computer Science, April 1998.

[Cmu02] *Capability Maturity Model Integration* ($CMMI^{SM}$) *Version 1.1*. Technischer Bericht CMU/SEI-2002-TR-011, Carnegie Mellon Software Engineering Institute, March 2002, http://www.sei.cmu.edu/publications/documents/02.reports/02tr011.html .

[Cob00] *COBIT 3rd Edition - Management Guidelines*. Released by the COBIT Steering Committee and the IT Governance Institute, Juli 2000.

[Cob01] *COBIT* 3^{rd} *Edition - Der international anerkannte Standard für IT-Governance*. ISACA Switzerland Chapter, Zürich, September 2001, http://www.isaca.ch/files/CobitBroschuere.pdf .

[Cob05] *COBIT 3rd Edition Framework*. IT Governance Institute, July 2000, www.isaca.org/cobitinput.htm .

[Comp 04] COMPUTER ASSOCIATES INTERNATIONAL: *Allfusion Endevor Change Manager Is Latest CA Solution To Achieve ITIL Certification*. http://www3.ca.com/Press/PressRelease.aspx?CID=60022, June 2004.

[Crn99] CRNKOVIC, IVICA und MAGNUS LARSSON: *Managing Standard Components in Large Software Systems*. In: *Second International Workshop on Component-Based Software Engineering*, Los Angeles, May 1999. .

[Cros 96] CROSBY, PHILIP B.: *Qualität ist und bleibt frei: die Ratschläge des Qualitäspapstes für das 21. Jahrhundert*. Ueberreuter, 1996.

[CuGi 89] CURTH, MICHAEL und MARTIN L. GIEBEL: *Management der Software-Wartung*. B. G. Teubner Stuttgart, 1989.

[fQFI 03] QUALITY FACHGRUPPE INFORMATIK, SAQ SWISS ASSOCIATION FOR (Herausgeber): *MITO - ein Reifegradmodell für den Informatik Betrieb*. SAQ Swiss Association for Quality Fachgruppe Informatik, 2003.

[FrLi 04] FRÜHAUF, KAROL und GERALD LINHOFER: *Konfigurationsmanagement - das Bindeglied der IT-Prozesse*. HMD - Praxis der Wirtschaftsinformatik, 237:114 – 127, 2004.

[GiFr 05] GILLIS, MARY und MIKE FREEDLAND: *Distribution Completes the Software CM Process*. Technischer Bericht, Emperex Corporation, 2005, www.emperex.com/qunetix/distributioncompletes.pdf . Zugriff am 14.10.2005.

[Gree 00] GREEFHORST, DANNY: *Separating Concerns in Software Logistics*. In: *Advanced Separation of Concerns Workshop, OOPSLA 2000*, Minneapolis, October 2000. .

[Groh 03] GROHMANN, HELMUT H.: *Prinzipien der IT-Governance*. HMD - Praxis der Wirtschaftsinformatik, 232, 2003.

[Hal99] HALL, RICHARD S., DENNIS HEIMBIGNER und ALEXANDER L. WOLF: *A Cooperative Approach to Support Software Deployment Using the Software Dock*. In: *International Conference on Software Engineering*, Seiten 174–183, 1999.

[Havo 04] HAERING, ANTOINETTE und RETO VON ARB: *IT Performance Management - ein Ansatz zur Steuerung des IT-Bereichs*. HMD - Praxis der Wirtschaftsinformatik, 237, 2004.

[HHB 05] HOCHSTEIN, AXEL, ANDREAS HUNZIKER und WALTER BRENNER: *Serviceorientiertes IT-Management nach ITIL: Möglichkeiten und Grenzen*. HMD - Praxis der Wirtschaftsinformatik, 237, 2005.

[Hnet 04] HNETYNKA, PETR: *Making Deployment Of Distributed Component-Based Software Unified*. In: *Proceedings of CSSE 2004 (part of ASE 2004)*, Seiten 157–161, Linz, Austria, Sep 2004. Austrian Computer Society.

[HoCo 02] HOEKSTRA, ANGELI und NICOLETTE CONRADIE: *CobiT, ITIL and ISO17799 - How to use them in conjunction*. http://www.itsmf.org.za/Presentations/Presentations2002/CobiT0Bs7799.pdf, 2002.

