Institut für Rechnergestützte Ingenieursysteme

Fakultät Informatik, Elektrotechnik und Informationstechnik

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Diplomarbeit Nr. 3423

Entwicklung einer Vorgehensweise zur Testautomatisierung im Rahmen des PDM am Beispiel der HELLA KGaA Hueck & Co.

Huimei Liu

Studiengang:	INFORMATIK
Prüfer:	Univ.-Prof. Hon-Prof. Dr. Dieter Roller
Betreuer:	Dipl.-kfm., MBA Pascal Borst,
	M.Sc. Leila Zehtaban
begonnen am:	27.11.2012
beendet am:	28.05.2013
CR-Klassifikation:	D2.1, D2.2, D2.5, D2.9, I.2.2, J.7

Abstract

Software testing is one of the most important parts of the product development cycle. The widely using of Information Technology (IT) operations in product development has made the industries highly interested in the concept of testing & quality assurance of the system. In this regard, automate software testing assists the designer to find errors at an early stage of design to minimize the failure risks and to ensure the correct running of the complex and diverse processes. In addition, automatic Software testing assures to save time of cost.

In the current thesis, an automatic test process is designed. The approach is based on test-concepts provided by the available extended Computer Aided Test Tool (eCATT) designed by one of the most well-known companies in this field, SAP AG. The automated test cases that have been implemented in eCATT are investigated and also a complete test process-approach to extend the test automation will be developed. The main work is to implement an efficient test process to automate the prioritized test cases. The required test data will be structured and saved in the test process. This test data will be used afterwards for a decision making model to confirm the test automation of a test is valued.

To develop the procedure for test automation, waterfall model is implemented. It is based on eCATT in which the test process is directly applied to unstructured test data. Unstructured term refers to the test cases from test-blocks in Excel sheet (test data) are not specified by the test. In the current test processes, the test blocks are first analyzed and prioritized according to test requirements, and then the analyzed test cases and the sequence of the individual test cases are obtained in the eCATT scripts. The test script corresponds to repeatedly test cases which are created for automated testing. To construct a complete workflow in the testing process, a test model is presented with specified definition. Within this test model, several limited test workflows will be developed, with the appropriate test scripts which are designed according to the requested order in a workflow. The order in which the individual test scripts to be executed is determined by a prioritization method according to the Kano model. With the help of these automations, the automated test processes are clearly presented, and the reusable test scripts are accessed efficiently and frequently. The test processes are maintained temporarily although they can be expanded and developed. The eCATT test scripts allow repeated execution of each test. In addition, to minimize the implementation costs, an extended test specification was created for new tests by a uniform scheme. Furthermore, the test concept depicts a decision method based on mathematical analysis to confirm if the test automation has preference to the manual test. The results of each execution are recorded in a log archive.

Keywords

SAP - PDM - Software Testing - Test automation - Test tool - eCATT - Capture and Replay - Test Specification - test execution - Test Procedure

Kurzfassung

Softwaretests sind eine der wichtigsten Teile des Softwareentwicklungszyklus. Besonderes für Unternehmenssoftware wie SAP, bei der Firmen-Spezifische Anpassungen unerlässlich sind, spielt das Thema eine wichtige Rolle. Sie sollen Unternehmen dabei unterstützen frühzeitig Fehler in diesen komplexen und vielfältigen Funktionalitäten zu finden, damit Risiken zu minimieren und die reibungslosen Funktionen der Software weltweit zu gewährleisten. Tests und Testautomatisierung stehen immer im Dienst einer übergreifenden Qualitätssicherung. Durch den Einsatz von automatisierten Softwaretests verringert sich der zeitliche Aufwand erheblich und Kosten werden gesenkt.

In der vorliegenden Diplomarbeit werden die automatisierten Testprozesse erstellt. Die Vorgehensweise basiert auf dem Test-Konzept, dass dem extended Computer Aided Test Tool (eCATT) von SAP zu Grunde liegt. Den Kern der Arbeit bildet die Implementierung des effizienten Testprozesses in dem zuvor priorisierte Testfälle automatisiert ausgeführt werden. Die nötigen Testdaten werden dazu im Rahmen des Testprozess strukturiert und anhand eines Endscheidungsmodells wird ein Hinweis gegeben ob eine Automatisierung des jeweiligen Tests lohnenswert ist.

Zur Entwicklung einer Vorgehensweise zur Automatisierung wird in dieser Arbeit ein Wasserfall-Modell vorgestellt und anhand des ausgewählten Testwerkzeuges eCATT implementiert, bei dem der Testprozess direkt auf unstrukturierte Testdaten angewendet wird. Unstrukturiert bedeutet hier, dass die Testfälle aus einer Testblöcke-Excel Tabelle (Testdaten) nicht nach Testwerkzeug spezifiziert sind und keine direkten Nachbarschaftsinformationen zwischen den einzelnen Testblöcken existieren. In dem vorgestellten Testprozess werden zunächst die Testblöcke nach Test-Anforderung analysiert und priorisiert. Die analysierten Testfälle sowie der Ablauf aus den einzelnen Testfällen werden in der eCATT-Skriptsprache gewonnen. Das Testskript entspricht wiederholt getesteten Testfällen, und wird zum automatischen Testen erstellt. Um einen vollständigen Workflow im Testprozess zu konstruieren, wird ein Testautomaten-Modell mit spezifizierter Definition vorgestellt. Innerhalb dieses Modells werden mehrere eingeschränkte Test-Workflows entwickelt, mit der die entsprechenden Testskripts nach angeforderter Reihenfolge in einem Workflow entworfen werden. Die Reihenfolge in der die einzelnen Testskripts ausgeführt werden sollen, wird durch eine Priorisierungsmethode nach dem Kano-Modell festgelegt. Mit Hilfe dieser Automaten werden automatisierbare Testprozesse übersichtlich dargestellt, damit die wiederverwendbaren Testskripts effizient aufgerufen werden können und die häufig wiederholten Testskripts zeitig gepflegt, erweitert und entwickelt werden können. Zur Ansteuerung des Testskripts benötigt eCATT zahlreiche Treiber, um die Schnittstelle zu dem zu testenden System jederzeit sicher zu gewährleisten. Die eCATT Testskripte ermöglichen eine wiederholte Durchführung der einzelnen Tests ohne Mehraufwand. Zusätzlich wurde eine erweiterte Testspezifikation erstellt um den Implementierungsaufwand für neue Tests durch ein einheitliches Schema zu minimieren. Ferner bildet das Testkonzept eine Entscheidungsmethode ab, die durch

mathematische Analyse entscheidet ob sich ein Automatisierungstest zeitlich rentiert, oder dieser manuell getestet werden soll. Während ihrer Ausführung steuern eCATT-Testskripte die SAP-Anwendung genauso, wie sie von einem Benutzer bedient werden würde.

Die Ergebnisse jeder einzelnen Ausführung eines Testskripts werden in einem archivierbaren Protokoll festgehalten. Zum Schluss werden die Ergebnisse der Implementierung von diesem Testkonzept in mathematischer Weise analysiert und ein Fazit gezogen. bzw. ein Ausblick auf die Möglichkeit zur Erweiterung des Testprozesses geworfen.

Schlagwörter

SAP - PDM - Softwaretests - Testautomatisierung - Testwerkzeug - eCATT - Capture and Replay - Testspezifikation - Testausführung - Testvorgehensweise

Inhaltsverzeichnis

Abstract ... **3**

Kurzfassung ... **5**

Inhaltsverzeichnis ... **7**

Abbildungsverzeichnis ... **9**

Tabellenverzeichnis ... **11**

Abkürzungsverzeichnis ... **13**

1 Einführung ... **15**

1.1 Hintergrund der Forschung ... 15

1.2 Problemstellung ... 15

1.3 Ziel und Zweck der Diplomarbeit ... 16

1.4 Vorteile der Forschungsarbeit ... 16

1.5 Gliederung ... 17

2 Im Rahmen des PDM am Beispiel HELLA ... **19**

2.1 Produkt Daten Management Definition ... 19

2.2 Produkt Daten Management System bei HELLA ... 20

3 Einführung des Software-Testens ... **25**

3.1 die Qualitätskriterien von Software ... 25

3.2 das Testen im Allgemein ... 26

3.2.1 Notwendigkeit von Tests ... 27

3.2.2 Teststufen in der SAP-Projektmethodik ... 27

3.2.3 Testprozess ... 30

3.3 Testautomatisierung ... 31

3.3.1 Was bedeutet Testautomatisierung ... 31

3.3.2 Werkzeug zur Testautomatisierung ... 33

3.3.3 Grundlagen von ausgewählten Testtool eCATT ... 39

4 Implementierung und Visualisierung ... **45**

4.1 Testsystem-Architektur ... 45

4.1.1 Geschäftsprozess der Angebotserstellung ... 45

4.1.2 Darstellung der Testsysteme und deren Technologie ... 47

4.2 Konzeptentwicklung zur Testautomatisierung ... 49

4.2.1 Testplanug und Testorganisation ... 51

4.2.2 Testaktivität ... 54

4.2.3 Methode zur Analyse der Testdaten 54
4.2.4 Spezifikation 61
4.2.5 Testdurchführung 63
4.2.6 Testauswertung 67
4.3 Vorgehensweise zur Realisierung der Testautomation 68
4.3.1 Voraussetzung für den Einsatz eCATT 70
4.3.2 Technische Voraussetzung 70
4.3.3 Modularisierung und Ausführen von Testskripten mit eCATT 73
4.3.4 Workflow am Beispiel der HELLA KGaAVorstellung 74
5 Ergebnisse , Zusammenfassung und zukünftige Arbeit 85
5.1 Ergebnisse 85
5.2 Zusammenfassung 86
5.3 Zukünftige Arbeit 88
Appendix A:
Testskript Y_TS_CV01N_MERKMALANZEIGEN für Merkmalsüberprüfung von Dokumentarten D10 89
Appendix B:
Testkonfiguration Y_TK_CN01_MERKMALANZEIGEN für Merkmalprüfung von diversen Dokumentarten-/Inhalt bei HELLA KGaA 91
Appendix C: Spezifikation von Testskript Y_TS_CV01N_MERKMALANZEIGEN 94
Appendix D: Dokumentart und Dokumentinhalt 97
Appendix E :nicht automatisierbare Tests am Beispiel 98
Appendix F: Reguläre Sprache und Automat 101
Literaturverzeichnis 103

Abbildungsverzeichnis

Abbildung 1: Produkt Daten Management 20
Abbildung 2: PDM System von SAP bei HELLA 21
Abbildung 3: Übersicht der produktbeschreibenden Daten im PDM System 22
Abbildung 4: Dokumentinfosatz 23
Abbildung 5: Weltweite Verfügbarkeit des PDM-Systems 23
Abbildung 6: Die Qualitätskriterien für Software nach ISO 9126 26
Abbildung 7: Bottom-Up-Modell der Integrationsstufen 28
Abbildung 8: Blackbox-Tests 29
Abbildung 9: Whitebox-Tests 30
Abbildung 10: Fundamentaler Testprozess 31
Abbildung 11: Beispiel Modellbasiert Testwerkzeug 34
Abbildung 12: Bestehender Lösungsweg 38
Abbildung 13: Integration des SAP Solution Mangers mit eCATT Funktionsumfang 38
Abbildung 14: Übersicht eCATT-Objekte 40
Abbildung 15: eCATT starten 41
Abbildung 16: eCATT-Testtool 41
Abbildung 17: Zusammenhänge zwischen der Aufgaben des Architekturentwurfs 45
Abbildung 18: Dokument anzeigen 47
Abbildung 19: Test Environment bei HELLA 48
Abbildung 20: Ablauf eines Testprozesses 50
Abbildung 21: Struktur des Testorganizers 51
Abbildung 22: Testdaten über Detail der Funktionalität in PDM System 52
Abbildung 23: Eine Testfall-Beschreibung über Merkmale im PDM-System 53
Abbildung 24: Ein Testfall über Merkmal Prüfung bei Status-Wechseln in PDM System 53
Abbildung 25: Zusammenspiel zwischen Anforderungen und Testfälle 58
Abbildung 26: Testfälle Analyse Ablauf 58
Abbildung 27: Testspezifikation Übersicht 62
Abbildung 28: Testfall-Beschreibung 63
Abbildung 29: eCATT Funktionsumfang 64
Abbildung 30: Testautomation-Modell 67
Abbildung 31: Testberichts bei der Statusübersicht 68
Abbildung 32: Vorgehensweise-Modell 69
Abbildung 33: Systemumgebung 71
Abbildung 34: RFC-Verbindungen in H42 71
Abbildung 35: Profilparametereigenschaften 72
Abbildung 36: Pflege eingener Benutzervorgaben 73
Abbildung 37: Sequenz von eCATT-Skripten 73
Abbildung 38: Automatisierungstest vs. Manuell Test 76
Abbildung 39: Die Allgemeine Daten eines Testskripts 77
Abbildung 40: Testskript-Screenshot 78

Abbildung 41: Spezifikation für Merkmal Prüfung von Dokumentart D10 und Dokumentinhalt D10.1 79
Abbildung 42: Testfall Beschreibung-DIS_Merkmal_001 81
Abbildung 43: Workflow für Dokumentart-/Inhalt abhängige merkmale prüfen 82
Abbildung 44: Protokoll von Dokumentart D10 in Testsystem H42 83
Abbildung 45: Testkonfiguration Y_TK_CN01_MERKMALANZEIGEN 85
Abbildung 46: Statische Analyse Überblick 87
Abbildung 47: Testautomatisierung 1.0 mit eCATT 87
Abbildung 48: Testautomatisierung 2.0 87

Tabellenverzeichnis

Tabelle 1: Manuell vs. Automatisiert 56
Tabelle 2: Kalkulation I auf Basis einer Entwicklungszeit von 240 Minuten 57
Tabelle 3: Kalkulation II auf Basis einer Entwicklungszeit von 180 Minuten 57
Tabelle 4: Kalkulation III auf Basis einer Entwicklungszeit von 120 Minuten 57
Tabelle 5: Testskripts benennen 60
Tabelle 6: Kriterien bewerten 60
Tabelle 7: Testskript i (Alternativen) bewerten 60
Tabelle 8: Priorität setzen 61
Tabelle 9: Vorschäge zur Namenskonvention 64
Tabelle 10: Vorteile und Nachteile von Einsatz Modellbasiert Testkonzept 88

Abkürzungsverzeichnis

ÄST	Änderungsstamm
BAPI	Business Application Programming Interface
CAO	Computer Aided/Assisted Office
CAP	Computer Aided Planning
DOK	Dokument
DIS	Dokumentinfosatz
eCATT	extended Computer Aided Test Tool
ECAD/EDA	Electronic Computer Aided Design/Electronic Design Automation
ISTQB	International Software Testing Qualifications Board
MAT	Materialstamm
MBT	Modellbasierter Test
MCAD	Mechanical Computer Aided Design
NFA	Nondeterministic Finite Automation
NWBC	Netweaver Business Client
PDM	Produkt Daten Management
PreBOM	frühe Konstruktionsstücklisten auf Basis von SAP bei HELLA
RFC	Remote Function Call Schnittstelle
SAP AG	Anbieter von Unternehmenssoftware
SolMan	SAP Solution Management
SUT	System under Test
TIS	Technische Informationsmanagement System

1 Einführung

1.1 Hintergrund der Forschung

Hochintegrierte Enterprise-Software verändert sich in ihrem Lebenszyklus kontinuierlich – veränderte gesetzliche Anforderungen, neue Technologien oder einfach nur Upgrades [Lang10]. Die dazukommenden Anpassungen müssen im System getestet werden. Tests und Testautomatisierung stehen immer im Dienst einer übergreifenden Qualitätssicherung: Sie sollen Unternehmen dabei unterstützen, Kosten und Risiken zu senken sowie Personal- und Zeitaufwand erheblich zu verringern, während die Komplexität der getesteten Systeme in der Regel steigt.

In der Regel kaufen viele Unternehmen das Basis-Paket der Software ein und passen dieses-gemäß den firmenspezifischen Anforderungen durch erweiterte Funktionen an. Doch auch die Test Funktionen müssen diese Anpassung abbilden und bei Bedarf müssen diese neue Test Funktionen erstellt werden. Fehlerhafte neue geeignete Funktionen können das Geschäft jeder Firma ins Stolpern bringen. Denn nur wer seine Anwendungen richtig testet, vermindert die Funktionen fehlerbehaftet firmenweit in Betrieb zu nehmen [Micr10]. D.h. fehlende Qualität kommt allerdings noch um ein Vielfaches teurer. Denn einer der häufigsten unsichtbaren Kostentreiber bei einer Anwendung sind die Mängel, die erst nach der Entwicklung im laufenden Betrieb auftreten. Solche Fehler führen häufig zu hohen Produktivitätsverlusten, steigern die Unzufriedenheit bei Kunden und Mitarbeitern und sorgen für einen erhöhten Support-Aufwand. Für das Test-und Qualitätsmanagement lassen sich die daraus resultierenden Kosten reduzieren, wenn Fehler frühzeitig entdeckt werden. Vermeidbare Effizienzverluste entstehen auch dadurch, dass teure Test-und Entwicklerressourcen oft für einfache, eigentlich automatisierbare Testaufgaben eingesetzt werden. Testautomatisierung ist ein mächtiges Werkzeug, um Tests wiederholbar zu machen und effizienter zu gestalten.

1.2 Problemstellung

Zu einer Testautomatisierung findet man Arbeitsweisen in Unternehmen, welche die Durchführung von Testaktivitäten unstrukturiert und unüberlegt umsetzen oder auch in den schlimmsten Fall in keiner Weise berücksichtigen. Bei solch einem Sachverhalt kann keine verlässliche Aussage über die Güte als auch über die Zuverlässigkeit getroffen werden. Ebenso sind Risiken nur sehr schwer zu identifizieren [HLT06]. Ob eine Modifikation, Optimierung oder Eigenentwicklung den Anforderungen entspricht bzw. diese erfüllt, kann nur durch ausreichende Softwaretests garantiert werden.

Vor diesem Hintergrund ist es absolut notwendig, individuelle Adaptionen präzise zu validieren. Eine ausführliche Validierung durch manuelle Testtätigkeiten erfordert allerdings einen hohen zeitlichen Arbeitsaufwand. Die Ressourcen hierfür sind in der Praxis jedoch nur selten vorhanden. Durch Testautomation kann dieser Aufwand stark minimiert werden, so dass dabei auch ein wirtschaftlich positiver Aspekt für das Unternehmen möglich ist [Meie12]. Bei der Testautomatisierung werden manuelle Testaktivitäten durch Computer automatisiert bzw. maschinell vorgenommen. Die Automatisierung ermöglicht somit eine verlässliche Aussage

über die Qualität als auch über die Zuverlässigkeit der aktuellen Software zu ermitteln. Gleichzeitig können Risiken bei Aktualisierung von Testprozess bis ins Detail identifiziert werden.

1.3 Ziel und Zweck der Diplomarbeit

Ziel dieser Diplomarbeit ist es, für Tester, eine organisierte und strukturierte Durchführung von automatisierten Testszenarien zu ermöglichen. Und ferner die ein Teil von manuellen durchgeführten Testaktivitäten durch Rechnerstützung zu automatisieren. Der Systemumgebung und deren Technologien zum Testautomatisieren sollte betrachtet und überprüft. Die Hauptaufgaben sind die vorhandenen Prozesse dieser Softwaretests zu analysieren, nach neuen innovativen Methoden zu recherchieren und diese optimierend in den Prozessablauf zu integrieren. Eine konkrete Vorgehensweise wird zu effizient Testprozess entwickelt. Durch den Einsatz von automatisierten Softwaretests verringert sich der zeitliche Aufwand. Es soll eine feste Aussage über die Software-Qualität getroffen werden können. Zugleich soll die Qualität der Testfälle gesichert werden. Da die Ergebnisse nur so gut sind wie Testfälle selber. Der Test der Anforderungen, also dessen, was der Kunde bzw. Auftraggeber bestellt hat, hängt von der Qualität der Testfälle ab.

1.4 Vorteile der Forschungsarbeit

Zum oben genannten Testziel steht hier ein kostenloses Testtool anhand Unternehmen zur Verfügung. Das gesamte Testkonzept wurde auf Basis der zur Verfügung stehenden Funktionen des von der SAP AG angebotenen Testtool eCATT erstellt. Derzeit stehen einige automatische Testprozesse zur Verfügung, die mit Berechtigungseinschränkung implementiert wurden. Diese existierenden Testprozesse basieren funktional auf dem Testwerkzeug eCATT. Diese sind teilweise durchführbar, aber die Abdeckungsgrade sind relativ eingeschränkt, sie sind eher dazu geeignet die eCATT-Fähigkeit herauszufinden.

Im Rahmen dieser Diplomarbeit werden einige neuen Methoden nach bestehendem Lösungsweg entwickelt, um die Testprozess zu optimieren bzw. neue Erweiterungen zu ermöglichen. Im Vergleich zu den existierenden Testprozessen, haben die Optimierungen und Erweiterungen folgende Vorteile:

a) Effizientere Lösungen über die mathematischen Verfahren von automatisierbaren Testfällen zu entscheiden.

b) Erweiterte Spezifikation wird einfacher und deutlicher ergänzt.

c) Das neue Vorgehensmodell beschränkt jede Phase mit stabilen Anforderungen und einer klaren Abschätzung von Kosten und Umfang.

d) Der gesamte Testprozess zur Automatisierung aus analysierten Testdaten wird mit einem übersichtlichen Automation-Modell beschleunigt, damit die Rekonstruktion für den Entwicklungsdienst erleichtert wird.

e) Außerdem besteht die Möglichkeit, den modulierten Testprozess in neue Test Software sowie von SAP angeboten Lösung: Solution Management weiter zu intergieren.

1.5 Gliederung

Für eine übersichtliche Beschreibung und Darstellung des Stoffes, ist der Inhalt dieser Diplomarbeit in fünf Kapitel wie folgt aufgegliedert.

Kapitel 1: Einführung

Kapitel 2: Im Rahmen des PDM am Beispiel HELLA

Kapitel 3: Einführung des Software-Testens

Kapitel 4: Implementierung und Visualisierung

Kapitel 5: Ergebnisse, Zusammenfassung und zukünftige Arbeit

Im Abschnitt Appendix werden die Testergebnisse Protocol bzw. weiterführende Informationen untereinander gegliedert und erläutert.

2 Im Rahmen des PDM am Beispiel HELLA

In diesem Abschnitt wird das Produkt Daten Management (PDM) für technische Daten- und Prozessmanagement erläutert. Im Unternehmen HELLA KGaA Hueck & Co. wird Produkt Daten Management System benutzt um die Produktdaten zu verwalten.

2.1 Produkt Daten Management Definition

Produkt Daten Management oder kurz PDM ist das Technische Informationsmanagement System (TIS) für Fertigungsunternehmen, Anlagenbauer und Engineering-Dienstleister. Es ist eine unternehmensweite Integrationsplattform für alle Daten erzeugenden und - nutzenden IT-Applikationen sowie Mechanical Computer Aided Design (MCAD), Computer Aided Planning (CAP), Electronic Computer Aided Design/Electronic Design Automation (ECAD/EDA), Computer Aided/Assisted Office (CAO), Viewer etc., die das Daten-, Prozess- und Projektmanagement organisiert und durch eine einheitliche Benutzeroberfläche den Datenzugriff für alle Anwender geregelt ermöglicht [JSIC13]. D.h. Die Voraussetzung dafür, diese Vorteile der digitalen Produktdaten nutzen zu können ist das Produkt Daten Management System als die beliebte Unternehmenssoftware, die nicht nur die Ablage organisiert, sondern auch die Freigabeprozesse und die Berechtigungen der unterschiedlichen internen und externen Benutzer verwaltet. Gleichzeitig ist sie günstiger und einfacher in der Bedienung.

Durch sein verknüpfendes Informationsmanagement ist PDM das effiziente Navigationssystem durch den technischen Produkt beschreibenden Datenbestand, sein Workflow-Management steuert gruppenorientierte Arbeitsprozesse, und sein Projektmanagement unterstützt die Abwicklung typisierter bzw. standardisierter Vorhaben/Aufgaben [Jose13]. Für den einzelnen Mitarbeiter ist das PDM ein multifunktionaler Arbeitsplatz zur Steigerung der persönlichen Produktivität, für das Unternehmen ist es ein strategisches IT-Werkzeug zur idealen Nutzung des Produktionsfaktors Information und zur Optimierung der Leistungsfaktoren Zeit, Kosten und Qualität.

Der Begriff Produktmanagement nach Joself Schöttner-Industrie-Consultant erläutert [Jose13]:“ Die neueste Generation von Web-basierten PDM-Systemen fungiert als elektronische Werkbank zur standort- und firmenübergreifenden Wertschöpfung in der Fertigungsindustrie. Ihre Funktionalität erstreckt sich dabei von der Steuerung der Prozesse zur Entwicklung (Herstellung), Modifikation und Nutzung des Virtuellen Produkts bis hin zur lückenlosen Dokumentation seiner "Lebensgeschichte".“

Die modernen PDM-Lösungen sind zum Beispiel eng in die CAD-Systeme integriert und nutzen die grafischen Möglichkeiten für die Visualisierung der Daten, so dass der Konstrukteur nicht mit abstrakten Nummern und Textattributen arbeiten muss, sondern immer eine lebendige Vorstellung von den Teilen und Baugruppen hat, die er verwaltet.

2.2 Produkt Daten Management System bei HELLA

Bei dem Unternehmen HELLA KGaA Hueck & Co werden bereits seit 1999 produktbezogene Daten und Dokumente in einem zentralen Produkt Daten Management (PDM) System verwaltet. Mit dem PDM System wird jedes der HELLA Produkte eindeutig identifizierbar und vollständig beschrieben. Jeder Mitarbeiter hat zu jedem Zeitpunkt Zugriff gemäß seiner Rolle auf die aktuellen Produktdaten.

Das Produkt Daten Management (PDM) System ist bei HELLA eine (SAP R/3) Software von einem Unternehmenssoftware-Anbieter SAP zur konzernweiten Verwaltung der Produktdaten, die sich aus unterschiedlichen Richtungen entwickelt haben. Ein Artikel oder Produkt-Teil kann eine physische oder virtuelle Komponente, eine Baugruppe, Unterbaugruppe oder ein komplettes Produkt repräsentieren. Er hat immer eine eindeutige Bezeichnung oder Nummer. Die zu einem Artikel gehörenden Kerninformationen stehen im Artikelstammsatz. Um komplexe Artikelstrukturen darstellen zu können, sind die Stammdaten lediglich Metadaten, die auf weitere Dokumente, Materialstamm, Stückliste und Änderungsstamm verweisen.

Siehe Abbildung 1 Produktdatenmanagement bzw. Abbildung 2 PDM System bei HELLA [Hell10].

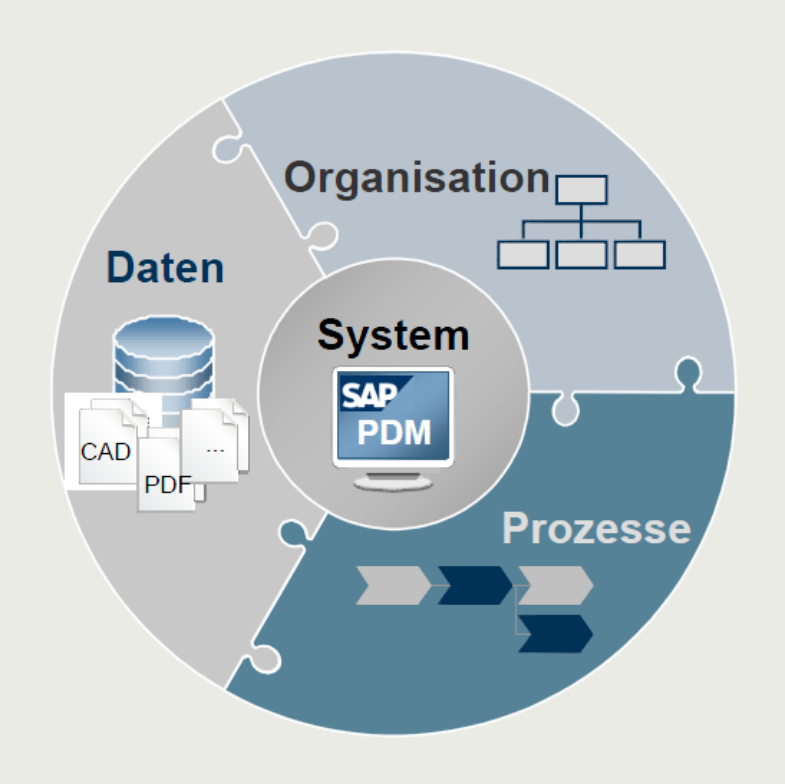


Abbildung 1: Produkt Daten Management [Hell10a]

Abbildung 1 zeigt: Produkt Daten Management ist ein strategisches Konzept zur Unterstützung der Engineering-Prozesse durch strukturierte und konsistente Verwaltung aller produktbeschreibenden Daten sowie die Abbildung von Prozessen. PDM System von SAP ermöglicht die IT-gestützte Erstellung und Verwaltung von Produktdaten über den gesamten Produktlebenszyklus. Die Verwaltung entspricht der vier Objekte Materialstamm, Stückliste,

Änderungsstamm und Dokumente. Abbildung 2 ist ein dazu entsprechender Screenshot des PDM Systems in der Firma HELLA gemäß einem PDM-User Zugriff. Im PDM System bietet sie einige Grundfunktionen sowie Änderungsdienst, Dokument, materialstamm, Stückliste und Sonstiges, die durch Transaktionscode aufgerufen werden können. Es gibt darin noch Firmen-Spezifische Anpassungen.

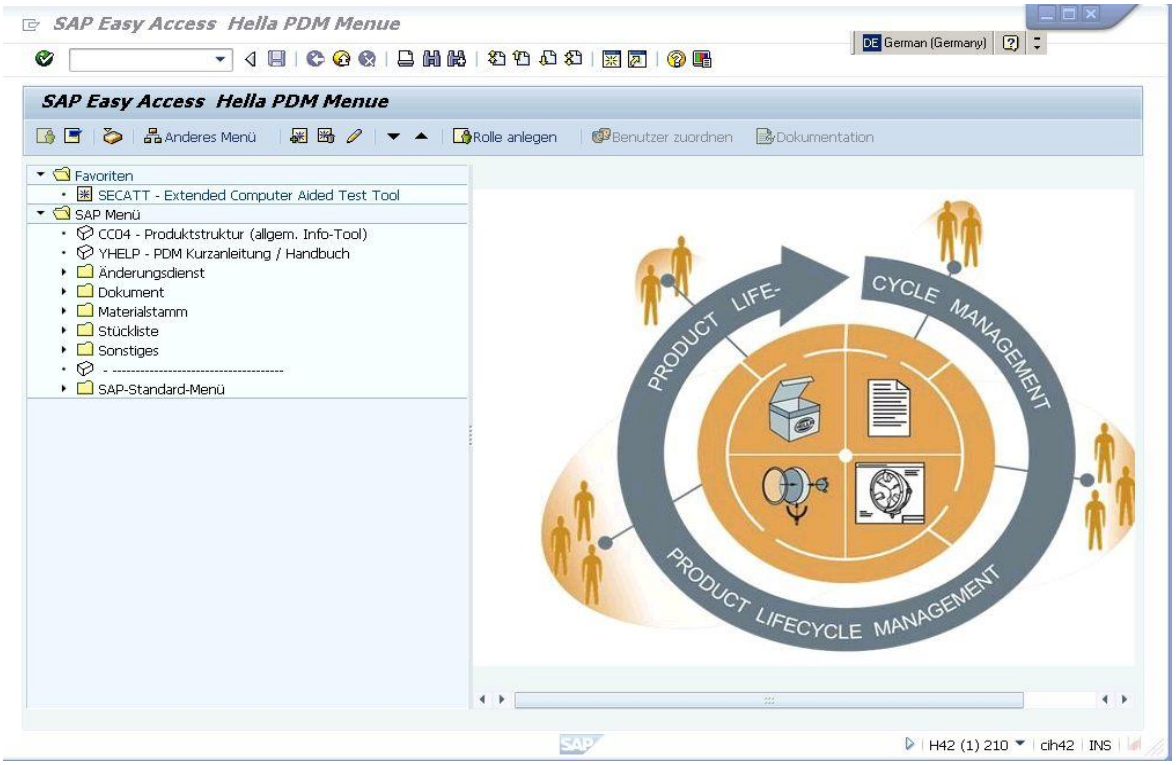


Abbildung 2: PDM System von SAP bei HELLA [Hell10a]

Produktbeschreibende Daten werden als typische Stammdaten in Vier Datenobjekte im PDM System angelegt und gepflegt. Typische Stammdaten bestehen aus Benennung, Identifizierung sowie Material-Nr., Funktionsspezifische Daten (Status, Revisionsstand, Verantwortliche Abteilung) und sonstige.

Im PDM-System werden folgende Objekte (mit Objektanzahl) verwaltet:

- MATerialstamm (Basisdaten): 808.000, davon 513.000 aktive
- Konstruktionsstücklisten (PreBOM): 177.000, davon 154.000 freigegebene
- ÄnderungsSTamm: 41.000
- Produktbeschreibende DOKumente: 960.000, davon 422.000 freigegebene

Materialstammdaten repräsentiert ein/e Bauteil, -gruppe, oder Produkts und enthält alle Grundinformationen zum Material. Die Stücklistenpositionen zeigen das Material innerhalb einer Stückliste. Stückliste bedeutet mehrstufige Anzeige. Dokumentinfosatz (DIS) ist das Verwaltungselement für Dokumente im PDM System. Darin werden alle Informationen zu Dokumenten, die im PDM abgelegt werden, gespeichert. Änderungsstammsatz repräsentiert die Änderungsprozesse sowie Gültigkeiten und Bearbeitungsstand der Änderung.

Eine Übersicht der produktbeschreibenden Daten mit einem konkreten HELLA Produkt, dem „Scheinwerfer“ im PDM System, siehe Abbildung 3. Die Beziehung und Abhängigkeit zwischen diesen 4 Objekten werden anschaulich bezeichnet.

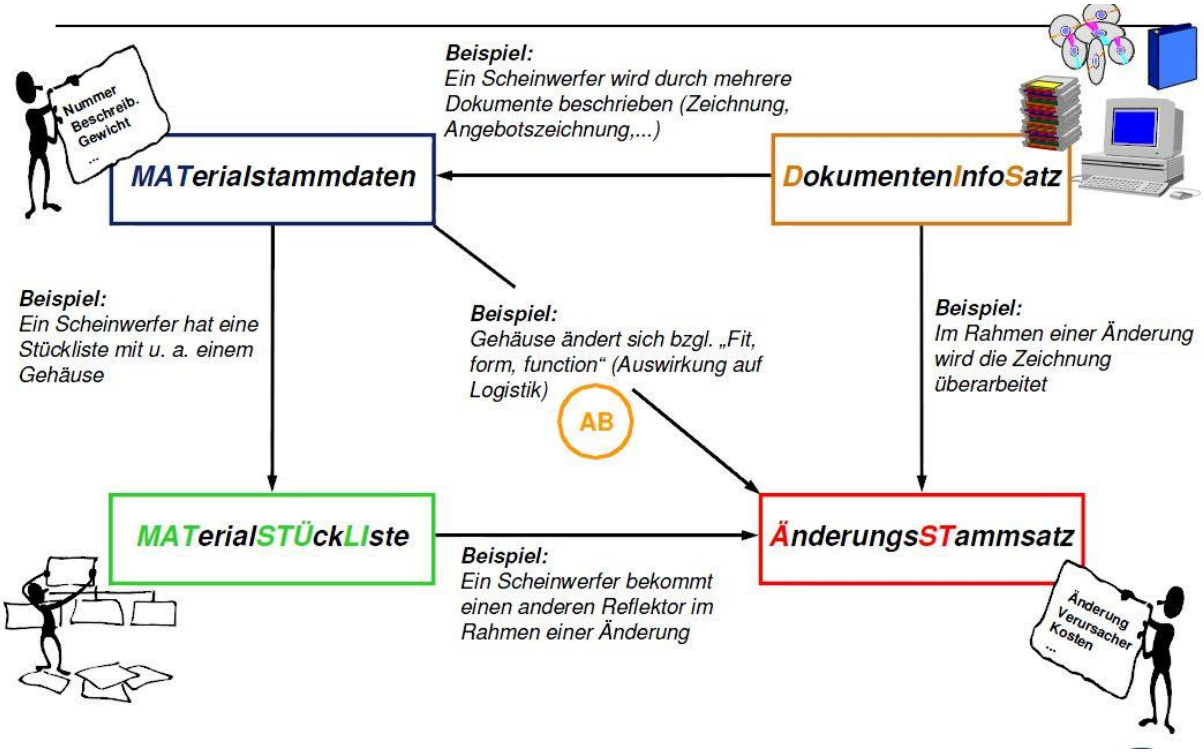


Abbildung 3: Übersicht der produktbeschreibenden Daten im PDM System [Hell10a]

Der Dokumentinfosatz (DIS) enthält die beschreibenden Daten zu einem Dokument und bildet das Bindeglied zur Datenablage. Der Dokumentinfosatz ist quasi wie die Karteikarte einer Bibliothek zum Buch. Auf ihr sind alle relevanten Informationen zum Buch bzw. Dokument und dessen Inhalt vermerkt. Siehe Abbildung 4 Dokumentinfosatz.

Dokumente werden im PDM verschiedenen Dokumentarten zugeordnet, die sich anhand von charakteristischen Merkmalen sowie dem sich daraus ergebenden Bearbeitungsablauf im Unternehmen unterscheiden. Somit bilden die Dokumentarten einen zentralen Bestandteil der Dokumentenverwaltung im PDM System [Hell10a].

Dabei handelt es sich um PDM-Basis-Funktionalitäten, die für das Anlegen, Ändern, Suchen, Einchecken und Ausschecken von Dokumenten erforderlich sind. Einzelne Abläufe können abhängig von Dokumentart und -inhalt anders gestaltet sein.

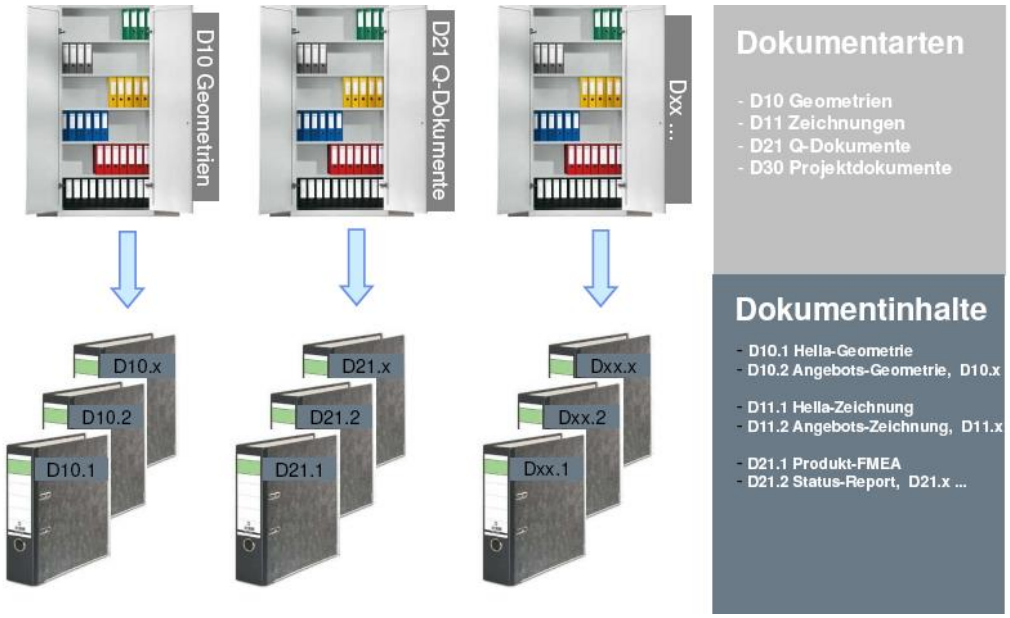


Abbildung 4: Dokumentinfosatz [Hell10a]

Im PDM System existiert noch ein Spannungsfeld von Zugriff und Informationsschutz, die dem Berechtigungskonzept entspricht. D.h. einerseits wollen die Anwender schnellstmöglich auf Dokumente zugreifen andererseits muss Informationsschutz sichergestellt werden, das HELLA Berechtigungskonzept unterstützt die Steuerung der Zugriffsrechte bei verteilter Entwicklung. Abbildung 5 zeigt die weltweite Verfügbarkeit des PDM Systems unter Berücksichtigung des Berechtigungskonzepts.

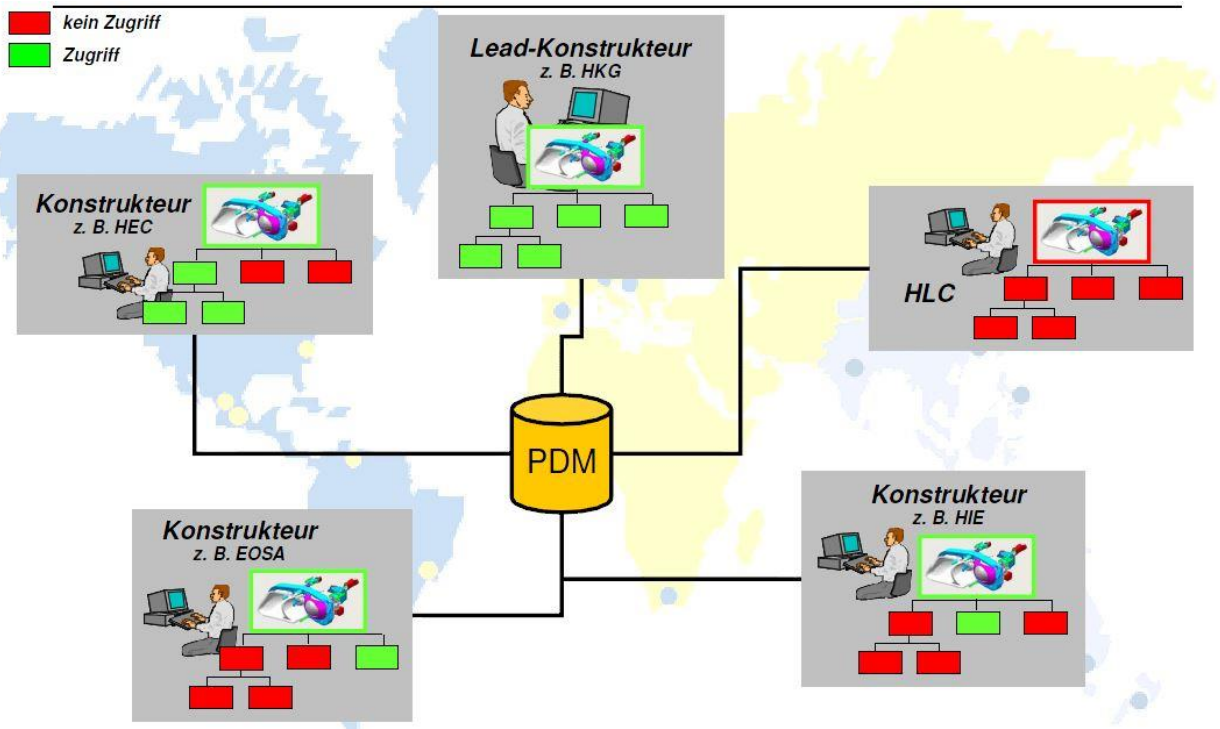


Abbildung 5: Weltweite Verfügbarkeit des PDM-Systems [Hell10a]

3 Einführung des Software-Testens

In diesem Abschnitt werden die Grundkenntnisse für Software Testen erläutert. Wie Qualität definiert ist, wird in Unterkapitel 3.1 erläutert. Die Notwendigkeit von Tests, Teststufen und Testprozess werden im Unterkapitel 3.2 näher beleuchtet. Für das Verständnis der Testautomatisierung werden die Grundlagen aus Kapitel 3.3 vorausgesetzt.

3.1 die Qualitätskriterien von Software

Testen von Software ist beschrieben als stichprobenartiges Ausführen von Testobjekten, die einer Überprüfung dienen, ob das Ist-Verhalten der Software auch dem Soll-Verhalten entspricht. Es dient also dem Erkennen, ob die durch die Softwareentwickler bereitgestellte Software auch den im Vorfeld der Programmierung definierten Anforderungen und Konzepten des Auftraggebers entspricht und fehlerfrei ist [StEs13].

Qualitätsmanagement ist definiert durch die DIN-EN-ISO-9000:2005, welche Qualitätsmanagement darstellt als aufeinander abgestimmte Maßnahmen zur Leitung und Lenkung von Organisationen in Bezug auf Qualität, welche im Allgemeinen Qualitätsziele, -planung, -durchführung, -sicherung und -verbesserung umfassen [StEs13].

Die ISO-Norm definiert Qualität folgendermaßen: Die Gesamtheit von Merkmalen einer Einheit [SATF] bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen. Das bedeutet, dass Qualität die Diskrepanz zwischen dem festgelegten Soll-Zustand eines Produktes und dessen Ist-Zustand ist.