[HoHu 03] HOCHSTEIN, AXEL und ANDREAS HUNZIKER: *Serviceorientierte Referenzmodelle des IT-Managements*. HMD - Praxis der Wirtschaftsinformatik, 232:45 – 56, 2003.

[HSW 04] HAFNER, MARTIN, JOACHIM SCHELP und ROBERT WINTER: *Architekturmanagement als Basis effizienter und effektiver Produktion von IT-Services*. HMD - Praxis der Wirtschaftsinformatik, 237:54 – 66, 2004.

[HTB 05] HOCHSTEIN, AXEL, GERRIT TAMM und WALTER BRENNER: *Service-Oriented IT Management: Benefit, Cost and Success Factors*. In: *Proceedings of the Thirteenth European Conference on Information Systems*, Regensburg, 2005. .

[IBM 06] IBM: *Tivoli Software Distribution white paper*. www.ibm.com, 2006. Zugriff am 28.02.2006.

[IEEE610] *IEEE Standard Glossary of Software Engineering Terminology - IEEE 610.12-1990*. Institute of Electrical and Electronics Engineers, 1990.

[ITGI-OGC] *Aligning COBIT, ITIL and ISO 17799 for Business Benefit - A Management Briefing from ITGI and OGC*. http://www.isaca.org, 2005.

[ITIL03] http://www.itil.org. Zugriff am 10.01.2006.

[ITIL04] http://www.itilsurvival.com/itsmf.html. Zugriff am 10.10.2006.

[ITIL05] http://www.itil.co.uk/. Zugriff am 10.10.2006.

[Jan05] JANSEN, SLINGER und SJAAK BRINKKEMPER: *Definition and Validation of the Key Process Areas of Release, Delivery and Deployment of Product Software Vendors: turning the ugly duckling into a swan*. Technischer Bericht UU-CS-2005-041, Institute of Information and Computing Sciences, Utrecht University, 2005.

[Jans 05] JANSEN, SLINGER: *Alleviating the Release and Deployment Effort of Product Software by Explicitly Managing Component Knowledge*. In: *Workshop an Development of Product Software, San Diego, CA, USA*, June 2005.

[JBBvN 05] JANSEN, SLINGER, SJAAK BRINKKEMPER, GERCO BALLINTIJN und ARCO VAN NIEUWLAND: *Integrated Development and Maintenance of Software Products to Support Efficient Updating of Customer Configurations: A Case Study in Mass Market ERP Software*. In: *IEEE International Conference on Software Maintenance (ICSM2005)*, Budapest, Hungary, September 25-30 2005. .

[KHP 04] KEMPER, HANS-GEORG, EVANGELOS HADJICHARALAMBOUS und JÖRG PASCHKE: *IT-Servicemanagement in deutschen Unternehmen - Ergebnisse einer empirischen Studie zur ITIL*. HMD - Praxis der Wirtschaftsinformatik, 237:22–31, 2004.

[KM 04] KAJKO-MATTSSON, MIRA: *Diary of the SERVIAM Evolution and Maintenance Sub-project*. Technischer Bericht SERV-FORV-1, Department of Computer and Systems Sciences Stockholm University and Royal Institute of Technology, Octobre 2004.

[KMMe 05] KAJKO-MATTSSON, M und P MEYER: *Evaluating the Acceptor Side of EM3: Release Management at SAS*. In: *4th International Symposium on Empirical Software Engineering*, Noosa Heads, Australia, November 2005. .

[KMTe 05] KAJKO-MATTSSON, MIRA und MICHAL TEPCZYNSKI: *SERVIAM Maintenance Framework*. Technischer Bericht, Department of Computer and Systems Sciences Stockholm University & Royal Institute of Technology, Sweden, September 2005.

[Kö 05] KÖHLER, PETER T.: *ITIL*. Springer-Verlag, 2005.

[Kruc 99] KRUCHTEN, PHILIPPE: *Der Rational Unified Process - Eine Einführung*. Addison-Wesley, 1999.