Diese beschreibt ein Qualitätsmodell, das die Eigenschaften von Software in 6 Kategorieneinteilt:

- **Funktionalität** ist die Übereinstimmung der Software mit der Spezifikation. Sie ist eine der wichtigsten Qualitätsmerkmale von Software, das ist die grundlegenden Eigenschaften zu den Funktionen der Software, was sie funktional leisten soll und wie.

- **Zuverlässigkeit** ist ein Maß, das sich aus der fehlerfreien Funktion einer Software über einen Zeitraum ergibt. Diese Eigenschaft beinhaltet deshalb auch die Fehlertoleranz sowie die Fähigkeit einer Software aufgetretene Fehler zu behandeln und weiter zu funktionieren.

- **Benutzerfreundlichkeit** beinhaltet Unterkategorien wie die Erlernbarkeit, Verständlichkeit und Benutzererwartungskonformität. Sie ist ein Maßstab für den Aufwand, den ein Benutzer der Software für das Verstehen und die Verwendung der Software aufbringen muss [Schm].

Siehe Abbildung 6. Die Qualitätskriterien für Software-Tests nach ISO-9126. Diese Norm definiert die dargestellten Qualitätskriterien für Software in zweistufiger Struktur. Danach wird deutlich, dass unter Softwarequalität mehr als nur Fehlerfreiheit verstanden wird. Die

Qualitätsmerkmale benennen unterschiedliche Eigenschaften, die die Software aufweisen soll. Dies ist auf oberster Ebene.

Auf unterer Ebene sind die Qualitätskriterien zu Effizienz, Änderbarkeit (Wartbarkeit) und Übertragbarkeit ausgerichtet, die ggf. erforderliche Anpassungsmaßnahmen ermöglichen und vereinfachen soll.

- **Effizienz** misst die Leistung der Software durch deren Zeitbedarf bei der Verarbeitung oder deren Ressourcenverbrauch.

- **Änderbarkeit** ist ein Maß, das durch den Aufwand bestimmt ist, der betrieben werden muss, um die Software zu verbessern, Fehler aufzufinden und zu korrigieren.

- **Übertragbarkeit** ist die Fähigkeit einer Software, in unterschiedlichsten Anwendungsumgebungen zu funktionieren.

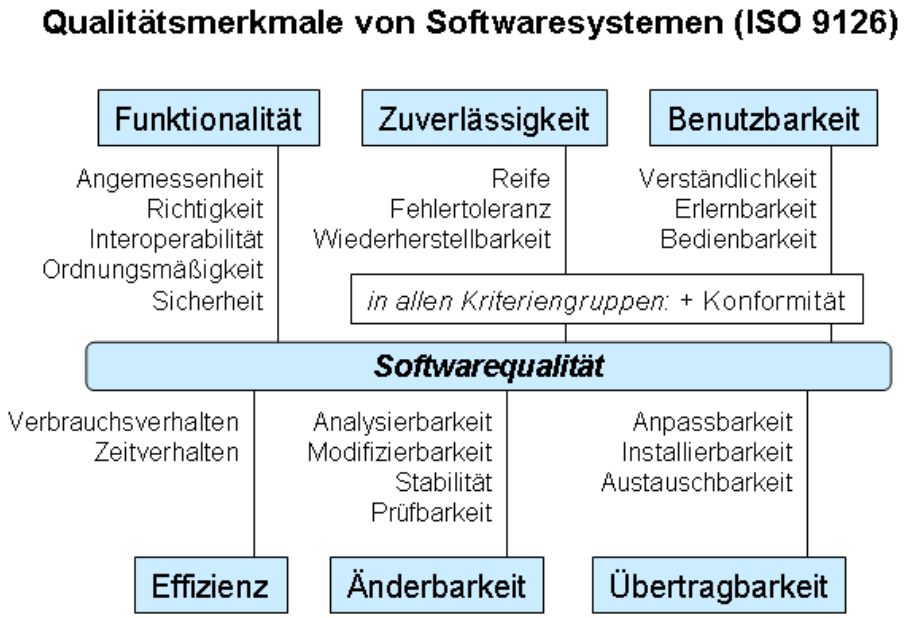


Abbildung 6 Die Qualitätskriterien für Software nach ISO 9126 [Wiki12b]

Diese 6 Kriterien stellen einen Rahmen dar, der für einzelne Softwareprodukte in individuellen Spezifikationen konkretisiert werden kann, um in der Softwareentwicklung berücksichtigt zu werden [Wiki12b].

3.2 das Testen im Allgemein

In diesem Kapitel erfolgt eine Notwendigkeit von Test, Teststufen in der SAP-Projektmethodik und Testprozess: Warum braucht man Tests und wie testet man genau.

3.2.1 Notwendigkeit von Tests

„Kein Produkt menschlicher Intelligenz kommt fehlerfrei zur Welt [Wien94].“ es geht ebenso für Software. Bei der Softwareentwicklung werden die zugrunde liegenden IT-Infrastrukturen immer komplexer, Software muss in eingeschränkten Entwicklungszeitraum höchsten Anforderungen genügen, besonders ist das bei betrieblicher Standardsoftware. Um für diese komplexen und vielfältigen Funktionalitäten den reibungslosen Ablauf weltweit zu gewährleisten, sind im Qualität Management intensive Softwaretests notwendig.

Testen ist das einzige Verfahren, um ein auslieferbares Software-Produkt zu analysieren. Dabei kann es die gesamte Software- und Hardware-Umgebung des Programms berücksichtigen [Jose13].

Softwaretests spielt eine wichtige Rolle besonderes bei den Firmen-Spezifische Anpassungen von Unternehmenssoftware. Nur wer seine Anwendungen richtig testet, vermindert das Risiko und sorgt damit für mehr Stabilität. Sowie im Kapitel 1.1 Hintergrund der Forschung beschreibt.

3.2.2 Teststufen in der SAP-Projektmethodik

Im Rahmen von Einführungs-, Upgrade - oder Entwicklungsprojekten werden in der Regel 7 Teststufen unterschieden. Diese lassen sich anhand der Integrationsebene ordnen, auf der die Software getestet wird. Im SAP-Umfeld wird funktional nach einer Bottom-Up-Strategie [Hoop03]. getestet. d.h. vom Teil zum Ganzen. Diese erlaubt im zeitlichen Verlauf eines Projektes nach Integrationsstufen vorzugehen. Die unterste Stufe der funktionalen Tests bilden im SAP-Umfeld in der Regel die Funktionstests, auch Modultests oder Unit-Tests genannt. „Ein Funktionstest überprüft eine einzelne Transaktion oder einen Funktionsbaustein. Er konzentriert sich schwerpunktmäßig auf die inneren Funktionen, nicht auf die Schnittstellen oder die Integration[HLT06].“ Siehe Abbildung 7 Bottom-Up-Modell der Integrationsstufen.

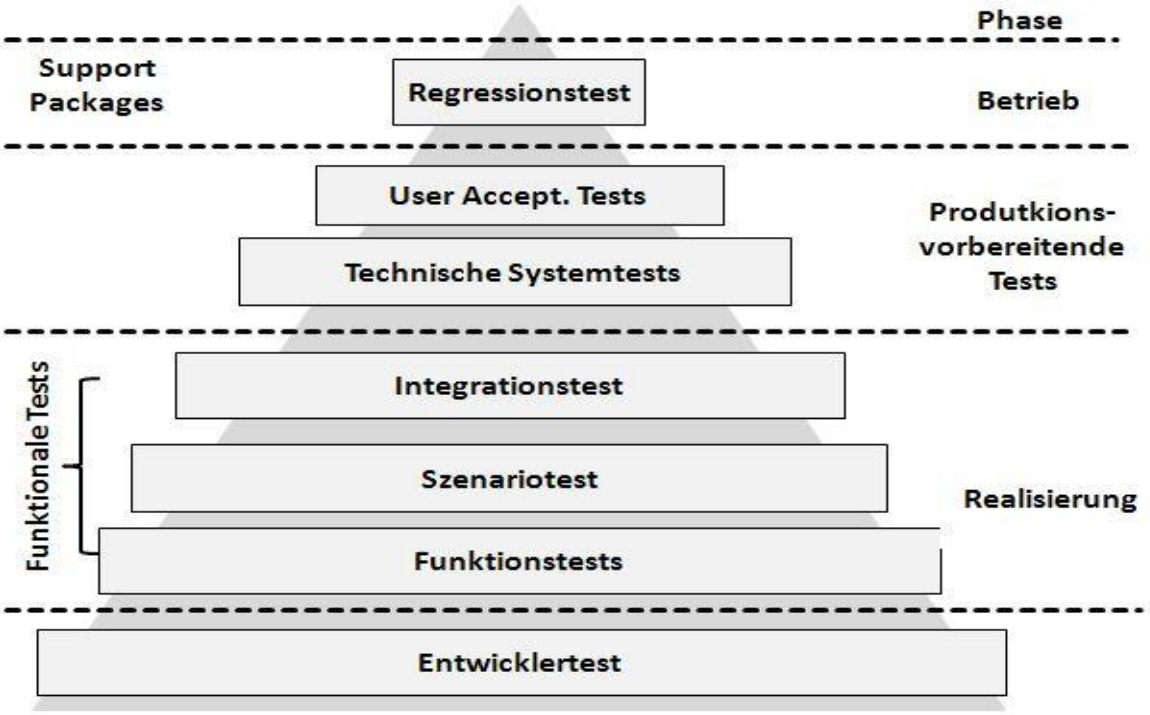


Abbildung 7: Bottom-Up-Modell der Integrationsstufen [HLT06]

Teststufe 1: Entwicklertest

Wie der Name bereits sagt, wird diese Testart in der Entwicklungsphase vom Entwickler selbst durchgeführt. Die Tests finden auf technisch niedriger Ebene statt und können neben der formalen Ablauffähigkeit auch einzelne Codezeilen und Funktionsbausteine testen [Gaer12].

Teststufe 2: Funktionstests

Ein Funktionstest, auch Modultest oder Unit-Test genannt, überprüft eine einzelne Transaktion oder einen Funktionsbaustein [Gaer12]. Dabei liegt der Fokus des Tests auf den inneren Funktionen und nicht auf der Schnittstelle oder Integration.

Teststufe 3: Szenariotests

Mit einem Szenario Test werden mehrere zusammenwirkende Transaktionen z.B. in einem Geschäftsprozess getestet. Bei dieser Testart soll das Zusammenspiel einzelner Transaktionen und ihrer Schnittstellen geprüft werden [Gaer12].

Teststufe 4: Integrationstest

Die fehlerfreie Integration von SAP-Lösungen mit Non-SAP-Anwendungen und Systemschnittstellen ist in einer heterogenen Systemlandschaft von hoher Bedeutung und muss daher durch einen Integrationstest geprüft werden[Gaer12]. Die Geschäftsprozesse werden hierfür in Szenarien abgebildet und die Software wird entlang dieser modul- und systemübergreifenden Szenarien getestet.

Teststufe 5: Technische Systemtests

Bei den technischen Systemtests wird das Gesamtsystem mit allen Komponenten, wie Datenbank, Applikationsserver, Netzwerk usw. [Gaer12], gegen die gesamten Anforderungen in einer simulierten Produktivumgebung getestet.

Teststufe 6: User Accept. Tests

Ein weiterer bedeutender Test vor dem Go - Live ist der User Acceptance Test. Hauptsächlich wird bei diesem Test auf Probleme in der Bedienbarkeit geprüft [Gaer12]. Dieser Test wird durch den Anwender manuell durchgeführt, weshalb ein erheblicher Aufwand verursacht wird und mitunter keine automatisierten Tests verwendet werden können.

Teststufe 7: Regressionstest

Das Ziel eines Regressionstests ist es sicherzustellen, dass nach einer prozessualen oder technischen Veränderung keine Fehler in der Funktionalität oder am Geschäftsprozess entstehen. Im SAP-Umfeld [Gaer12]werden solche Tests hauptsächlich nach Korrekturen, Enhancement-/Support-Package-Implementierungen oder nach Änderungen des Customizings eingesetzt.

Diese Diplomarbeit wird auf den Bereich „Funktionale Tests“ fokussiert. Die von [Koch03] erläutert ist, darauf beschränkt sich Blackbox Tests. Diese Funktionstests werden auf Basis der Spezifikationen erstellt. Sie sollten gewährleisten, dass die Anforderungen auch durch das System erfüllt werden.

Im SAP-Umfeld werden schwerpunktmäßig die datengetriebenen Blackbox Tests durchgeführt, Whitebox Test besitzt eine eher geringe Relevant [HLT06].

Blackbox Tests und Whitebox Tests: Wenn man den Aufbau des zu testenden Systems nur von außen betrachtet, ohne die interne Struktur zu verwenden, so spricht man von einem Blackbox-Test. Die Durchführung von Blackbox-Tests setzt nicht zwingend das Vorhandensein eines User-Interface voraus. Da Blackbox - Tests ohne Wissen über den Ablauf des Programms vorgenommen werden, fokussieren sie auf die Eingabedaten und die Ergebnisse. Die Qualität des Tests steht und fällt mit der Qualität der verwendeten Testdaten [HLT06].

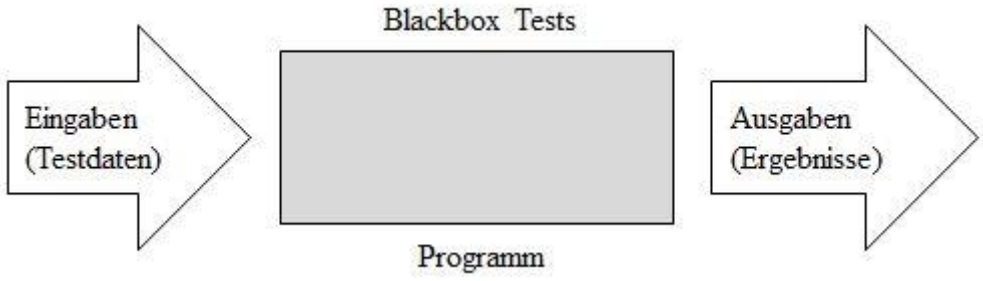


Abbildung 8: Blackbox Tests

Wird wissen über die interne Struktur der Software verwendet, zum Beispiel den Quellcode oder interne Schnittstellen, spricht man von einem Whitebox Tests. Siehe Abbildung 9 Whitebox Tests.

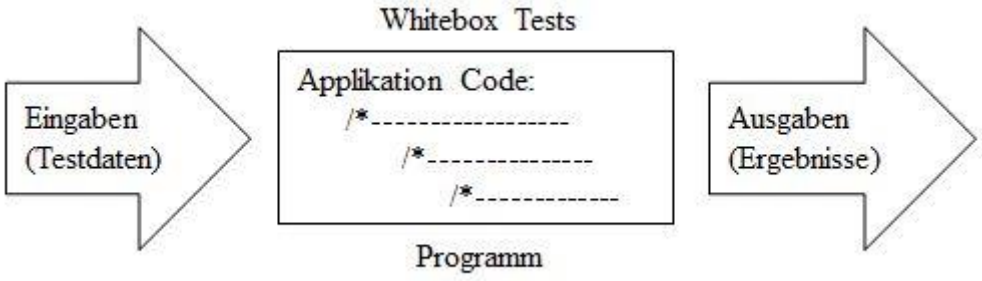


Abbildung 9: Whitebox Tests

3.2.3 Testprozess

Die Tätigkeit des Testens ist in Softwareentwicklungsmodellen wie dem Wasserfallmodell zwar vorgesehen, die bloße Eingliederung in den gesamten Prozess reicht aber nicht aus um Tests strukturiert durchführen zu können. Hierzu ist eine detaillierte Aufgabenplanung der Testaktivitäten notwendig. Nach dem International Software Testing Qualifications Board (ISTQB) gliedern sich Softwaretests in insgesamt 5 Phasen:

Phase 1: Planung & Steuerung

Phase 2: Analyse & Design

Phase 3: Realisierung & Durchführung

Phase 4: Auswertung & Bericht

Phase 5: Abschluss

Abbildung 10 Fundamentaler Testprozess [StEs13] zeigt diese Phasen inklusive möglicher Durchlaufwege. Obwohl die Phasen sequenziell dargestellt sind, können diese sich auch überschneiden oder parallel laufen. Der Testprozess teilt sich dabei von Analyse und Design, Realisierung und Durchführung, Auswertung und Bericht sowie folglich der Abschluss. Dabei geht die Auswertung und der Bericht rekursiv auf Realisierung & Durchführung und Analysiere & Design zurück. Der Abschluss geht auf Planung & Steuerung rekursiv zurück.

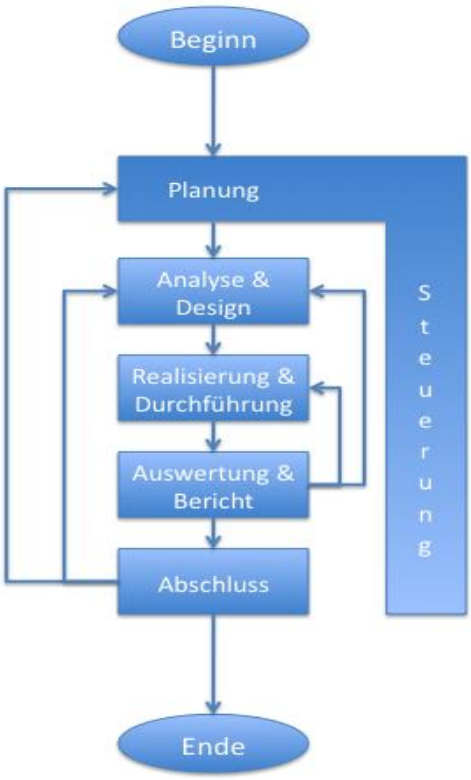


Abbildung 10: Fundamentaler Testprozess [StEs13]

In dieser Diplomarbeit wird eine Vorgehensweise nach dem Wasserfallmodell betrachtet, dabei wird der eigenständige Testprozess gemäß ausgewähltem Testtool entworfen. Im Kapitel 4.2 „Konzeptentwicklung zur Testautomatisierung“ wird ein Ablauf eines Testprozesses mit konkreter Komponente ausführlich beschrieben.

3.3 Testautomatisierung

Unter Testautomatisierung (auch Testautomation) ist die Automatisierung von Aktivitäten im Test zu verstehen. Sie ist ein Bereich des Testens, in dem die Arbeit des Testers erleichtert werden kann. Die Testautomatisierung auf eine allgemeine Bedeutung wird im Kapitel 3.3.1 eingegangen. Die Auswahl der Werkzeuge zur Testautomatisierung wird im folgenden Kapitel 3.3.2 ausführlich vorgestellt.

3.3.1 Was bedeutet Testautomatisierung

Es gibt unter Fachleuten, Testern und Programmierern unterschiedliche Ansichten und Erwartungshaltungen, was Testautomatisierung bedeutet und was für Auswirkungen sie hat. Einige Fachleute der Testautomatisierung, reduzieren deren Nutzen auf die Vereinfachung von Regressionstests [Zamb98]. Dies ist eine zu starke Einschränkung der Möglichkeiten der Testautomatisierung, denn sie betrachtet lediglich die Testdurchführung und Auswertung als Gegenstand der Automation. Eine allgemeinere Definition [PKS02] gibt: Ein Testtool ist ein automatisiertes Hilfsmittel, das bei einer oder mehreren Testaktivitäten, beispielsweise Planung und Verwaltung, Spezifikation, Aufbau von Ausgangsdateien, Testdurchführung und Beurteilung, Unterstützung leistet. Zerlegt man die Gesamtaufgabe des Testens also in seine

Teilaufgaben, so ergeben sich viele Möglichkeiten zur Automatisierung und Vereinfachung dieser Teilaufgaben durch Werkzeugunterstützung. Im Einzelnen sind dies:

- Testmanagement (Ressourcenplanung, Teamauswahl, Testwerkzeugauswahl)
- Testplanerstellung (Festlegung eines Zeitplans, von Teststrategien und Testkriterien)
- Testdesign (Erstellung von Testszenarien und Testfällen)
- Testdurchführung
- Testauswertung

Besonders lohnenswert ist eine Automatisierung bei sich wiederholenden Aufgaben, zu denen besonders die Testdurchführung und die anschließende Auswertung zählen. Dies ist der Grund für die derzeitige Konzentration von Testwerkzeugen, die auf diese Teilaufgaben spezialisiert sind. Jedoch sind auch die Testfallerstellung und die Testdatengenerierung, bei sich ändernden Anforderungen eines Produkts durch Weiterentwicklung oder Modifikation, ein sich wiederholender Vorgang. Dieser ist zudem sehr zeitaufwändig, denn es muss besonders sorgfältig gearbeitet werden, um die geforderten Testkriterien zu erfassen und jeden Testfall korrekt auszuarbeiten. Es gibt jedoch nur sehr wenige Testwerkzeuge, die diese Aufgabe erleichtern, wie sich in Kapitel 3.3.2 zeigen wird. Der Grund hierfür ist in der Art der Tests zu suchen, auf welche die Testwerkzeuge zur Testdurchführung spezialisiert sind. Dies sind meist Blackbox-Tests, für deren Testdesign das Tool auf eine automatisch verwertbare Spezifikation zurückgreifen müsste. Eine solche Voraussetzung, also eine Spezifikation in formaler Form, ist aber in den seltensten Fällen gegeben, so dass die Testfallerstellung bei automatisierten Blackbox-Tests Aufgabe des Testers bleibt. Einige Fachleute der Testautomatisierung gehen sogar soweit, dass sie die Testautomatisierung der Automatisierung von Blackbox Tests gleichsetzen [PiMu03]. Doch auch diese Einschränkung der Testautomatisierung auf eine Testmethode kann so nicht hingenommen werden [Gaer12]. Testautomatisierung ist eine Automatisierung einer oder mehrerer der Teilaufgaben des Testens.

Im Allgemeinen ist Testen abhängig vom Umfeld (nicht kontextfrei): Je nach Einsatzgebiet und Umfeld des zu prüfenden Systems ist das Testen anzupassen. Keine zwei Systeme sind auf die exakt gleiche Art und Weise zu testen. Intensität des Testens, Definition der Ausgangskriterien usw. ist bei jedem System entsprechend seines Einsatzumfeldes festzulegen.

Testen dient primär dazu, die Fehler in der Software aufzudecken. Die Abwesenheit von Fehler soll nicht bewiesen werden, und kann es aus Zeit-und Komplexitätsgründen auch meistens nicht. Anhand der gefundenen Fehler kann die Qualität des Produktes gemessen werden, und schließlich Anstoß zur Verbesserung dieser geben. Die Korrektur gefundener Defekte ist dabei nicht Aufgabe des Testens [Koch03].

Vollständiges Automatisierungstesten ist auch nicht möglich [Gaer12]: Ein vollständiger Test, bei dem alle möglichen Eingabewerte und deren Kombinationen unter Berücksichtigung aller unterschiedlichen Vorbedingungen ausgeführt werden, ist nicht durchführbar, mit Ausnahme bei

sehr trivialen Testobjekten. Tests sind immer nur Stichproben, und der Testaufwand ist deshalb nach Risiko und Prioritäten zu steuern.

Funktionsorientierte automatisierte Tests sind daher nahezu immer Regressionstests, also eine "Wiederholung aller oder einer Teilmenge aller Testfälle, um Seiteneffekte von Modifikationen in bereits getesteten Teilen der Software aufzuspüren [Wiki12a]."

Meistens wollen die Anwendungstests per Capture/Replay automatisieren d.h. Testvorgang wird mit der simplen Aufzeichnung der Testschritte moduliert und anschließender Parametrisierung durch ein Testautomation-Werkzeug.

Capture & Replay als eine Methode für Testautomatisierung: Grundlegende Anforderungen von Capture & Replay sind:

- Capture Mode
- Programming Mode
- Checkpoints
- Replay Mode

Alle Anwendungstests werden in Capture Mode Schritt vom ausgewählten Werkzeug aufgezeichnet. Die Aufzeichnung erfolgt mittels eines Skriptes, welches den genauen Ablauf des Testvorgangs und die Testparameter festhält. Die im Capture Mode generierten Testschritte werden in Skripten-Format gespeichert. Mit Hilfe der Skripte können dann Komplexe Testszenarien erstellt werden. Das Testskript-Format ist abhängig von Testwerkzeug und ist editierbar. Mit Checkpoints [OGTC06] ist möglich einzelne Testszenarien nach Anforderung zu erstellen und später zur einen Testblock zusammenzufassen, gleichzeitig zur Verifizierung der Testergebnisse. Checkpoints kümmern sich also um die tatsächlichen Messergebnisse.

Capture & Replay als eine Testautomatisierungsmethode ermöglichen eine Wiederverwendung innerhalb der Tests. Besonders bei häufigen Regressionstests stellt sich bei der Testautomatisierung ein deutlicher Kosten/Nutzen Faktor dar. Interessant sind die zusätzlichen Möglichkeiten, die solche Werkzeuge bieten. Beispielsweise kann auf einfache Weise ein Anwenderverhalten simuliert und ausgewertet werden, was sicherlich im Zusammenhang mit nicht-funktionalen Anforderungen wie Usability oder Performance.

Die Auswahl der Testwerkzeuge wird im nächsten Kapitel 3.3.2 beschrieben.

3.3.2 Werkzeug zur Testautomatisierung

Es existieren die verschiedensten Arten von Testwerkzeugen, die zur Unterstützung von Tests oder zur Testautomatisierung eingesetzt werden können. Die verschiedenen Testwerkzeuge lassen sich in Gruppen einordnen, die je nach Testaktivität in den einzelnen Phasen der Tests unterstützen können. Normalerweise werden in Projekten nicht alle Arten von Testwerkzeugen eingesetzt. Eine vorhergehende Analyse der Testwerkzeuge ist jedoch vor der Auswahl

unbedingt notwendig, da sonst nicht bestimmt werden kann, ob ein Werkzeug sinnvoll innerhalb eines Projektes eingesetzt werden kann [Gada10]

Auf der unteren Abbildung 11 sieht man dass es verschiedene Arten von Testwerkzeugen im Testprozess gibt:

- Werkzeuge für Testmanagement
- Werkzeuge für Testspezifikation
- Werkzeuge für Durchführung

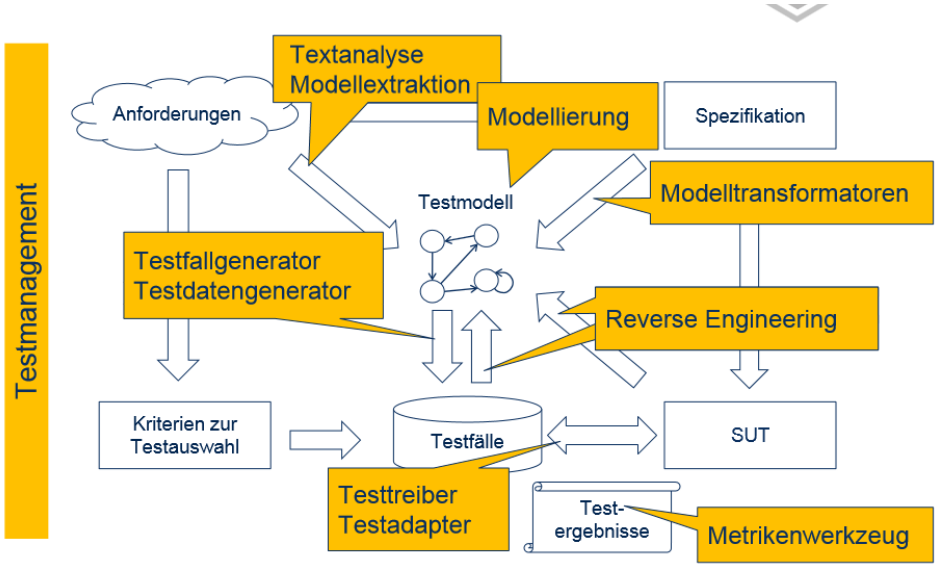


Abbildung 11: Beispiel Modellbasiert Testwerkzeug [Gada10]

Während des Toolauswahlprozesses gehört es zu den Aufgaben des Testteams, ein geeignetes Werkzeug für die Organisation aus den am Markt erhältlichen Produkten auszuwählen. Der Implementierungsprozess stellt sicher, dass das Testteam das ausgewählte Werkzeug in der Organisation effektiv nutzt. Bevor man Tests automatisiert, vergleicht das Testteam das einsetzbare Werkzeug mit der Anforderung. Zur Unterstützung einer Entscheidung wird der folgende Lösungsweg (aus bestehende SAP Lösungstechnologie) betrachtet.

a) SAP Solution Management (SolMan) als SAP Standard Lösung betrachten

Es ist die Möglichkeit, den SAP Solution Management (SolMan) für das Testmanagement zu nutzen. Er kann als auch ein „Test Haupt-Tool“ bezeichnet werden und dient in der Regel in der Basisadministration als Tool Set für die Lösungsverwaltung.

Das „Test Haupt-Tool“ SolMan unterstützt den gesamten Testprozess, von der Vorbereitung, Planung und Durchführung manueller und automatisierter Tests bis hin zur Erstellung von Statusberichten über den Testverlauf. Abweichungen werden dokumentiert und die Testergebnisse revisionssicher abgelegt.

Voraussetzung für den Einsatz des SolMan ist zunächst das Grundcustomizing sowie die Koppelung desselben an die Systemlandschaft (SolMan 7.0 EHP1).

Dazu bietet der SolMan drei Werkzeuge:

- Die Test Workbench: die unterstützt die Testplanung und Durchführung. dazu werden manuelle oder automatische Testfälle angelegt und systematisch archiviert. Für die Durchführung wird eine Auswahl von Testfällen in einem Paket zusammengefasst und einem Tester zugeordnet. Innerhalb eines Pakets können die einzelnen Fälle in Sequenzen angeordnet werden, um den genauen Ablauf während eines bestimmten Zeitrahmens zu regeln.

- Der Service Desk: er dient der Erfassung und dem Tracking von Abweichungen. Können diese in den Unternehmen nicht selbst korrigiert werden, besteht die Möglichkeit, die Abweichungsmeldungen an den SAP Support weiterzuleiten.

- eCATT: Die Testautomatisierung wird durch eCATT (extended Computer Aided Test Tool) unterstützt. Das Werkzeug bietet auch für Mitarbeiter aus Fachabteilungen die Möglichkeit, Testskripts ganz einfach durch die Aufzeichnung von Benutzereingaben zu generieren. Mit dem generierten eCATT-Skript können Testdaten automatisiert erzeugt und der Ablauf einer Transaktion erprobt werden.

b) Wer nutzt schon den SolMan für die Prozesse im Test-und Abweichungsmanagement

Es liegt an der Organisation im Unternehmen, weil die Gründe für die Auswahl des Werkzeugs unterschiedlich sind. (Fachverantwortlicher Experte im PDM System oder Entwickler und Pfleger des PDM Systems)

Bei der Auswahl des Werkzeuges sind die Varianten „teuer und leistungsfähig“ Oder „historisch gewachsen und auf Office-Produkten basierend“ verbreitet [Inno13].

Konkrete Beschreibung zu Variantenreich:

- „teuer und leistungsfähig“: Testtool ist teuer aber relativ stark leitungsfähig

- „historisch gewachsen und auf Office-Produkten basierend“: gewöhnliche Arbeitsschritte und geringe Einführungskosten (Entscheidung zu selbst entwickelter Lösung im Testmanagement)

c) Vorteile und Nachteile den SolMan als Alternative in Betracht

Im Vergleich zu anderen Testwerkzeugen am Markt haben die SAP Solution Management folgende Vorteile, die aus Allgemeinseite und Durchführungsseite bzw. in Module besteht.

Vorteile aus Allgemeinseite:

- Die Nutzung der einigen Teil-Werkzeuge(Beispiel eCATT) ist frei von Lizenzkosten

- Die Anwender erkennen sich im den SAP-Oberflächen wieder

Vorteile aus Durchführungsseite:

- Unterstützungsthema in SAP Software

- selbst Bestandteile sind gut integrierbar

- Reversionssicherheit

Vorteile von den drei weiteren Modulen [Inno13] in SolMan:

- Das Projektmanagement: So kann die komplette Steuerung eines Projekts mit dem Projektmanagementmodul im SolMan abgebildet werden.

- Knowledge Management: Durch das Knowledge-Management können beispielsweise notwendige Hintergrundinformationen zur Testdurchführung oder Fehlerbehebung direkt mit dem jeweiligen Vorgang verknüpft werden. Damit sind alle relevanten Informationen jederzeit parat.

- Der SAP Change Management: Mit dem SAP Change Manager werden Fehlerkorrekturen automatisiert per Transportauftrag in ein Testsystem zum Nachtest importiert. (So besteht es ein kompletter Workflow von der Abweichungsmeldung über die Korrektur bis zum Retest.)

Nachteile aus Funktionalen Schwächen gegenüber den Werkzeugen (der etablierten Spezialanbieter): hier wurde das nur beispielweise [Inno13] aufgelistet:

- Der SolMan bietet beim Reporting nicht die Möglichkeit, die einzelnen Testfälle auf Tagebasis zu planen. (im Zuge der Weiterentwicklung sind jedoch bereits einige Verbesserungen implementiert worden) Das neue Release 7.0 EHP1 bietet funktionale Erweiterungen wie die Abbildung von Sequenzen (z.B. für die Definition von Abhängigkeiten zwischen Testschritten in End-to-End-Tests) und ein ausgefeiltes Workcenter-Konzept für eine komfortable Bedienung.

Häufig sind in Unternehmen umfangreiche Testfallbibliotheken in Microsoft Word oder Excel vorhanden. Dahinter stecken in der Regel jahrelange Aufbauarbeit und viel gesammeltes Wissen über zu testende Systeme, Module und Prozesse. Diese wertvollen Daten gehen durch den Einsatz der SAP Test Workbench nicht zwangsweise verloren, da die Office-Dokumente als Anlage zu den Test-Workbench Testfällen in den SolMan hochgeladen werden können. [Inno13] Unternehmen, die wissen wollen, ob sich der Einsatz der Testwerkzeuge für sie lohnt, müssen einige entscheidende Parameter betrachten, z. B. die Anzahl der Testvorhaben, die Anzahl der Tester in den jeweiligen Testvorhaben, die Anzahl der zu verwaltenden Testfälle oder die Möglichkeiten zur Wiederverwendung (beispielsweise in Release Wechsel Projekten). Die Werkzeuge sind auch sukzessive einführbar. Denkbar ist z. B. der Aufbau einer eCATT-Bibliothek für die Automatisierung von Regressionstests oder für die Bereitstellung von Testdatenkonstellationen. Viele Betriebe starten mit der Nutzung der SAP Test Workbench und

setzten dann darauf aufbauend den SAP Service Desk ein. Alles in allem hat sich die Praxistauglichkeit der Werkzeuge in Großprojekten mit weit über 1000 Testfällen in mehreren Testzyklen bewiesen. Gleichzeitig ist die Einführung kostengünstig und einfach.

d) Fünf Argumente von [Inno13] zusammengefasst für den Einsatz des SolMan im Testmanagement:

- Integriertes Test- und Abweichungsmanagement

Test- und Abweichungsmanagement sind in einem Werkzeug integriert. Dadurch besteht eine automatisierte Referenzierung zwischen Testfall und Abweichungsmeldung. eCATT-Skripte lassen sich direkt aus den Testfällen aufrufen und dokumentieren die Ergebnisse maschinell.

- Transparenter Testverlauf

Der aktuelle Status der Bearbeitung der Testfälle sowie der Status im Abweichungsmanagement lassen sich jederzeit über die Reportingfeatures im Service Desk und in der Test Workbench abrufen.

- Wiederverwendbarkeit für zukünftige Projekte

Testfälle, Testfallkataloge, Testpakete und insbesondere eCATT-Skripte können für zukünftige Projekte wiederverwendet werden, gerade für Regressionstests in Releasewechsel-Projekten eine interessante Option.

- Anwenderfreundlichkeit

Die Bedienung der Tools erfolgt über die vertraute SAP-Oberfläche. Somit ist der Einarbeitungs- und Schulungsaufwand für Tester und Testmanager gering.

- Geringer Einführungsaufwand

Das Grundcustomizing für den Service Desk und die Test Workbench ist jeweils innerhalb von zwei bis drei Tagen realisierbar. So ist schon innerhalb kurzer Zeit eine effektive Nutzung im Testmanagement möglich. eCATT-Skripte lassen sich je nach Komplexität in einem Zeitraum von einer Viertelstunde (z. B. für die Aufzeichnung der Anlage eines Geschäftspartners) bis zu mehreren Stunden (z. B. bei verschachtelten Aufrufen) erstellen.

e) Eine Entscheidung über Teil-Testtool aus SAP Solution Manger

Auf Grund der teuren Lizenz- und Wartungsgebühren des ganzen SolMan für die Firma HELLA KGaA Hueck & CO. wird eCATT zuerst als Teil-Testtool zur Testautomatisierung in der Durchführung-/Auswertungsphase ausgewählt, damit wird ein Individuallösungskonzept in diese Diplomarbeit erstellt. Die Testfälle, Testfallkataloge, Testpakete und insbesondere eCATT-Skripte können für zukünftige Projekte wiederverwendet werden. Zum Überblick der bestehenden Test-Lösung mit SolMan siehe Abbildung 12 bestehender Lösungsweg.

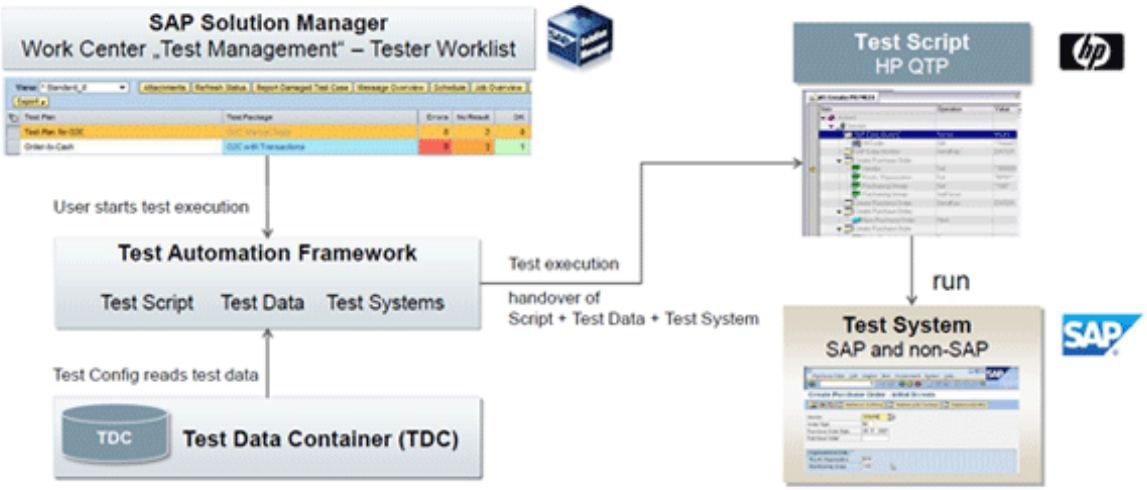


Abbildung 12: bestehender Lösungsweg [SAP13d]

Ein gemeinsamer Überblick für Integration mit ausgewählten Testtool eCATT siehe Abbildung 13 Integration des SAP Solution Mangers mit eCATT Funktionsumfang.

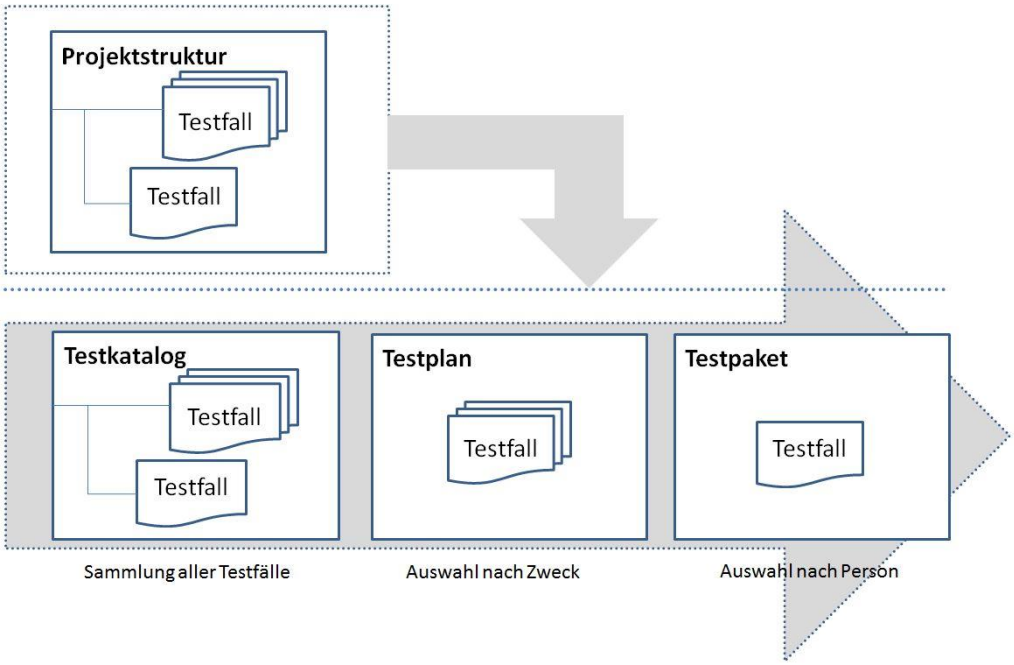


Abbildung 13: Integration des SAP Solution Mangers mit eCATT Funktionsumfang

f) Warum sollte man eCATT insbesondere bei HELLA einsetzen und sich nicht dafür entscheiden, etwas Eigenes zu programmieren

Die unteren Punkte werden durch praktische Durchführung der Tests mit Hilfe [Hell10b] zusammengefasst:

- eCATT ist in den Anwendungsserver integriert und deshalb ist eCATT in der Lage, Ergebnisse auf verschiedenen Systemebenen zu prüfen.

- Es ist möglich, durch einen direkten Zugriff auf Tabellen die korrekte Verbuchung von Ergebnissen in der Datenbank zu überprüfen, ohne den Transaktionsfluss der Testsequenz zu unterbrechen.

- Durch eCATT wird nicht nur aufgezeigt, ob eine Transaktion ein korrektes Ergebnis ausgeworfen hat, sondern es bietet auch die Möglichkeit zu überprüfen, ob es wirklich in der Datenbank verbucht worden ist. Dies ist etwas, was viele Testtools auf dem Markt nicht beherrschen. Sie decken häufig nur die Anwendungsoberfläche ab

- Es gibt viele Funktionalitäten, die über kein GUI verfügen. Um die Durchgängigkeit der Tests zu gewährleisten, lassen sich beispielsweise über eCATT auch Funktionsbausteine und BAPIs vollautomatisch testen.

- Testfälle können von Mitarbeitern angelegt werden, die keine Programmierkenntnisse haben und das in relativ kurzer Zeit. Dabei wird am Ende eines Testlaufs ein ausführliches Ergebnisprotokoll ausgeworfen. Dadurch haben auch Mitarbeiter ohne Programmierkenntnisse die Möglichkeit, nicht nur die Anwendungsoberfläche zu testen, sondern sogar tiefer ins System dabei zu gehen.

- eCATT wird von der SAP AG auch in Zukunft weiter entwickelt werden. D. h., dass noch weitere Funktionalitäten hinzukommen werden, die nicht selbst programmiert werden müssen.

- Die Komplexität und Planung mit eCATT Testszenarien zu automatisieren und abzudecken ist generell geringer als eine noch zu programmierende Softwarelösung zu analysieren und auf die eigenen Testzwecke abzustimmen.

- In eCATT lassen sich schnell und übersichtlich die betroffenen Programme, Dynpronummern bis hin zu den einzelnen Feldnamen auf Programmebene auslesen. So hat der Entwickler sämtliche Hinweise, die er braucht, um gezielt in der SE 80 suchen zu können.

3.3.3 Grundlagen von ausgewählten Testtool eCATT

eCATT steht hierbei für „extended Computer Aided Test Tool“ und ist ein Tool der SAP AG, das als Projekt Testtool in dieser Diplomarbeit ausgewählt wird.

Die Grundfunktionsweise lautet wie folgt: mit eCATT hat man eine Möglichkeit, Testfälle im Testpakete zu integrieren und diese ausgewählten Testern zuzuordnen. Über Testpläne kann man einzelne Testpakete überwachen, testen und auswerten. In der Testplanverwaltung kann man Testberichte im Microsoft Word generieren. Dazu General Methode wird Capture/Replay verwendet, dadurch Testskript wird erstellt.

Die eCATT stellt vier separate Objekttypen zur Verfügung: Die ersten drei Typen bilden die Grundlage eines Tests und der vierte kombiniert die anderen zu einem vollständigen Testfall. Testworkbench* steht dafür dass wie oder womit die Testfälle organisiert werden können.

Die folgende Abbildung fasst den Entwicklungsprozess zusammen und veranschaulicht, welche Rolle die unterschiedlichen eCATT Elemente (Abbildung 14) darin spielen.