[Macf 02] MACFARLANE, IVOR: *IT Service Management - Ein Begleitband zur IT INFRASTRUCTURE LIBRARY*. itSMF, 2002.

[Mat05] MATTHIESSEN, BJÖRN, WOLFGANG ANGER und DAVID KIRSCHENEDER. http://www5.in.tum.de/lehre/seminare/semsoft/unterlagen_02/vanc/website/finanz.html. Zugriff am 15.10.2005.

[McNi 04] MCNISH, PATSY: *SMART and ITIL*. http://www.risglobal.com, December 2004.

[MD 93] MC DERMID, JOHN: *Software Engineer's Reference Book*. Butterworth Heinemann, 1993.

[Merc 06] MERCURY INTERACTIVE CORPORATION: *Mercury TestDirector*. www.mercury.com/us/products/quality-center/testdirector/, 2006.

[Micr] MICROSOFT CORPORATION: *Informationen zu Microsoft Operations Framework (MOF)*. http://www.microsoft.com/technet/itsolutions/cits/mo/mof/default.mspx. Zugriff am 17.10.2005.

[Mof05] *Microsoft Operations Framework Changing Quadrant*. Foliensatz zu Course 1787B, http://www.microsoft.com/technet/itsolutions/cits/mo/mof/mofeo.mspx . Zugriff am 01.11.2005.

[MSFa] MICROSOFT CORPORATION: *MSF Process Model v. 3.1*, June 2002.

[MSFb] MICROSOFT CORPORATION: *MSF Team Model v. 3.1*, June 2002.

[MSF] MICROSOFT CORPORATION: *Microsoft Solutions Framework version 3.0 Overview*. http://www.microsoft.com/msf, June 2003.

[MSI] MICROSOFT CORPORATION: *Windows Installer*. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/windows_installer_start_-page.asp, Dezember 2005.

[NCv 05] NIESSINK, FRANK, VIKTOR CLERC und HANS VAN VLIET: *The IT Service Capability Maturity Model*. http://www.itservicecmm.org/, January 2005. IT Service CMM Version 1.0, Release Candidate 1.

[Niva 98] NIESSINK, FRANK und HANS VAN VLIET: *Towards Mature IT Services*. Software Process - Improvement and Practice, 4:55–71, 1998.

[Niva 00] NIESSINK, FRANK und HANS VAN VLIET: *Software maintenance from a service perspective*. Journal of Software Maintenance: Research and Practice, 12:103–120, 2000.

[Obje 03] OBJECT MANAGEMENT GROUP: *Deployment and Configuration of Component-based Distributed Applications Specification*. Technischer Bericht, Object Management Group, June 2003, www.omg.org/docs/ptc/03-07-02.pdf .

[OGC04] OGC (Herausgeber): *Service Support - The Stationary Office*. CCTA, 2004.

[Olbr 04] OLBRICH, ALFRED: *ITIL kompakt und verständlich*. Vieweg, 2004.

[Pigo] PIGOSKI, THOMAS M.: *SWEBOK Knowledge Area Description for Software Evolution and Maintenance (version 0.6).*

[PuQu 03] PULTORAK, DAVE und PETE QUAGLIARIELLO: *Das MOF-Taschenbuch.* Van Haren Publishing, 2003.

[PVCS05] *PVCS Version Control Software*. http://www.serena.coml. Zugriff am 20.1.2005.

[Rama 04] RAMAKRISHNAN, MURALI: *Software release management*. Bell Labs Technical Journal, 9(1):205–210, 2004.

[RUP] RATIONAL SOFTWARE CORPORATION: *Rational Unified Process - Best Practices for Software Development Teams.*

[Schn 06] SCHNÄGELBERGER, SVEN: *Jeder kocht eigenes Süppchen - Diese Referenzmodelle kommen in den Unternehmen zum Einsatz.* Computer Zeitung, (10), März 2006. Quelle: Kompetenzzentrum für GPM.

[Segu 06] SEGUE SOFTWARE INC.: *SilkTest.* www.segue.com, 2006. Zugriff am 27.02.2006.