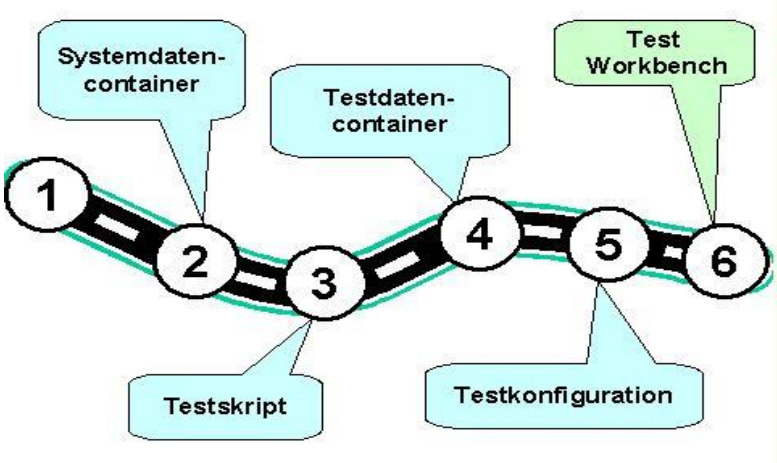


Abbildung 14: Übersicht eCATT-Objekte [SAP13a]

eCATT kann direkt aus der Baumstruktur im PDM System (siehe Abbildung 2) durch Eingabe der Transaktionscodes „SECATT“ in die Kommandozeile gestartet. Siehe dazu Abbildung 15 eCATT starten werden.

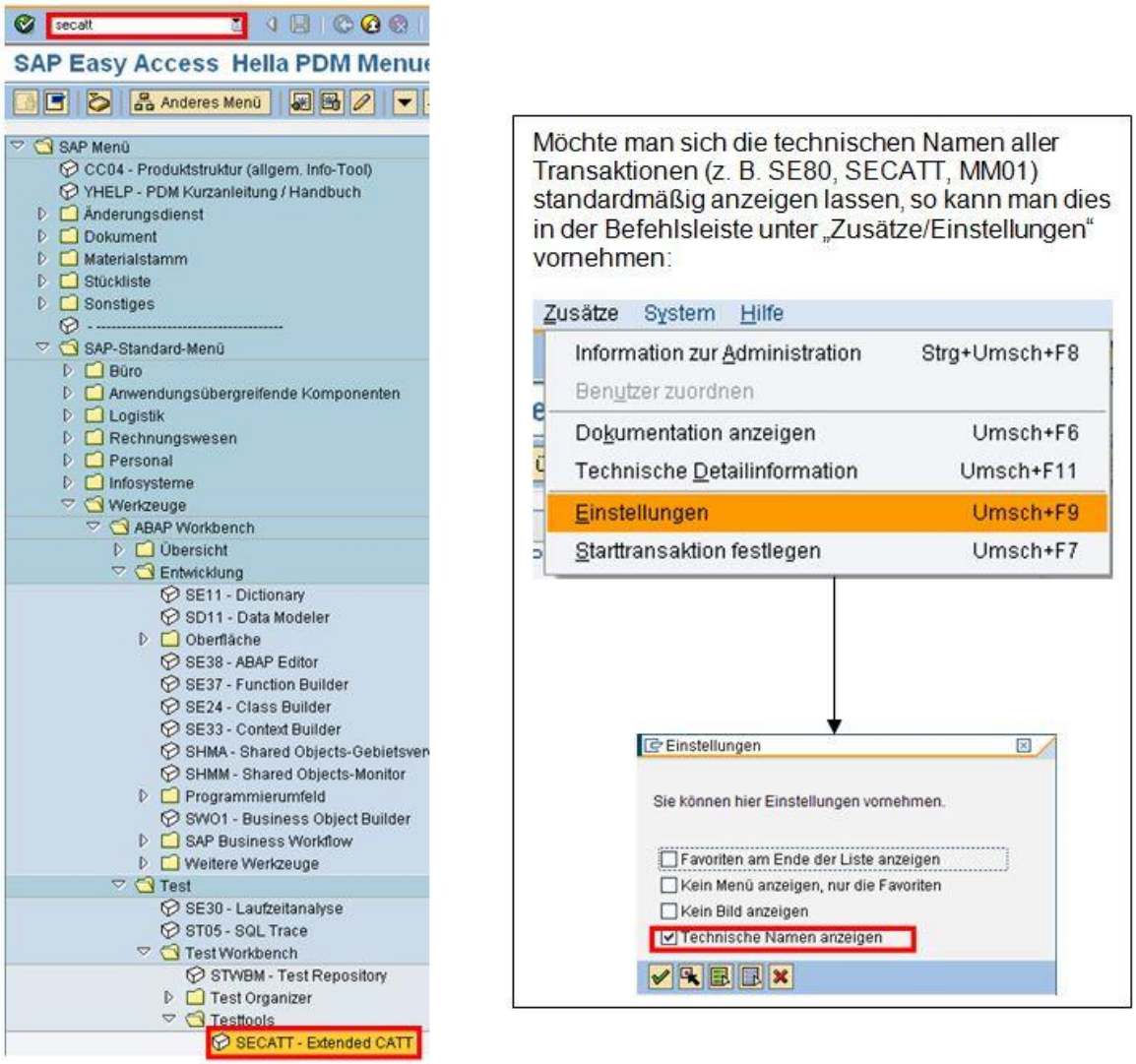


Abbildung 15: eCATT starten [Hell10b]

Dadurch gelangt man in die folgende Grundübersicht Abbildung 16 „eCATT Testtool“.

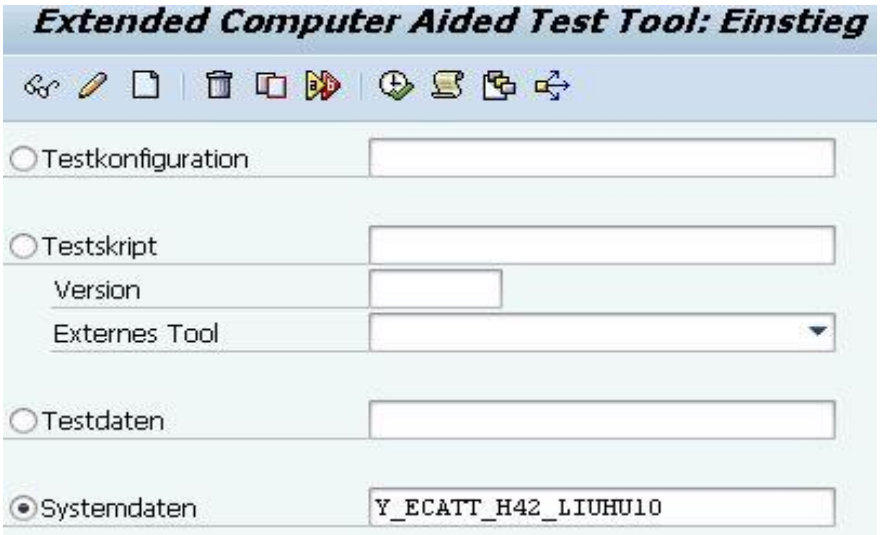


Abbildung 16: eCATT Testtool

In eCATT können die vier Grundelemente Systemdatencontainer, Testdatencontainer, Testskript und die Testkonfiguration angelegt werden. Diese vier Grundtypen werden im Einzelnen nachfolgend definiert:

Element 1: Systemdatencontainer

Der Systemdatencontainer stellt im weitesten Sinne einen Container dar, in dem man all seine Systeme definieren kann, mit denen man arbeiten möchte. Einen Systemdatencontainer braucht man für den Einsatz von eCATT aber nicht zwangsläufig. Dabei ist allerdings zu beachten, dass man ohne Angabe eines Systemdatencontainers an das System gebunden ist, in dem man seinen Testfall angelegt hat. Man könnte in diesem Fall aber im Nachhinein immer noch einen Systemdatencontainer angeben, um per RFC-Verbindung auf andere Systeme zugreifen zu können. eCATT werden also durch den Systemdatencontainer die einzelnen Systeme bekannt gemacht, mit dem es arbeiten soll.

Element 2: Testskript

Sämtliche Tests, die durchgeführt werden sollen, müssen in einem Testskript aufgezeichnet werden. Testskripte beinhalten generell die Aufzeichnung einer Transaktion und sämtliche Schnittstellen werden dabei von eCATT automatisch erzeugt. Die Testskripte müssen also nicht erst noch programmiert werden. Diese Arbeit nimmt eCATT dem Anwender ab und ist in etwa vergleichbar mit dem Makrorekorder in Microsoft Excel. Bei Ausführung von Testen werden also immer Testskripte irgendwo an einer Stelle gestartet.

Element 3: Testdatencontainer

Im Testdatencontainer kann der Anwender seine Testdaten für beliebige Testskripts hinterlegen. Der Testdatencontainer ist in beliebigen Testkonfigurationen, zu denen wir noch kommen werden, einsetzbar. Dabei ist zu beachten, dass der Testdatencontainer an sich nicht ausführbar ist. Er ist quasi nur der Behälter, der eine bestimmte Menge irgendwie gearteter Testdaten enthält. Wir werden an dieser Stelle einen Testdatencontainer anlegen, der verschiedene Nutzernamen enthält.

Der Testdatencontainer stellt Daten bereits für durchzuführende Testläufe zur Verfügung. Die Ausführung passiert dabei nicht hier, da der Testdatencontainer erst mal nur Testdaten enthält. Der Zugriff auf dem Testdatencontainer findet in der Testkonfiguration statt und er wird hier mit ausgeführt. An dieser Stelle soll es deshalb darum gehen, dass einen Testdatencontainer mit unterschiedlichen Nutzernamen angelegen werden, auf dem man später zurückgreifen wird, wenn man zum Thema Testkonfiguration kommt [Hell10b].

Element 4: Testkonfiguration

Eine Testkonfiguration bringt sozusagen die bereits bekannten Elemente zusammen. In der Testkonfiguration werden Systemdatencontainer, Testskript und Testdatencontainer zusammengeführt. Die Testkonfiguration kann anschließend mit allen referenzierten Elementen ausgeführt werden. Es kann an dieser Stelle auch der Testdatencontainer ausgeführt werden, der

als solches nicht selbstständig ausführbar ist. Testkonfigurationen können Testkatalogen und Testplänen innerhalb der Test Workbench zugewiesen werden.

4 Implementierung und Visualisierung

Für das Verständnis des innerhalb des erstellten Konzepts vorgestellten Testprozess zur Testautomatisierung im PDM System wurden die Einführung des Softwaretestens aus Kapitel 2 und Kapitel 3 vorausgesetzt. In den folgenden Abschnitten wird die Testsystem-Architektur bzw. Systemkontext vorgestellt. Eine exemplarische Konzeptentwicklung mit eCATT wird ebenso behandelt, darüber hinaus wird die Vorgehensweise zur Realisierung der Testautomation aufgezeigt.

4.1 Testsystem-Architektur

In dieser Diplomarbeit wird die Anwendung im PDM System intensiv getestet. Abhängig von der Größe eines Projektes werden die Prüfungen von individuellen Testteams geplant und realisiert. In den folgenden Abschnitten werden detaillierte Rollen mitsamt deren Aufgaben aus dem Testprozess aufgezeigt.

4.1.1 Geschäftsprozess der Angebotserstellung

„Das Vorgehen bei der Aufgabenanalyse hängt sehr stark von der Art der vorhandenen Vorgaben ab. Teilweise können die gleichen Techniken wie bei der Gewinnung von Anforderungen verwendet werden. Es geht darum, ein klares Verständnis zu erarbeiten, was verlangt wird und welche grundsätzlichen Lösungsmöglichkeiten in Frage kommen [MGSE02].“

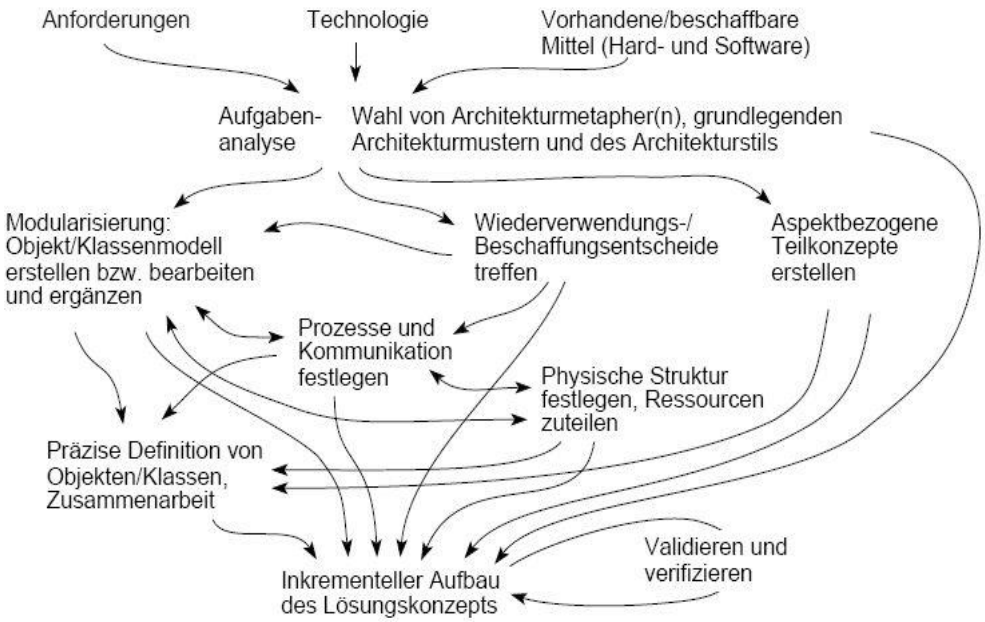


Abbildung 17: Zusammenhänge zwischen den Aufgaben des Architekturentwurfs

(am Beispiel eines objektorientierten Entwurfs) [Glin02]

Die obere Abbildung 17 zeigt die Zusammenhänge zwischen den Aufgaben des Architekturentwurfs.

Das PDM dient der strukturierten und konsistenten Verwaltung aller produktbeschreibenden Daten zur Unterstützung der Engineering-Prozesse bei Firma HELLA KGaA Hueck & Co. Sowie ist in Abbildung 1 gefasst. Das PDM-System (Abbildung 2) umfasst folgende Kernfunktionen:

a) Verwaltung von Produktdaten

- Administration von Entwicklungs- und Konstruktionsergebnissen
- Funktionen zur Klassifizierung und Abbildung von Nummernsystemen
- Produktstrukturmanagement

b) Prozessmanagement

- Steuerung von Geschäftsprozessen
- Abbildung ablauforganisierte. Merkmale, z.B. Freigabestatus, Version oder Gültigkeit

c) Allgemeine Systemfunktionen

- Z.B. Benutzer- und Zugriffsverwaltung

d) multifunktionaler Arbeitsplatz, kombiniert u.a.

- Stücklistenverwaltungssystem
- Digitale Archive
- CAD- und sonstige Entwicklungswerkzeuge

In dieser Diplomarbeit werden die Haupt-Testprozessen zur Prüfung der Funktionalitäten (in die Allgemeinen Systemfunktionen) im Bereich Dokumenteninfosatz (Abbildung 2) im PDM System eingesetzt.

Der Dokumentinfosatz (DIS) enthält die beschreibenden Daten zu einem Dokument und bildet das Bindeglied zur Datenablage. Der Dokumentinfosatz ist quasi wie die Karteikarte einer Bibliothek zum Buch. Auf ihr sind alle relevanten Informationen zum Buch bzw. Dokument und dessen Inhalt vermerkt. Siehe Abbildung 4 „Dokumentinfosatz“.

Dabei handelt es sich um PDM-Basis-Funktionalitäten, die für das Anlegen, Ändern, Suchen, Einchecken und Ausschecken von Dokumenten erforderlich sind. Einzelne Abläufe können abhängig von Dokumentart und -inhalt anders gestaltet sein. Die Dokumentart fasst die Dokumente zusammen, die einen ähnlichen Lebenszyklus haben. Nach dem Anlegen eines Dokuments wird automatisch eine Dokumentnummer vom System generiert, die aus bis zu 25

Ziffern besteht. Mit dieser Nummer kann man das entsprechende Dokument wieder finden um es anzuzeigen.

Die folgende Abbildung 18 zeigt die Basis-Funktion „Dokument anzeigen“.

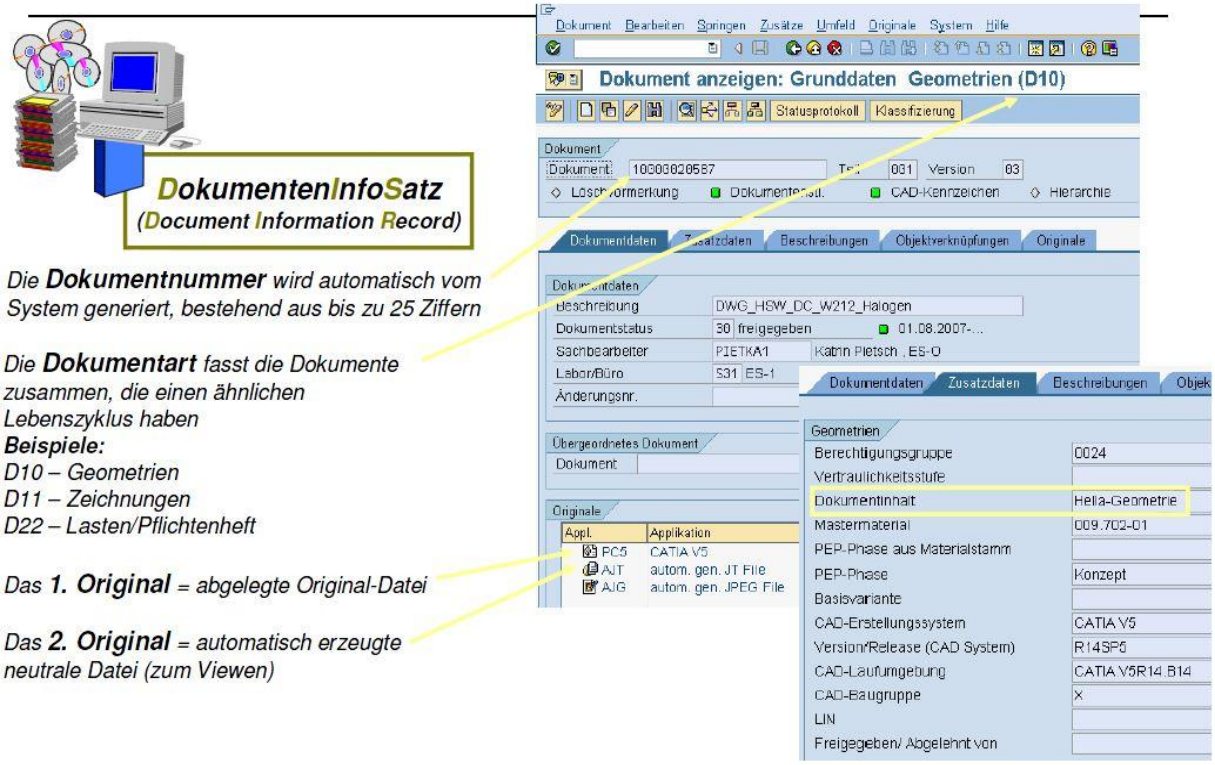


Abbildung 18: Dokument anzeigen [Hell10a]

4.1.2 Darstellung der Testsysteme und deren Test Technologie

Die PDM System Umgebung bei HELLA, siehe die folgende Abbildung 19 im rechten Block, enthält die Firmen-Spezifische Anpassung die der neu entwickelten Funktion von HELLA entspricht. Die Ausführung der Tests erfolgt zunächst im Entwicklungssystem (H42). Erst nachdem die Modifikationen in das nächsthöhere System transportiert wurden, erfolgt die Durchführung im Testsystem (H45) bzw. Produktivsystem (H51). Die Ausführung durchläuft somit die Systemstrukturen parallel zu den Entwicklungsphasen [Meie12].

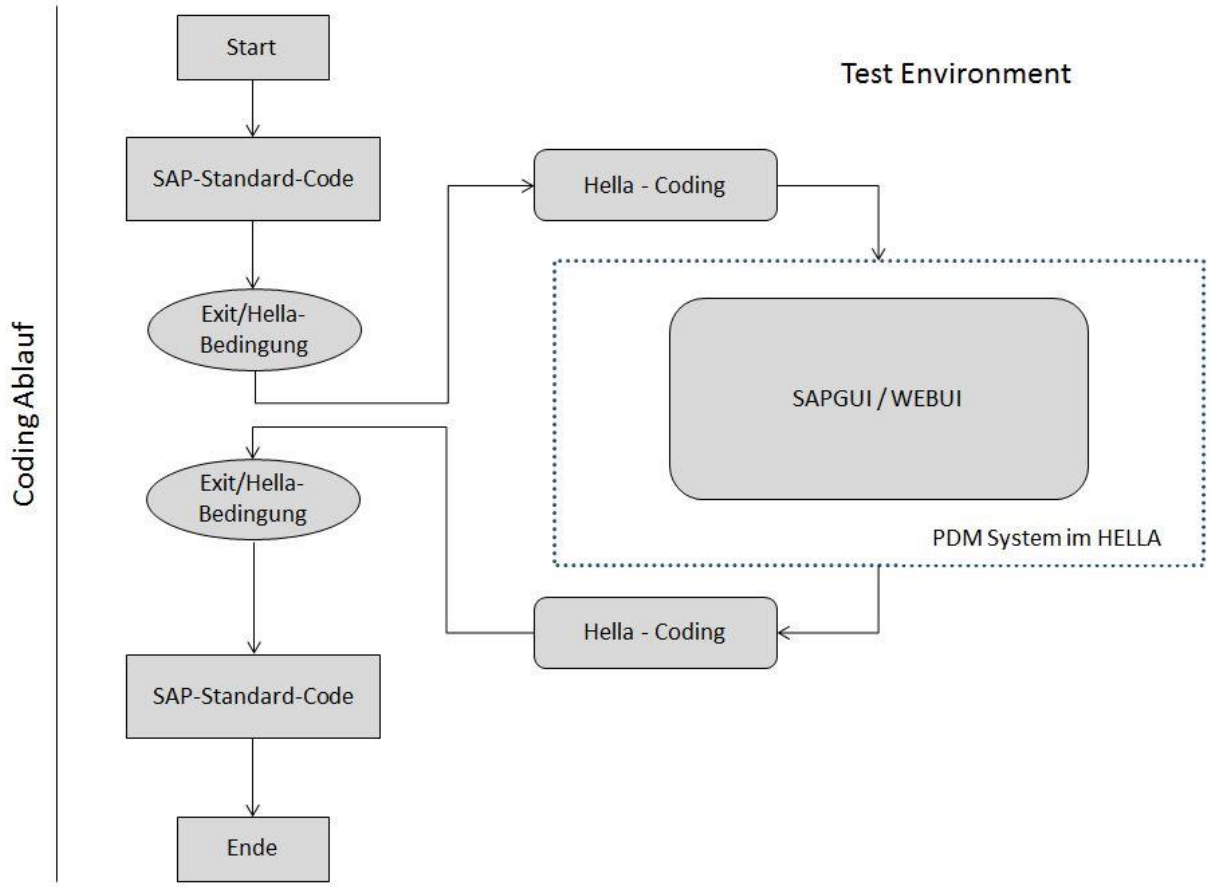


Abbildung 19: Test Environment bei HELLA

Das PDM System als Testsystem besteht aus zwei Versionen, SAPGUI und WEBUI bei HELLA, die WEBUI basiert auf HELLA Eigenentwicklung und wurde neu erweitert, sie ist eine neue Oberfläche für das SAP PLM. Der Netweaver Business Client (NWBC) [PLMS12] wird den Zugang zum SAP PLM über den SAPGUI ablösen. Jedoch ist der NWBC nicht nur eine neue Oberfläche. Er ist auch ein erster Schritt auf dem Weg zu einer integrativen Benutzeroberfläche, dadurch können die Informationen aus unterschiedlichen Quellen gebündelt dargestellt werden. Auf Grundlage des NWBC werden Schritt für Schritt neue Funktionen entwickelt und zur Verfügung gestellt.

Der SAP Solution Manger wird bei HELLA aus zwei Gründen nicht eingesetzt, er ist erstens nicht anpassbar wegen der HELLA Eigeneentwicklung (Schnittstelle), und zweitens wegen hoher Lizenzkosten für Komplette Softwarelösung-Paket. Dafür sucht die Firma HELLA eine individuelle Lösung. Das Testtool eCATT wurde als ein Testautomatisierungswerkzeug ausgewählt, dazu in dieser Arbeit wird ein Testkonzept und eine entsprechende Vorgehensweise erstellt.

extended Computer Aided Test Tool (eCATT) wird für das Anlegen und Durchführen von Funktionstests von PDM System aus SAP bei HELLA verwendet. Es soll vorrangig dem automatischen Testen von SAP-Geschäftsprozessen dienen. Jeder Test legt ein detailliertes Protokoll an, in dem der Testverlauf und die Ergebnisse dokumentiert werden [SAP13b].

eCATT ermöglicht das automatischen Test in SAP GUI for Windows und SAP GUI for Java. eCATT kann mit der Testworkbench verwendet werden [SAP13b]. Sie können die Testfälle von Computer Aided Test Tool (CATT) migriert werden, um die besseren Funktionen von eCATT zu nutzen. eCATT ist auch in dem Objekt Navigator (Transaktionscode: SE80) integriert.

Sie haben folgende Möglichkeiten [SAP13b]:

- Testen von Transaktionen, Berichten und Szenarios
- Aufrufen von BAPIs und Funktionsbausteinen
- Testen von entfernten Systemen
- Überprüfen von Berechtigungen (Benutzerprofilen)
- Testen von Aktualisierungen (Datenbank, Anwendungen, Benutzeroberfläche)
- Testen der Auswirkungen von geänderten Customizing-Einstellungen
- Prüfen von Systemnachrichten

4.2 Konzeptentwicklung zur Testautomatisierung

In dieser Diplomarbeit wird ein Modellbasierter Test zur Testautomatisierung durchgeführt. Ein Modellbasierter Test bedeutet, dass der Kern der Arbeiten bzgl. Testplanung und Testspezifikation an einem grafischen Modell des zu testenden Systems stattfindet. Und das Testfälle, Testfallskripte für Testautomationstools sowie der größte Teil der Testdokumentation automatisch aus diesem zentralen Modell generiert wird. Dazu besteht eine Werkzeugkategorie aus Testplanung und –Management, Testanalyse, Testspezifikation und Testdesign, Testdurchführung, Aufzeichnung dynamischer Tests, Testobjektanalyse und Teststatus-Auswertung.

Im Allgemeinen besteht Softwaretesting aus grundsätzlich 5 Test-Schritten, die sind jeweils [Meis12]: Vorstufe-Try Prinzip, Manuelles Testen mit Testplan, Unit-Tests, Automatisierte Anwendungstests und Tests im verteilten System.

Bei der Testautomatisierung steht die Qualität der Anforderungen für die Effizienz der Automatisierung. Es ist nötig die Anforderungen von Beginn an qualitativ vorzulegen. Sie müssen dementsprechend konsistent, vollständig und insbesondere prüfbar sein [SBMB11] Testautomatisierung ist ein mächtiger Schritt und wird durch Testtool unterstützt, um Tests wiederholbar zu machen und effizienter zu gestalten. Diese Diplomarbeit wird eine Testautomatisierung mit Fokus auf den funktionalen Tests im PDM System aus SAP bei HELLA konzipieren, der Testprozess wird anhand des Testtools eCATT implementiert und entsprechende Testfallspezifikationen durchgeführt. Eine formale Spezifikation für analysierte Testfälle und eine Automation-Modell werden für zukünftige Arbeit sichergestellt, damit

werden die Tests regelmäßig durchgeführt, und sind auch nach Änderung im existierenden Testprozess möglichweise einfach zu pflegen.

Dieses systematische Testkonzept ist auf Speziallisten durchgeführt:

a) Test ist geplant, eine Testvorschrift liegt vor

b) Das Test Tool wird gemäß Testvorschritt – der Testspezifikation – ausgeführt

c) Ist-Resultate werden mit Resultaten verglichen.

d) Testergebnisse werden dokumentiert

e) Fehlersuche- und Behebung erfolgen separat

f) Nicht bestandene Tests werden wiederholt

g) Test endet, wenn vorher definierte Test Ziel erreicht sind

Die Testspezifikation wird laufend aktualisiert auch mit dynamische Ausführungszeit bzw. dazu gehörige Protokoll Nummer.

Auf diese Konzeptionsschritte wird im folgenden Testprozess detailliert eingegangen.

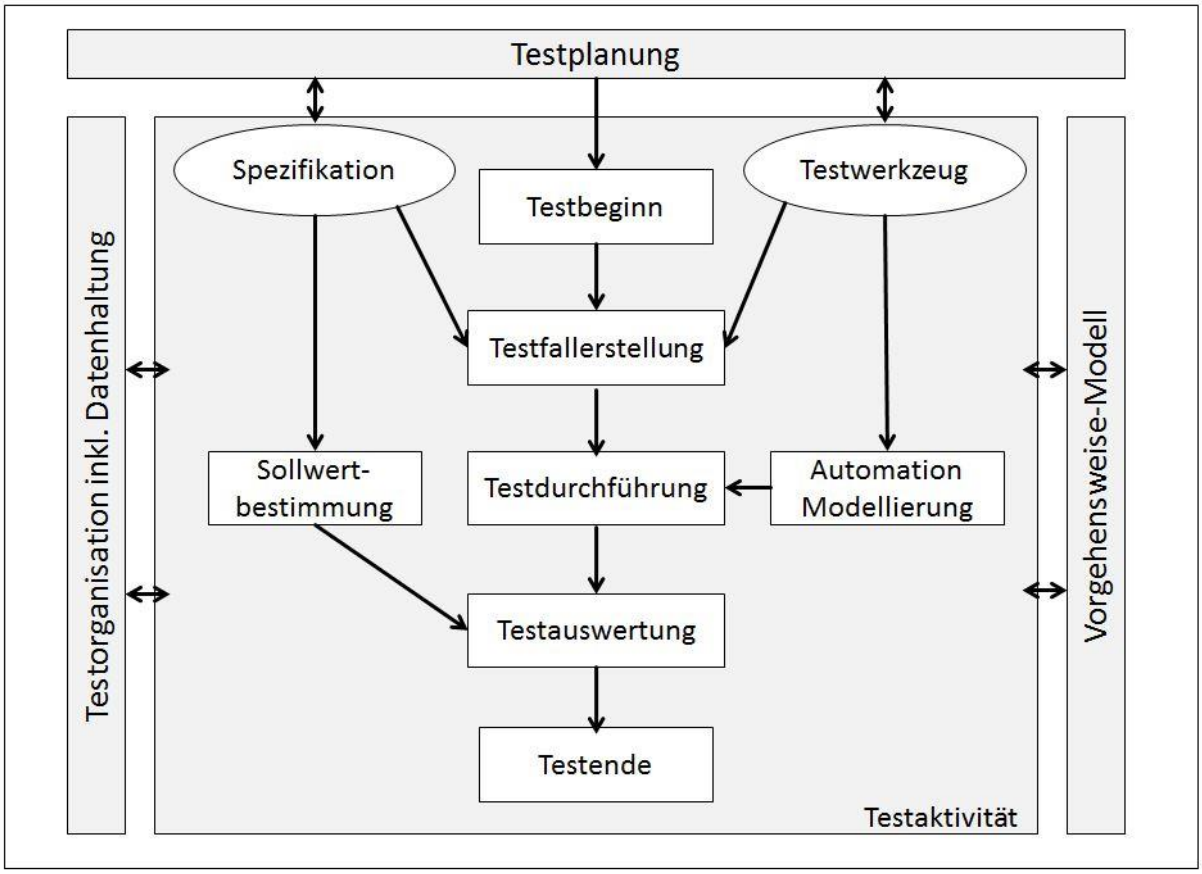


Abbildung 20: Ablauf eines Testprozesses (Erweiterung von [Koch03])

Der Testprozess besteht aus Testplanung, Testorganisation und entsprechende Testaktivität. Wie in Abbildung 20: Ablauf eines Testprozesses (Erweiterung von [Koch03) dargestellt, wird zunächst die Testplanung durchgeführt, während die Testorganisation, Testdatenhaltung und Testvorgehensweise parallel zu den eigentlichen Testaktivitäten stattfinden. Die konkrete Vorgehensweise wird im nächsten Kapitel 4.3 mit dem Workflow-Exemplar in Praxis PDM Testsystem H42 am Beispiel HELLA Testsystem-umgebung implementiert und ausgewertet.

4.2.1 Testplanung und Testorganisation

Die Testplanung ist ein sehr wichtiges Element des Testprozesses, da es die Grundlage für alle anderen Phasen des Testprozesses bildet. Hier wird bestimmt, wann man womit beginnen kann. Dies bezieht sich sowohl auf die Testentwicklung als auch auf die Durchführung, Testorganisation etc.

Testorganisator wird bei der ständigen Anpassung des Systems an die Kundenanforderungen durch Customizing, Modifikation, Release-Upgrade und Eigenentwicklung fortgesetzt. Mit Organisator wird Testkatalog, Testplan und Testpaket erstellt und entsprechende Tests nach Anforderung ausgewertet [Koch03].

Um sie auf Testaktivität zu unterstützen, liefert SAP das Test Tool eCATT. [SHSM13] In der folgenden Abbildung 21 Strukturen des Testorganizers sind die einzelnen Komponenten des Test Organizers aufgeführt:

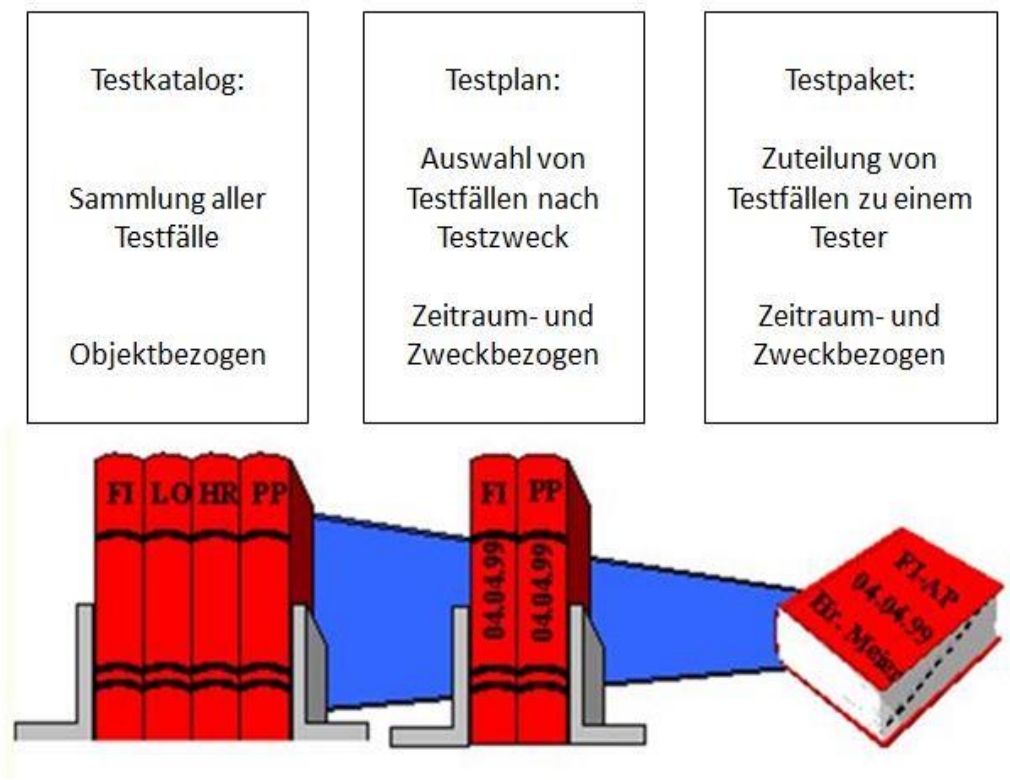


Abbildung 21: Struktur des Testorganizers [SHSM13]

Zur Organisation und Planung von Testfällen bietet das Testtool eCATT die beiden Komponenten Testkatalog und Testplan an.

Testdatenhaltung meint die Speicherung der Testdaten. Testdaten wird in Change Request Management gesammelt und auch von Kunde-Feedback-Seite berücksichtigt, die bestehen aus Testfällen und Test-Umgebung. Alle Testfälle stehen im Moment in der Microsoft Excel-Tabelle nach dem fachlichen Verantwortungsbereich DOK, MAT, PreBOM aufgeteilt. Die Test-Umgebung entspricht dem System-Mandanten zur Ausführung der Tests. D.h. die Tests werden zunächst im Entwicklungssystem ausgeführt, nach Modulierung in das nächste Testsystem transportiert, dann erfolgt die Durchführung im Produktivsystem bzw. Schulungssystem. Die Ausführung durchläuft somit die Systemstrukturen parallel zu den Entwicklungsphasen. Sie wird als H42, H45 und H5E im PDM System unter SAPGUI bei HELLA bezeichnet.

Für die existierenden Testdaten für diese Diplomarbeit siehe folgende Abbildung 22 „Testdaten über Detail der Funktionalität in PDM System“.

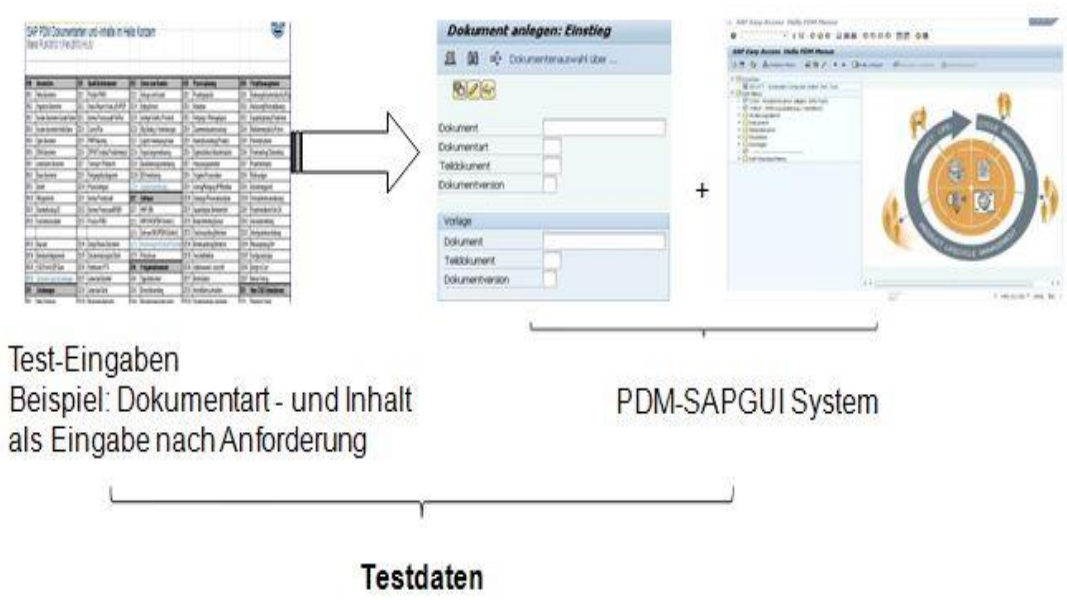


Abbildung 22: Testdaten über Detail der Funktionalität in PDM System

Die Testdaten werden auf sowohl funktionale als auch nichtfunktionale Anforderungen betrachtet. Als funktionaler Test [HLT06] wird man hier die Laufzeit des Programms nicht berücksichtigen, sondern es wird nur die funktionale Korrektheit überprüft. Nichtfunktionale entspricht hier den Testfällen über die Funktionsweise-Feindlichkeit im PDM System. Ein Testfall ist ein Auflisten in Form von vorliegenden Microsoft Excel-Tabellen von Eingabe und erwarteten Ergebnissen, sowie die Beschreibung des Tests. Dabei sollte darauf geachtet werden, dass jeder Testfall zu einem unterschiedlichen Ablauf führt bzw. eine andere Funktion des Systems testet. Siehe die Abbildung 23 eine Testfall-Beschreibung über Merkmale im PDM System und Abbildung 24 Ein Testfall über Merkmal Prüfung bei Status-Wechseln in PDM System.

System:	**Objekt:**
Mandant:	**Hardware:**
UserID:	**Betriebssystem:**
Datum:	**CAD System:**
Name:	**DIS des Protokolls:**

Funktion	Test erforderlich	Trans-aktion	Hinweise / Ablauf	Erwartetes Ergebnis [Fehlermeldungen / Hinweismeldungen]	iO / niO	Fehlerbeschreibung [falls n.ok]	Bemerkungen
Dokumentinhalt-abhängige Merkmale							
Funktion "Dokument anlegen"		CV01N	Dokumentart D10 auswählen	Es werden folgende Merkmale unter den Zusatzdaten angezeigt: - Berechtigungsgruppe - Vertraulichkeitsstufe - Dokumentinhalt - Mastermaterial - PEP-Phase aus Materialstamm - PEP-Phase (erst nach Speichern des DIS sichtbar) - Musterstand Hella - Basisvariante - CAD-Erstellungssystem - Version/Release (CAD-System) - CAD-Laufumgebung - CAD-Baugruppe - LIN - Freigegeben/Abgelehnt von			
Dokumentinhalt eintragen			Dokumentinhalt D10.6	Es werden zusätzlich noch die folgenden Merkmale angezeigt: - Simulationstyp			
Dokumentinhalt eintragen			Dokumentinhalt D10.9	Es werden zusätzlich noch die folgenden Merkmale angezeigt: - Simulationstyp			
Funktion "Dokument anlegen"		CV01N	Dokumentart D11 auswählen	Es werden folgende Merkmale unter den Zusatzdaten angezeigt: - Berechtigungsgruppe - Vertraulichkeitsstufe - Dokumentinhalt - Mastermaterial - PEP-Phase aus Materialstamm - PEP-Phase (erst nach Speichern des DIS sichtbar) - Musterstand Hella - Erstellt am			

Merkmale | Statuswechsel | Neue Version | Berechtigung | Prüfungen | Dokumentsuche | Sonstiges | Screenshots

Abbildung 23: Eine Testfall-Beschreibung über Merkmale im PDM System

System:	**Objekt:**
Mandant:	**Hardware:**
UserID:	**Betriebssystem:**
Datum:	**CAD System:**
Name:	**DIS des Protokolls:**

Funktion	Test erforderlich	Trans-aktion	Hinweise / Ablauf	Erwartetes Ergebnis [Fehlermeldungen / Hinweismeldungen]	iO / niO
Einchecken - Auschecken					
Einchecken (Status 20 - 21)			DIS unter Dokumentart D21 anlegen und speichern	DIS wird im Status 20 angelegt.	
			Status auf 21 ändern	Nach dem Speichern wird die Datei automatisch in der Datenbank abgelegt (Schloss zu)	
				Die Datei wird am lokalen Ablageort gelöscht	
			Datei in der Anwendung geöffnet lassen	Die Datei kann nicht gelöscht werden und es kommt die Fehlermeldung: Datei ... kann auf dem Frontend nicht gelöscht werden.	
			Datei mit dem Button unterhalb der Originale ablegen	Die Datei wird abgelegt (Schloss zu), der Status aber nicht umgesetzt	
Zwischenstand erzeugen (Status 95)			Status auf 95 ändern und mit Enter bestätigen	Es öffnet sich ein Protokollfeld, welches gefüllt werden muss (z.B. 1. Überarbeitung). Ein Überspringen mit Enter ist nicht möglich.	
			DIS speichern	Nach dem Speichern hat der DIS wieder den Status 21	
			Statusprotokoll anzeigen	Zwischenstand wird angezeigt	
Auschecken (Status 21 - 20)			Original selektieren und mit dem Button "Bleistift" unterhalb der Originale in Arbeit nehmen	Zielverzeichnis wird vorgeschlagen	
				Die Datei wird im Zielverzeichnis abgelegt und automatisch in der Applikation geöffnet.	
				Der DIS wurde automatisch auf Status 20 gesetzt und das Schloss ist offen	
Auschecken mit anderem Sachbearbeiter			Original selektieren und mit dem Button "Bleistift" unterhalb der Originale in Arbeit nehmen	Der DIS wurde automatisch auf Status 20 gesetzt, das Schloss ist offen und der Sachbearbeiter wurde geändert.	
Einchecken (Status 20 - 21)			Datei ändern, speichern und wieder einchecken	Auf dem Reiter "Originale" taucht der Zwischenstand unterhalb der Originaldatei auf (kleines Dreieck öffnet die Struktur) - grüner Punkt: aktuelle Datei - gelbes Dreieck: Zwischenstand	
Automatische Erzeugung des neutralen Format (Status 21 - 25)			Status von 21 auf 25 ändern	Es öffnet sich ein Protokollfeld, welches mit Enter übersprungen werden kann.	
				Nach einiger Zeit (ca. 30 sec.) wird das neutrale Dokument (z.B. PDF) erzeugt und als 2. Original dem DIS hinzugefügt (während der Erzeugung ist der DIS gesperrt).	
				Der Eintrag aus dem Protokollfeld (z.B. 1. Review) wird im Statusprotokoll	

Abbildung 24: Ein Testfall über Merkmal Prüfung bei Status-Wechseln in PDM System

Um einen effizient Testprozess zu gewährleisten, müssen die komplett Testdaten zuerst nach einer Statistische Risikoanalyse Schrittgemäß dem ausgewählte Testtool durchgeführt werden, sowie der Planung für die manuellen Tests und automatischen Test. Oder ein manueller Test, als wäre es automaischer. Eine nachträgliche Automatisierung der Tests ist dann natürlich mit geringerem Aufwand durchführbar.