[SFS 00] SCHEUING, ARNOLD Q., KAROL FRÜHAUF und WOLFGANG SCHWARZ: *Maturity Model for IT Operations (MITO).* In: 2^{nd} *World Congress on Software Quality*, Seiten 25–29, Yokohama, September 2000. .

[Somm 04] SOMMER, JOCHEN: *IT-Servicemanagement mit ITIL und MOF*. mitp, 2004.

[SOSA 99] STARK, GEORG, PAUL OMAN, ALAN SKILLICORN und CAPT. RYAN AMEELE: *An Examination of the Effects of Requirements Changes on Software Maintenance Releases.* Journal of Software Maintenance, 11:293–309, 1999.

[Spil 03] SPILLNER, ANDREAS: *Das W-Modell - Vorteile der agilen Prozesse in einem konservativen Umfeld nutzen.* In: *Vortrag bei der GI-Regionalgruppe Bremen/Oldenburg*, Bremen, Mai 2003. .

[Stei 04] STEINWEG, CARL: *Management der Software-Entwicklung - Projektkompass für die Erstellung von leistungsfähigen IT-Systemen.* Vieweg, 2004.

[Stie 99] STIENEN, HANS: *Nach CMM und BOOTSTRAP: SPiCE Die neue Norm für Prozessbewertungen*. Informatik, 6:16–22, 1999. http://www.svifsi.ch/revue/pages/issues/n996/a996Stienen.pdf.

[SWE-BOK] HILBURN, THOMAS B., IRAJ HIRMANPOUR, SOHEIL KHAJENOORI, RICHARD TURNER und ABIR QASEM: *A Software Engineering Body of Knowledge Version 1.0.* Technischer Bericht CMU/SEI-99-TR-004, Carnegie Mellon University, Software Engineering Institute (SEI), Pittsburgh, PA, April 1999.

[Teic 06] TEICHNER, PETRA: *Server Based Computing boomt - und niemand spricht darüber*. http://www.silicon.de, 2006. Zugriff am 05.03.2006.

[Thom 05] THOMPSON, PETER: *Responsible Technology Governance*. www.cipstoronto.ca/activities/2005/4, 2005. Präsentation beim March Breakfast Meeting am 10.März 2005.

[TTS 04] THANHEISER, STEFAN, FREDERIC TOUSSAINT und HARTMUT SCHMECK: *Die „Softwaretankstelle“mit integriertem Verleihsystem für mobile Geräte*. In: *DeLFI 2004: Die e-Learning Fachtagung Informatik, Tagung der Fachgruppe e-Learning der Gesellschaft für Informatik e.V. (GI)*, Seiten 67–78, Paderborn, September 2004. .

[V-Modell-a] *Das V-Modell - Vorgehensmodell zur Planung und Durchführung von IT-Vorhaben*. www.v-modell.iabg.de. Zugriff am 16.11.2005.

[V-Modell-b] *Vorgehensmodell (V-Modell).* http://www.iese.fhg.de/VModell/Intro/vm.introMain.htmlrl. Zugriff am 16.11.2005.

[V-Modell-c] *V-Modell*. http://www.iese.fhg.de/VModell/. Zugriff am 20.10.2005.

[V-Modell XT] *Das V-Modell XT, Version 1.1.0.* http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.1/Dokumentation/html/. Zugriff am 16.11.2005.

[V-Modell] DRÖSCHEL, WOLFGANG und MANUELA WIEMERS: *Das V-Modell 97.* Oldenburg Verlag München Wien, 2000.

[vHC+ 98] VAN DER HOEK, ANDRÉ, RICHARD S. HALL, ANTONIO CARZANIGA, DENNIS HEIMBIGNER und ALEXANDER L. WOLF: *Software Deployment: Extending Configuration Managemnt into the Field.* CrossTalk The Journal of Defense Software Engineering, 11:9–13, Feb. 1998.

[ViG 04] VICTOR, FRANK und HOLGER GÜNTHER: *Optimiertes IT-Management mit ITIL.* Vieweg, 2004.

[vKP 02] VON BON, JAN, GEORGES KEMMERLING und DICK PONDMAN: *IT Service Management, an introduction.* Van Haren Publishing, May 2002.