4.2.2 Testaktivität

Bei den Testaktivitäten werden die Testfälle entwickelt, durchgeführt und ausgewertet, siehe Abbildung 20 in der Mitte Block-Testaktivität. Die Grundlage bildet hierfür neben der Testplanung und –Organisation auch die Spezifikationen und das Testwerkzeug. Auf die einzelnen Phasen der Testaktivitäten wird im Folgenden eingegangen. Aber zuvor sei noch etwas zur Automatisierung der Testaktivitäten gesagt.

Automatisierung der Testaktivitäten ist für das Testen sehr wichtig. Sie soll nach vordefinierter Metrik sicherstellen, dass die Tests regelmäßig (auch nach jeder Änderung im Code) durchgeführt werden. Wenn der Entwickler beim Testen nur wenig zu tun hat im Idealfall nur Anstoß der Tests und Angucken der **Auswertungstatistik** werden die Tests auch häufiger durchgeführt. Zudem erspart es eine Menge Zeit. Denn ein Mensch benötigt im Vergleich zu einem Computer bei weitem mehr Zeit, um dieselben Ergebnisse zu erzielen.

Die automatische Testfallermittlung ist generell möglich. Jedoch ist sie mit viel Aufwand verbunden, da es nötig ist, dass eine formale **Spezifikation** des Systems und ein **Automation Modell** des Systems vorliegen. Auf dessen Grundlage können dann die Testfälle werkzeuggestützt automatisch erstellt werden und durchgeführt werden.

Bevor man mit dem Automatisierungstest anfängt, muss man schon eine Aussage treffen ob es sich überhaupt lohnt den Test zu automatisieren. Darauf folgend wird mit einer mathematischen Methode ein statischer Grenzwert analysiert. Die vorgestellten Konstanten sind aus der SAP-Standard Norm.

4.2.3 Methode zur Analyse der Testdaten

Für ein qualitatives Vorgehen benötigt man qualitative Testdaten, dazu werden die erhaltenen Messwerte statistisch mit analysiert und verarbeitet. In diesem Kapitel werden die Bestimmungsmethoden zur Analyse der Testdaten implementiert. Diese entspricht der mathematischen Methode, dem priorisierungsverfahren bzw. der Grundidee des Textautomaten.

Methode 1: für Statistische Grenzwertanalysen

Viele Tests werden nicht nur einmal sondern viele Male im Laufe des Produktlebenszyklus durchgeführt. Zur Unterstützung der Bestimmung wird folgende Fragenstellung aus Wirtschaftlichkeit „effizienter“ Sicht vorgestellt.

a) Was ist zu beachten, bevor der Test gestartet wird?

Die Entwicklung vom automatisierten Testskript verursacht Kosten. Diese Kosten werden durch das extensive und häufige Testen der aufgezeichneten Prozesse gedeckt. Darüber hinaus ist es sinnvoll zu herauszufinden was zu automatisieren und was im Manuell Test günstig sind.

Kosten: bei der Statischen Analyse entsprechen die Kosten dem Zeitaufwand zur Prüfung der aufgebauten Struktur und Semantik. (hier z.B. die Aufzeichnung der Standardprozedur eines Moduls. Diese Standardprozedur ist Teil der PDM aus SAP. Spezielle Einstellungen, die nur für einen Kunden gemacht werden, eignen sich in der Regel nicht automatisiert zu werden, d.h. es wird manuell getestet und dies ist günstiger als Automation).

b) Aufwandsvergleich

Es wird festgelegt welche Testfälle/Tests/ Testablauf sich lohnen automatisiert zu werden. Hier wird zwischen Automatisierter Fall und Manueller Fall unterschieden.

Die Ausführung eines manuellen Testfalles erfordert etwa 10 bis 30 Minuten [SpLi09] in Abhängigkeit der Komplexität des Testfalles. Diese Zeitspanne bezieht sich auf die erfolgreiche Ausführung ohne Auftreten von Fehlern. Sollte ein Fehler auftreten, erhöht sich die für diesen Testfall erforderliche Zeit sofort. Der Fehler muss dokumentiert und zum entsprechenden Entwickler weitergeleitet werden. Sobald der Entwickler den Fehler behoben hat, kann der Testfall erneut durchgeführt werden.

Die Ausführung eines automatisierten Testfalles erfordert meist weniger als 5 Minuten [SpLi09], gleichzeitig können mehrere Testfälle kombiniert werden. Das bedeutet einen hohen zeitlichen Vorteil, wenn die Testfälle automatisiert ausgeführt werden. Tritt ein Fehler auf, dokumentiert dies das Capture and Replay Methode automatisch im entsprechenden Protokoll. Nach Korrektur des Fehlers durch den Entwickler kann das Testskript bezogen auf das betroffene Modul erneut ausgeführt werden.

Neben dem zeitlichen Vorteil bei der Ausführung der Testfälle muss auch die Entwicklung der Automation betrachtet werden. Die Entwicklung eines Testskripts muss geplant werden. Als Basis für den Plan können die manuell ausgeführten Testfälle dienen, die in vorhergehenden Testprojekten erstellt wurden. Darüber hinaus ist es wichtig zu wissen, welche Eingangs-und Ausgangsparameter [SpLi09] für die Skripte definiert werden müssen. Die folgende Tabelle zeigt eine Aufwandsschätzung für einen Testfall.

Aktivität	manuell	automatisiert
Entwicklung(Planung, Aufnahme, Parametrisierung, Modularisierung, Testdaten)	-/-	4h(240 Minuten) *
Ausführung	10-30 Minuten	< 5 Minuten
Behandlung von Fehlern	15 Minuten	< 5 Minuten
Ausführung (Regressionstest)	10-30 Minuten	< 5 Minuten

Tabelle 1: manuell vs. Automatisiert [SpLi09]

4h (240 Minuten)*, diese Zeitaufwand für die Entwicklung ist die Durchschnittszeit einer Person, die bisher wenig Erfahrung mit der Testautomatisierung hat. Für einen erfahrenen Testautomatisierer erfolgt die Entwicklung wesentlich schneller [SpLi09]

Zu der Gewinnung eines Testfalls mit konkretem Analysewert hier wird eine Beispielrechnung angeführt. Angenommene Konstant Kosten-Zeitaufwand sind aus SAP Standard Press.

Definition:

- **p** steht für die Anzahl der Projekte oder entsprechende Anzahl der Testsystemen, in denen die Testfälle genutzt werden
- **w** steht für die Anzahl an Wiederholungen eines Testfalls innerhalb eines Projekts
- **f** steht für die Menge der Fehler, die während der Testfallausführung auftreten (f<w)
- als durchschnittliche Ausführungszeit für einen manuellen Testfall werden 20 Minuten angenommen

$$\mathbf{p\ (20\ w+15\ f) > p\ (5\ w + 5\ f) + 240}\ \text{[SpLi09]}$$

Um die automatisierten Testskripte auch in anderen Projekten nutzen zu können, ist ein entsprechender Anpassungsaufwand vorzusehen. Für das erste Projekt ist dafür nichts vorzusehen, daher wird der Faktor n-1 vorangestellt. Die Zeit für Anpassung wird mit 10 Minuten angenommen.

$$\mathbf{p\ (20\ w+15\ f) > p\ (5\ w + 5\ f) + 240 + (p - 1)\ 10}\ \text{[SpLi09]}$$

Als Erstes ist es interessant herauszufinden, in wie vielen Projekten der Testfall genutzt werden muss, um den Automatisierungsaufwand zu rechtfertigen.

$$\mathbf{p > 230 / (15\ w + 10\ f - 10)}$$

Kalkulation I auf Basis einer Entwicklungszeit von 240 Minuten und einer Dauer für Anpassungen von 5 Minuten.

f, w (f < w)	2	3	4	5	6	7
0	11,5	6,57142857	4,6	3,53846154	2,875	2,42105263
1	7,66666667	5,11111111	3,83333333	3,06666667	2,55555556	2,19047619
2		4,18181818	3,28571429	2,70588235	2,3	2
3			2,875	2,42105263	2,09090909	1,84
4				2,19047619	1,91666667	1,7037037
5					1,76923077	1,5862069
6						1,48387097
7						

Tabelle 2: Kalkulation I auf Basis einer Entwicklungszeit von 240 min

Kalkulation II auf Basis einer Entwicklungszeit von 180 Minuten und einer Dauer für Anpassungen von 5 Minuten.

f, w (f < w)	2	3	4	5	6	7
0	8,5	4,85714286	3,4	2,61538462	2,125	1,78947368
1		3,77777778	2,83333333	2,26666667	1,88888889	1,61904762
2		3,09090909	2,42857143	2	1,7	1,47826087
3			2,125	1,78947368	1,54545455	1,36
4				1,61904762	1,41666667	1,25925926
5						
6						
7						

Tabelle 3: Kalkulation II auf Basis einer Entwicklungszeit von 180 min

Kalkulation III auf Basis einer Entwicklungszeit von 120 Minuten und einer Dauer für Anpassungen von 5 Minuten.

f, w (f < w)	3	4	5	6	7
0	3,14285714	2,2	1,69230769	1,375	1,15789474
1	2,44444444	1,83333333	1,46666667	1,22222222	1,04761905
2	2	1,57142857	1,29411765	1,1	0,95652174
3		1,375	1,15789474	1	0,88
4			1,04761905	0,91666667	0,81481481

Tabelle 4: Kalkulation III auf Basis einer Entwicklungszeit von 120 min

Die Anzahl der Projekte nimmt mit zunehmender Wiederholungsfrequenz ab. Diese Tabelle ist ein direkter Vergleich zwischen manuell und automatisiert ausgeführter Testfällen. **Je häufiger ein Testfall genutzt wird, desto mehr lohnt es sich ihn zu automatisieren.** In den meisten Fällen wird ein Testfall für ein Modul nicht nur für den Modultest, sondern auf für den Systemtest, Integrationstest und den Abnahmetest genutzt. Das bedeutet, dass die Anzahl der Läufe steigt. Darüber hinaus läuft das Skript, nachdem Änderungen im System durchgeführt wurden, beispielsweise wenn neue Features freigegeben wurden [SpLi09].

Zusammenfassung: mit dieser Statistische Grenzwertanalysen Methode wird jeder Testfall aus Testdaten genau über Automation oder Manuell Test entscheidet. In Vorgehensweise-Modell sieht man konkrete Auswertung der automatisierbare Testfälle.

zu Ziel von Software Qualität Aussagen: Das Design der Testfälle ist durchzuführen, unabhängig davon, ob sie manuell oder automatisiert ausgeführt werden. Einer der wichtigsten Schritt beim Testen im PDM System aus SAP ist der Test der Anforderungen, also dessen, was der Kunde bzw. Auftraggeber bestellt hat. Hier existiert eine direkte Abhängigkeit: Die Qualität der Testfälle hängt von der Qualität der Anforderungen ab. Siehe Abbildung 25 Zusammenspiel zwischen Anforderungen und Testfall.

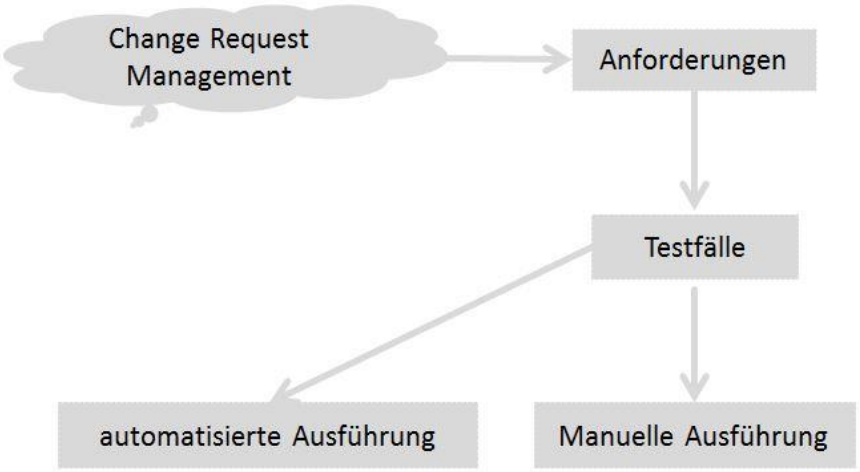


Abbildung 25: Zusammenspiel zwischen Anforderungen und Testfälle

Die untenstehende Abbildung 26 „Testfälle Analyse Ablauf“ zeigt: Die Testfälle aus dem Anforderungsbereich werden mit initialisierte Wert in einer Anweisung „Wiederholbar“ einfließen (zum Beispiele die Wiederholungsanzahl der Testfällen und Projektanzahl). Diese Anweisung ist eine einseitige Auswahl, die prüft, ob Testfälle wiederholbar sind. Falls dies der Fall ist, wird für wiederholbare Testfälle über Automatisierbares Testen entscheidet. Die Entscheidungsmethode wird durch mathematische Analyse durchgeführt. Der Manuelle Zeitaufwand kann durch die Formel **p (20 w+15 f)** ausgerechnet werden und der Zeitaufwand für den Automatisierungstest durch die Formel **p (5 w + 5 f) + 240 + (p - 1) 10**. Nach dem Vergleichen sieht man die Zeitersparnis.

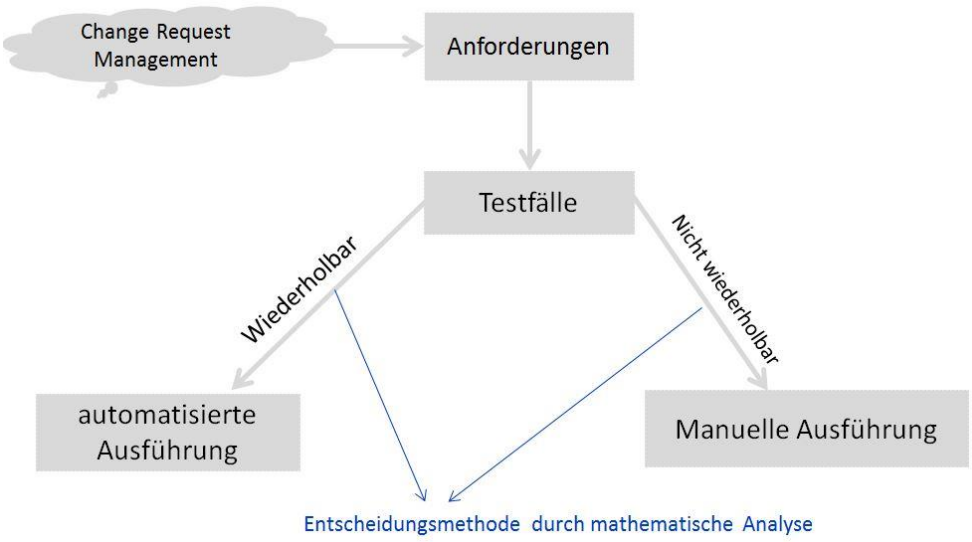


Abbildung 26: Testfälle Analyse Ablauf

Die analysierten Testfälle sowie der Ablauf aus den einzelnen Testfällen werden in der eCATT-Skriptsprache gewonnen. Die eCATT-Skripte ermöglichen eine wiederholte Durchführung der einzelnen Tests ohne Mehraufwand.

Die Fachverantwortlichen müssen anhand der Anforderungen, welche im Change Request dokumentiert sind, eine Risikoabschätzung und Priorisierung der Testaktivitäten durchführen. Durch dieses Vorhaben sollen zeitliche Engpässe bei der Entwicklung und bei der Testdurchführung verhindert werden. Ebenso kann dadurch eine ausgeglichene Testdurchführung ermöglicht werden [Meie12].

Um das Testskript effizient durchzuführen wird folgende Priorisierungsmethode vorgestellt.

Methode 2: zur Priorisierung des Testskripts

Argumente:

- Wiederholbar? (Mathematische Verfahren)

- Wiederholbare Testfällen priorisieren

Priorität des Testskriptes sollte nach Leistungsanforderungen gesetzt werden, daraus folgende die 5 Arbeitsschritten. Der Auswertungsprozess basiert auf Kano-Modell [Carl13].

Schritt 1: Formulierung der Kriterien

- Testskript i ist **deutlich** (nachvollziehbar) zu automatisieren (1-5).

- Testskript i ist erfolgreich ausführbar, aus Implementierung ist die **Korrektheit** sehr hoch.

- Testskript i ist einfach zu erstellen, die entsprechende **Durchführungszeit** ist kurz.

- Testskript i wird sehr **häufig** benutzt, d.h. es ist sehr oft wiederverwendbar.

- Testskript i hat kleinen oder großen ***Entwicklungsraum***. Ein Kleiner Entwicklungsraum ist gut zu automatisieren, da das Testskript nicht oft neue gemacht werden muss.

Schritt 2: Benennung des Testskripts (Alternativen, die priorisiert werden)

Testskript i, i =1, 2, 3,… Eine Konkrete Benennung am Beispiel siehe Tabelle 5:

Testskript i (nach Konverter Name)	Beschreibung
Testskript 1	Dokument Status prüfen
Testskript 2	Dokument Merkmal vergleichen
Testskript 3	Dokument Suchen/Funktionalität testen

Tabelle 5: Testskripts benennen

Schritt 3: Bewertung der Kriterien

Gewicht: Summe [Deutl.(1) + Korrektheit (2) + Durchfüh.z.(1) + Häufigkeit(2) + Entw.r.(0)] := 6

Faktor: Gewicht/Anzahl des Matrix-Elementes (Beispiel: Faktor von Deutlichkeit= 6/25 = 0.24)

	Deutl.	Korrekheit	Durchfü.z.	Häufigkeit	Entw.r.	**Gewicht**	**Faktor**
Deutlichkeit	1	2	1	2	0	6	0,24
Korrektheit	0	1	0	1	0	2	0,08
Durchführungszeit	1	2	1	2	1	7	0,28
Häufigkeit	0	1	0	1	1	3	0,12
Entwicklungsraum	2	2	1	1	1	7	0,28
Bewertung							
wichtig	2						
gleich wichtig	1						
weniger wichtig	0						

Tabelle 6: Kriterien bewerten

Schritt 4: Bewertung des Testskripts i (Alternativen)

Entsprechende Beschreibung zur Bewertungsmatrix siehe oben die formulierten Kriterien von schritt 1.

Durchführungszeit	0,28	0	6	4	4	
Häufigkeit	0,12	1	8	3	1	
Entwicklungsraum	0,28	6	4	3	2	
Bewertung						
trifft zu	6 bis 8					
trifft z.T. zu	3 bis 5					
tritt eher nicht zu	0 bis 2					

Tabelle 7: Testskript i (Alternativen) bewerten

Schritt 5: Bewertung der Kriterien und Alternativen mit manueller Übersteuerung

Gewicht Ausrechnungsfunktion: Summe von den Kriterienwerten

Faktor: Gewicht/Anzahl des Matrix-Elementes

Ergebnis: Summe (Faktor i * Testskript i)

d.h. Ergebnis für Testskript 1 = 7*0.24 +3*0.08 + 0*0.28 + 1*0.12 +6*0.28 = 3.72

	Faktor	Testskript 1	Testskript 2	Testskript 3	Testskript i	
Eindeutlichkeit	0,24	7	3	8	7	
Korrekheit	0,08	3	8	7	4	
Durchführungszeit	0,28	0	6	4	4	
Häufigkeit	0,12	1	8	3	1	
Entwicklungsraum	0,28	6	4	3	2	
Ergebnis		3,72	3,44	3,68	2,68	
Bewertung		**Priorität**				
trifft zu	6 bis 8	Hoch				
trifft z.T. zu	3 bis 5	Mittel				
tritt eher nicht zu	0 bis 2	Klein				

Tabelle 8: Priorität setzen

Aussagen: die Priorität von Testskript 1 ist Mittel. Es kann sowohl manuell als auch automatisiert getestet werden. Testskript i sollte manuell getestet werden

Mit Hilfe des Kano-Modell Verfahrens kann jedes erstellte Testskript flexibel nach eigen definierten Kriterien priorisiert werden. Das ist ein wichtiger Schritt für effiziente Testautomatisierung durchzuführen. So erhält man ein besseres Verständnis der Kundenanforderungen, wie in der Test-Entwicklungsphase in welche Richtungsweise weiter aufgebaut werden soll und in welche nicht. Sozusagen wird das priorisierte Testskript in dem folgenden Automation-Modell nach bestimmter Reihenfolge gesteuert/getestet.

4.2.4 Spezifikation

Durch das Spezifizieren von Testfällen zu Beginn der Anforderungsanalyse kann eine umfangreiche Beschreibung mit allen notwendigen Informationen der Testfälle vom Fachpersonal erfolgen. Dadurch wird eine inhaltlich konkrete Beschreibung der Testfälle

erlangt. In der Abbildung 27 wird der Aufbau als auch die Darstellung der Spezifikation abgebildet. Die erweitere Testspezifikation von [Meie12] bietet eine allgemeine Übersicht aller analysierten Testfälle (mit der oben lautet Methode) in einem tabellarischen Format. Die detaillierte Beschreibung der Testfälle erfolgt extern in einem separaten Dokument (Abbildung 28 Testfallbeschreibung). Man sieht in Tabelle (Abbildung 27) schon Testfalltyp und Testpriorität. Das priorisierte Testskript ist inhaltlich identisch mit dem Testfall aber wird gemäß dem Testtool in einem separaten Dokument-Format mit Identifikationsnummer geordnet.