[WSK 05] WIRSING, MARTIN, HARALD STÖRRLE und NORA KOCH: *Vorlesung Methoden des Software Engineering - Block E (SW-Prozess & Projektmanagement): Vorgehensmodelle.* http://www.pst.informatik.uni-muenchen.de/lehre/WS0304/mse/material.shtml, Januar 2005. Zugriff am 17.1.2006.

[ZaBr 04] ZARNEKOW, RÜDIGER und WALTER BRENNER: *Informationsmanagement - Konzepte und Strategie für die Praxis.* dpunkt, 2004.

[ZBP 05] ZARNEKOW, RÜDIGER, WALTER BRENNER und UWE PILGRAM: *Integriertes Informationsmanagement - Strategien und Lösungen für das Management von IT-Dienstleistungen.* Springer, 2005.

[Zell] ZELLER, ANDREAS: *Software Test.* www.st.cs.uni-sb.de/edu/einst/12-testen.pdf. Zugriff am 21.03.2006.

[ZeMa 03] ZEYDA, FRANZ und PETER MANDL: *Change-Management bei der Hausbank München.* HMD - Praxis der Wirtschaftsinformatik, 234:87 – 97, 2003.

Index

AQUA, 113
Customer Configuration Updating, 25

Abnahme Qualität - Projekt zur Einführung eines Release-Managements im Host-Bereich, 113
Abnahmeumgebungsstatus-Ampel - Aktueller Status der Abnahmeumgebungen wird mit Ampelfarben dargestellt, 95
AI, 113
ALADIN, 62
Anwendungsintegrator, 113
Anwendungssystem, 69
Anwendungssystemverantwortlicher, 77
Application Development Information Navigator , 62
Application Support and Maintenance, 39
ASM, 39
ASU-Ampel, 95
AWS, 69
AWSV, 77

BaFin, 168
BSI, 62
Bundesamt für Sicherheit in der Informatik, 62
Bundesanstalt für Finanzdienstleistungsaufsicht, 168

C/S, 68
Capability Maturity Model Integration, 23
CCB, 13
CCU, 25
Change Control Board, 13
CI, 43
Client/Server, 68
CMDB, 43
CMMI, 23
CobIT, 50
Common Object Request Broker Architecture, 56
Configuration Item, 43
Configuration Management Database, 43
Control Objectives for Information and Related Technology, 50
CORBA, 56
critical success factor, 50
CSF, 50

Definitive Hardware Store, 46
Definitive Software Library, 46
DHS, 46
Dokumentationstool im Intranet der HVBInfo basierend auf HiScout, 98
DSL, 46

EFQM, 51
EI, 113
Einführungsmanager, 77
Einführungsverantwortlicher, 77
Einsatzverantwortlicher, 113
EM, 77
EM^3, 40
ENDEVOR/MVS, 115
Entwicklungsumgebung, 74
Environment for Development and Operation, 115
EU, 74
European Foundation for Quality Management, 51
EV, 77
Evolution and Maintenance Maturity Model , 40

Fremdsoftware-Teilsystem, 70
FSW, 70

gemanagte Entwicklungsumgebung im Host-Bereich, 74
gemanagte Prelife-Umgebung - produktionsähnliche Abnahmeumgebung, 74
GET, 74
GPL, 74

HVB Corporates & Markets, 72
HVB Informations-Verarbeitungs-GmbH, 66
HVBInfo, 66
HyperDB, 88

I, 110
Information Systems Audit and Control Association, 50
Initial Operational Capability, 22
Installation, 110
Integrationstestphase, 73
Integrationstestumgebung, 71, 74
Intelligent Software Knowledge Base, 26
IOC, 22
ISACA, 50
ISKB, 26
IT Infrastructure Library, 43
IT Service Management Forum, 62

IT-Phase, 73
IT-Servicemanagement, 7
IT-U, 74
IT-Umgebung, 71
ITIL, 43
ITSM, 7
itSMF, 62

Key Process Areas, 53
Key-Performance-Indikator, 50
KM, 15
Komponentenverwaltungsystem, 70
Konfigurationsmanagement, 15
kontinuierlicher Verbesserungsprozess, 98
KPA, 53
KPI, 50
Kundenset für Vertrieb und Zentrale, 83
KVP, 98
KVS, 70
KVZ, 83

LCA, 22
LCO, 22
Life-Cycle Architecture, 22
Life-Cycle Objective, 22

Maturity Model for IT Operations, 51
MCB, 72
Microsoft Operations Framework, 37
Microsoft Solutions Framework, 18
Microsoft Windows Installer, 86
MITO, 51
MOF, 37
MSF, 18
MSI, 86

NDM, 110
Netview Datasupply Manager, 110

Object Management Group, 56
OLA, 49
OMG, 56
Operational Level Agreement, 49

PÜ, 75, 113
Paketmanager, 110
PARIS, 86
PDM, 32
PIR, 157
PKG, 110
PM, 15
PMS, 11
Post Implementation Review, 157
PR, 22
PREP, 119
PRK, 104
Problem Monitoring System, 11
PROD, 74
Product and Release Integration Suite - Datenbank zur Konfigurationsunterstützung, 86
Product Data Management, 32
Production Readiness Kriterien - Auflistung erwarteter Artefakte als Ergebnis des Softwareentwicklungsprozesses, 104
Produkt-Release, 22
Produktionsübergabe, 113
Produktionsübergabeumgebung, d.h. Pilotumgebung für Installationstests, 75
Produktionseinsatz und -planung, 119
Produktivumgebung, 74
Produktverantwortlicher, 77
Projektmanagement, 15
PV, 77

QS, 13
QS-Phase, 73
QS-U, 74
QS-Umgebung, 71
Qualitätssicherung, 13
Qualitätssicherungsphase, 73
Qualitätssicherungsumgebung, 71, 74

RAB, 41
Rational Unified Process, 21
RE, 60
Red Hat Package Manager zum (De-)Installieren und Aktualisieren von Software und Zusammenstellen von Softwarepaketen, 30
Release Advisory Board, 41
Release Quality established - Projekt im dezentralen RM mit Hauptfokus Einzelproduktversorgung und Virtualisierung von IT-U, 86
Release Readiness Review, 38
Release-Durchlauf, 71
Release-Management, 7
Release-Management-Gremium, 150
Release-Manager Client/Server, 77
Release-Manager im Host-Bereich, 113
ReQuest, 86
Request for Change, 11
Requirements Engineering, 60
RfC, 11
RM, 7
RM-C/S, 77
RM-Host, 113
RMG, 150
RPM, 30
RRR, 38
RUP, 21

SAQ, 51
SB-Services, 67

SCM, 32
SCOR, 63
SE, 15
SEI, 23
Selbstbedienungs-Services, 67
Serverice Level Agreement, 43
SERVIAM, 40
Service Level Agreement, 49
Service Management Function, 38
Serviceorientiertes IT-Management, 63
SI, 77, 83
SIM-Portal, 82, 98
SLA, 43, 49
SM^{CMM}, 53
SMART, 39
SMF, 38
Software Configuration Management, 32
Software Engineering Body of Knowledge, 35
Software Engineering Institute der Carnegie Mellon University, 23
Software Maintenance Capability Maturity Model, 53
Software Problem Reports, 11
Software Process Improvement Capability Determination, 62
Software Release-Management, 29
Softwareengineering, 7
Softwareversorger, 77
SOITM, 63
Solution Management and Request Tracking, 39
SPICE, 62
SPR, 11
SRM, 29
Supply-Chain Operation Reference, 63
SWE, 7
SWE-BOK, 35
Swiss Association for Quality, 51
SWV, 77
System Information Management-Portal; Instrument basierend auf *HiScout* zum dokumentieren und veröffentlichen von Dokumenten, 82
Systemerstellung, 15
Systemintegrationsumgebung, 83
Systemintegrator, 77

Theke, 68

UC, 49
UM, 150
Umgebungsmanager, 150
Umgebungsverantwortlicher, 77
UML, 21
Underpinning Contract, 49
Unified Modelling Language, 21
UV, 77

V, 110
Versorgung bzw. Verteilung, 110

zentrale Server, 68
zS, 68