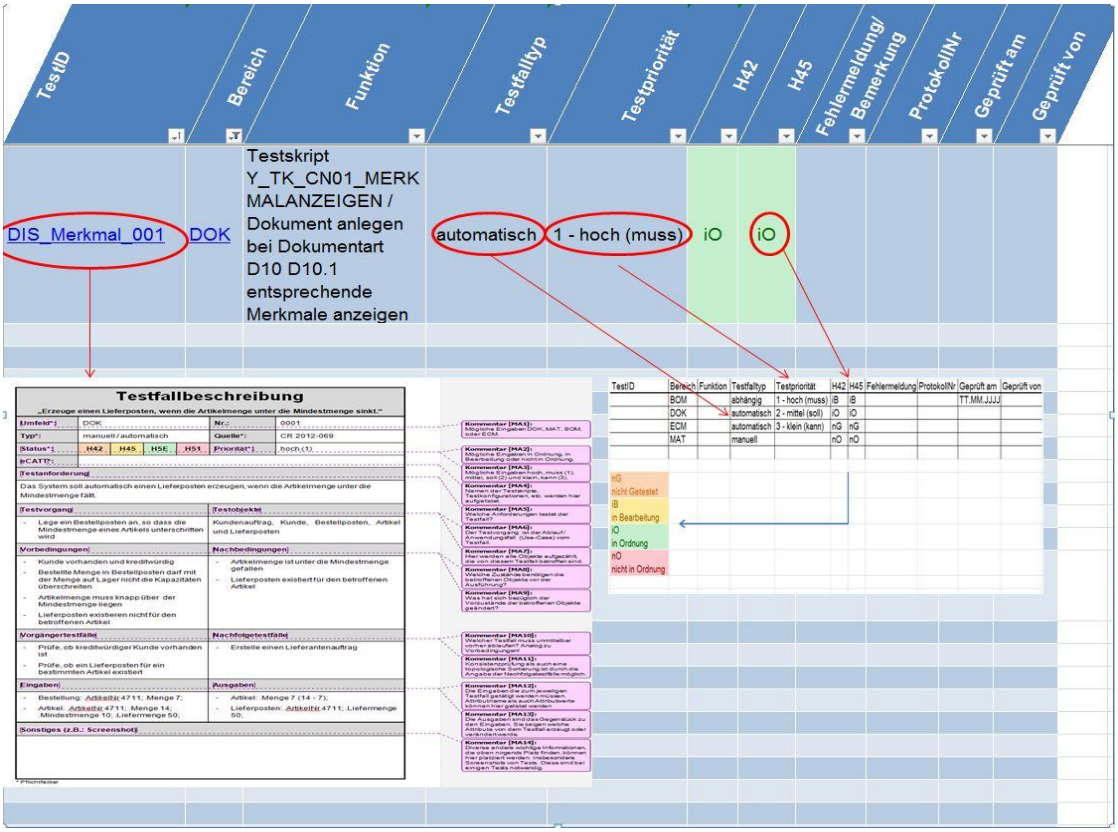


Abbildung 27: Testspezifikation Übersicht (Erweiterung von [Meie12])

Im Dokument-Format, siehe Abbildung 28 in der detaillierten Beschreibung, findet man zusätzlich zu den bereits oben genannten Informationen die Quelle und die eCATT Komponenten für diesen Testfall. Die Quelle gibt Auskunft über den Antragsteller bzw. dem dazugehörigen Change Request. Darüber hinaus werden die Testanforderung, der Testvorgang und die dafür benötigten Testobjekte tiefgehend erläutert. Daneben werden die Vor- und Nachbedingungen sowie die Vorgänger- und Nachfolgetestfälle aufgeführt, so dass auch die Verzweigungen zwischen diversen Testfällen verdeutlicht werden. Außerdem werden die Eingabe- und Ausgabewerte mit aufgeführt, wodurch eine simple und schnelle Kontrolle stattfinden kann. Letztlich gibt es noch einen Bereich für sonstige Informationen.

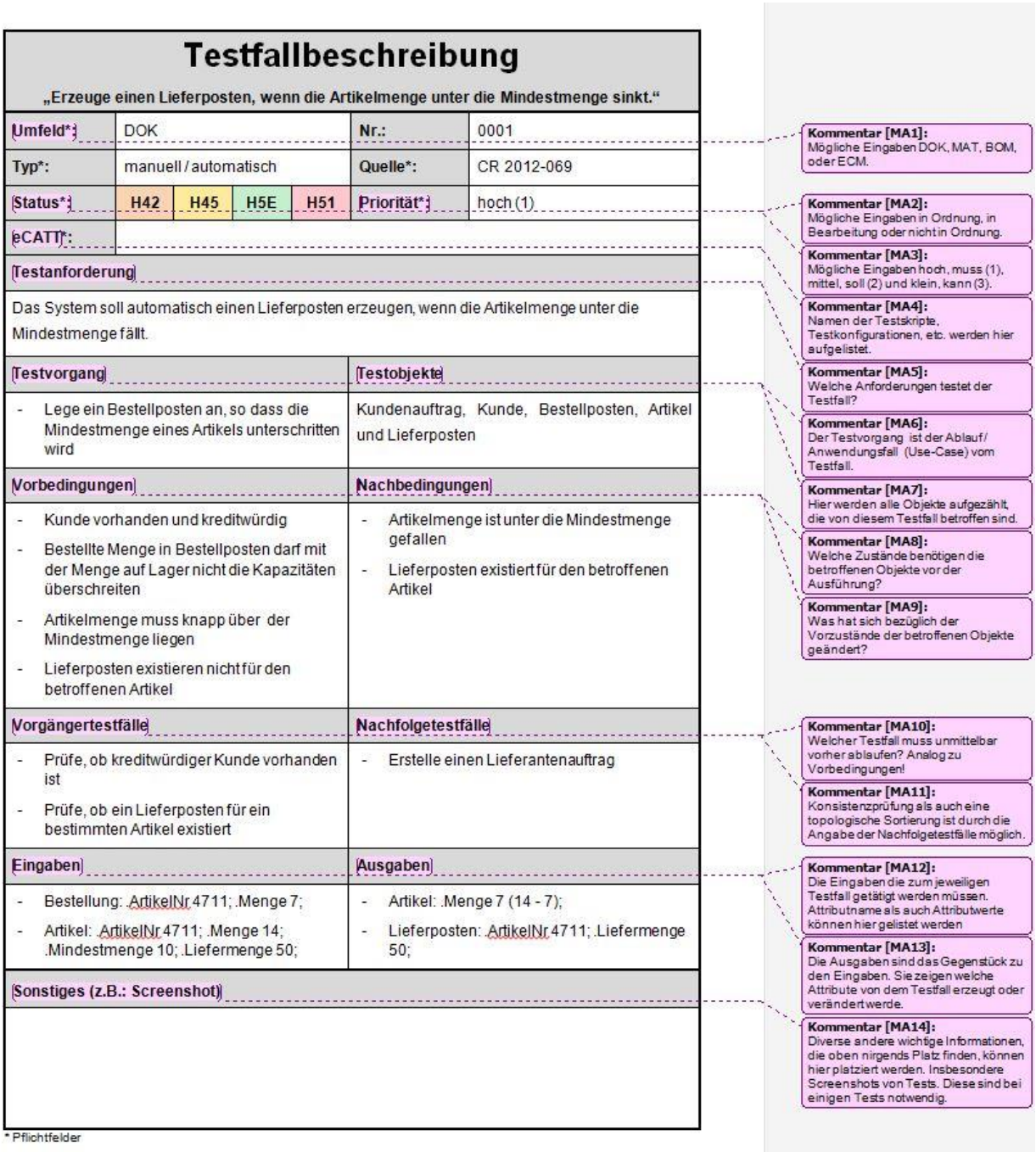


Abbildung 28: Testfallbeschreibung (Erweiterung von [Meie12])

4.2.5 Testdurchführung

Alle analysierten automatischen Tests werden nach vorgestellter Vorgehensweise gemäß ausgewähltem Testtool eCATT durchgeführt. eCATT stellt vier separate Objekttypen zur Verfügung (Abbildung 16). Die ersten drei Typen bilden die Grundlage eines Tests und der vierte kombiniert die anderen zu einem vollständigen Testfall. Die folgende Abbildung 29 bezeichnet man als den eCATT Funktionsumfang, es fasst den Entwicklungsprozess zusammen und veranschaulicht, welche Rolle die unterschiedlichen eCATT-Objekte darin spielen.

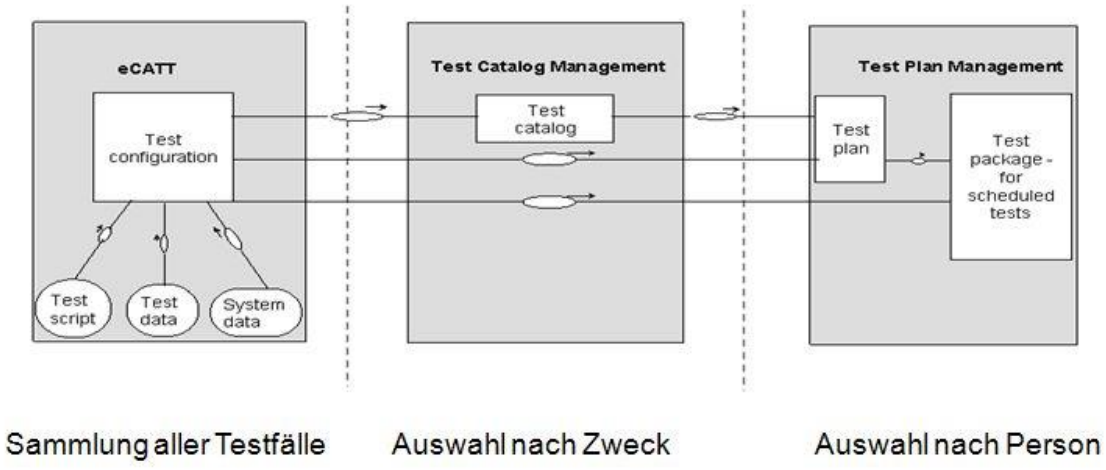


Abbildung 29: eCATT Funktionsumfang [SAP13b]

Bevor der Workflow mit dem Einstieg in eCATT beginnt, wird hier eine Namenskonvention festgelegt. Dadurch sollte ein einheitlicher Aufbau und ein Wiedererkennungswert hervorgebracht werden.

Eine mögliche Namenskonvention wäre also:

- Y = unternehmenseigener Schlüssel der HELLA KGaA Hueck & Co.
- H42 = für das betroffene Zielsystem
- liuhu10 = eigene User-ID

Um weitere Vorschläge zur Namenskonvention:

eCATT-Objekt	Vorschläge zur Namenskonvention
Testskript	Y_TS_<MODUL>_<TRANSAKTION>
Testkonfiguration	Y_TK_<MODUL>_<TRANSAKTION>
Testdatencontainer	Y_TD_<MODUL>_<TRANSAKTION>
Testbaustein	Y_BS_<MODUL>_<TCODE>_<TCODE>
Import-Parameter	P_I_<TABELLENFELDNAME>
Export-Parameter	P_E_<TABELLENFELDNAME>
Variablen-Parameter	V_BEZECHNUNG
Systemdatencontainer	Y_eCATT_SYSTEM_RFC
RFC-Verbindung	ZIELSYSTEM (TARGET)_<SYSTEM><MANDANT>
Testkatalog	<MODUL/SAP-Komponente>_Testkatalog
Testplan	<MODUL/SAP-Komponente>_Testplan
Testpaket	<MODUL/SAP-Komponente>_Testpaket

Tabelle 9: Vorschläge zur Namenskonvention [Naum12]

Methode 3: Zustandsbezogener Test durch Automation Modell

In dem vorgestellten Testprozess werden zunächst die Testblöcke nach Test-Anforderung analysiert und priorisiert. Die analysierten Testfälle so wie der Ablauf aus den einzelnen Testfällen werden in eCATT-Skriptsprache gewonnen, das Testskript entspricht wiederholt getesteten Testfällen und wird zum automatischen Testen erstellt. Um einen vollständigen Workflow in Testprozess zu konstruieren, wird ein Testautomaten-Modell mit spezifizierter Definition vorgestellt. Innerhalb dieses Modells werden mehrere eingeschränkte Test-Workflows entwickelt, mit der die entsprechenden Testskripts nach angeforderter Reihenfolge in einem Workflow entworfen werden. Mit Hilfe dieser Automaten werden automatisierbare Testprozesse übersichtlich dargestellt, damit die möglicherweise wiederverwendbaren Testskripts effizient aufgerufen werden können und die sehr aktiven aufgerufenen Testskripts zeitig gepflegt, erweitert und entwickelt werden können. Die Vorstellung wird im nächsten Kapitel 4.3 „Vorgehensweise“ ausführliche beschrieben.

Prinzip von der Testautomation: Anschauliche Abbildung 4.2.11: Testautomation-Modell, basiert auf NFA als 5-Tupel (NFA*: Nichtdeterministischer endlicher Automat), Testskripte die immer bis Endzustand fließen bedeuten einen automatisierbaren Workflow. Sonst kann man zusätzliche Testschritte anhand des Kontexts von einzelnen Testfällen manuell oder automatisch durchführen, die sind natürlich abhängig von dem Vorgänger (Testskript) und der Priorität des nachfolgenden automatisierbaren Testskript bzw. nach der Berücksichtigung von dem nächsten Endzustand setzen, die Priorität beeinflusst die Analyse und Implementierung von nächsten Workflows.

Definition: NEA Test - Automate A als 5-Tupel (T, Ɛ, δ, S, F)

- T ist eine endliche Menge von Testskripten. ($\left| T \right| < \infty$)

- Ɛ ist das Eingabealphabet, ($\left| Ɛ \right| < \infty$), entspricht die Import-/Export Parameter

- $\Delta \leq (T \text{ X } Ɛ) \text{ X } T$ ist die Übergangsrelation (Testskript_3) U (Additionale Anforderung) -> (Testskript_3)*

- S ist start-Testskript

- $F \leq T$ ist eine (endliche) Menge, möglicher akzeptierender Zustände (Finalzustände), wenn der Automat nach Durchführung des Eingabealphabet, $w \in Ɛ^*$ in einem Zustand aus F hält, so gehört w zur Testszenario: K(A).

Dieser Automat beginnt im Zustand Testskript_1, liest die Eingabe Buchstabe für Buchstabe (Eingabe Ɛ) und bewirkt Zustandsänderungen. Die Eingabe wird akzeptiert, wenn der zuletzt erhaltene Zustand akzeptierend ist. Der Anfangszustand wird durch einen Pfeil hervorgehoben, akzeptierende Zustände und Endete Zustände werden Doppelt umrandet. Was zusätzlich kommt, wird im Quadrat die Zusätzliche Anforderung zuerst als Eingabe bemerkt damit die Automaten normal enden konnte. Bei dem nächsten Durchlauf aber wird diese als neuer Zustand Testskript

erstellt. Dieses Zusätzliche Quadrat ist quasi für Entwicklungsphase vorbereitet, trotzdem lässt sich das alte Testszenario nicht negativ beeinflussen.

Simulation eines NFA Test - Automat:

S:= {Testskript_0} /* Testskript Index entspricht Priorität des Testskriptes, hier mit 0

anfangen, d.h. undefiniert */

while (es gibt noch „Testszenario“) {

c:= gehe „Ablauf des Testfälle“ rein;

H: = ∅;

For (t in S) {H: = H U delta (t , c); }

S: = H,

}

If (S ∩ $F \neq \emptyset$) return 1;

Return 0;

Datenstruktur für H:

- Stack (FIFO-Queue) und
- “Block”-Feld

Anmerkung: FIFO-Queue d.h. dass das Testskript zuerst priorisiert wird und danach nach Reihenfolge geordnet wird, „Block”-Feld entspricht den unterschiedenen Testszenarios, man kann es auch die Klasse des Testszenarios nennen.

Zusammenfassung:

Die Idee aus diesem Modell wird übereingestimmt mit dem Ziel des Testkonzepts von „effizient“, das überzeugende Argument wäre:

- Größe des NFA linear in der Länge des regulären Ausdrucks (des regulären Ausdrucks: (Testskript_3) U (Additionale Anforderung)*)
- Lineare Laufzeit O(| T | (| W |), falls | Ɛ | konstant

Man hat dadurch einen guten Überblick über das Testskript erstellen bzw. weiterentwickeln, vor allem einen vollständigen Abdeckungsgrad von einem Testszenario, es wird deutlich und einfach durch Format-Sprache erhöht. Damit ist es möglich besser für die Implementierung Test-Treiber aus Testtools auszuwählen. Siehe Abbildung 30 Testautomation-Modell.

Anmerkung zur Unterscheidung der zwei Begriffe Testszenario und Workflow:

- (Regulär Sprache)* heißt hier Testszenario und gleichzeitig neu erstelltes Testszenario, das nach neu bewerteter Priorität ist, als Workflow zu nennen.

- (Reguläre Sprache)* entspricht hier einem Testszenario, ist es gefolgt von einem weiteren, mit bewerteter Priorität.

Reguläre Sprache und Testautomaten siehe im Appendix F.

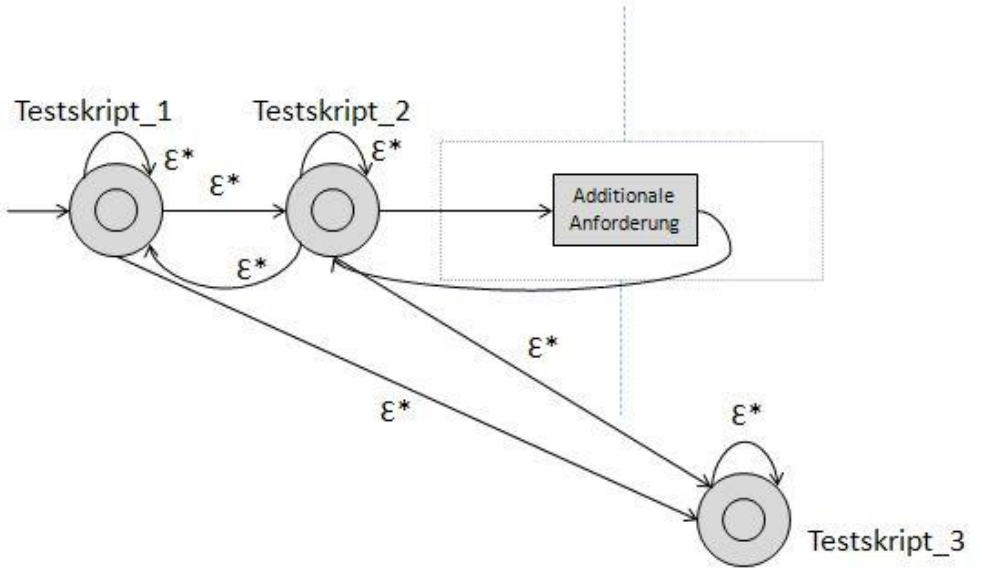


Abbildung 30: Testautomation-Modell

4.2.6 Testauswertung

Die Ergebnisse aus durchgeführten Tests (je Testfall) werden überprüft. Dabei wird das Ist-Ergebnis mit dem Soll-Ergebnis verglichen und anschließend eine Entscheidung über das Testergebnis (ok oder Fehler) herbeigeführt.

- Bei Fehler: Testfall bleibt offen, der in Manuelle Test Block einfließen soll und weiter getestet werden

- Bei OK: Testfall gilt als erledigt

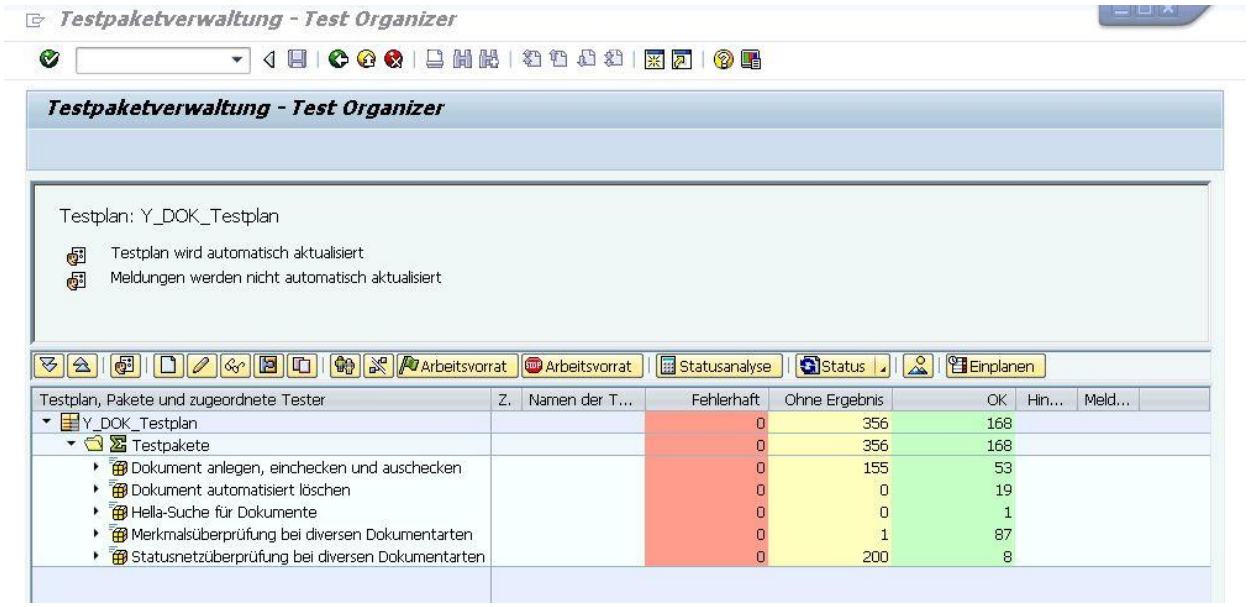


Abbildung 31: Testberichts bei der Statusübersicht

Zur Organisation und Planung von Testfällen bietet eCATT die beiden Komponenten Testkatalog und Testplan. damit ist eine Generierung eines Testberichts bei der Statusübersicht für den Entwickler möglich. Siehe Abbildung 31, der genaue Arbeitsschritt über Testkatalog und Testplan kann man das Praxis Buch [Naum12] nutzen. Im Dieser Diplomarbeit wird das Ergebnis direkt vom erfolgreichen Protokoll ausgelesen und dokumentiert. Die nicht bestandenen Tests werden manuell wiederholt getestet. Fehlersuche und Behebung erfolgen separat. Die gehört nicht zu Testautomatisierungskonzept.

4.3 Vorgehensweise zur Realisierung der Testautomation

Dieses prinzipielle Vorgehen schließt 6 Aspekte in einem Vorgehensweisemodell ein:

Schritt 1: Testplan Initialisierung

Schritt 2: Analyse

Schritt 3: Spezifizierung

Schritt 4: Realisierung

Schritt 5: Ergebnisse

Schritt 6: Dokumentation

Das Vorgehensweisemodell (siehe Abbildung 32) ist basierend auf einem Prototyp-Wasserfall [Kuhr13] erstellt, das mit Rücksprungs Möglichkeiten erweitert und in 6 Phasen organisiert wird. Diese werden jeweils als Testplan Initialisierung, Analyse, Spezifizierung, Realisierung Ergebnisse und Dokumentation bezeichnet. In diesem Modell hat jede Phase vordefinierte Start- und Endpunkte mit eindeutig definierten Ergebnissen, d.h. in jedem Schnitt am jeweiligen Phasenende werden die Ergebnisdokumente verabschiedet.

Zur ersten Phase Testplan Initialisierung sollte man die konkret beschriebenen Testdaten sowie Anzahl des Testobjekts und die entsprechende Test-Kontext (Systemumgebung) wissen. Diese Phase ist erst mal für den nächsten Schritt Analyse vorbereitet, also alle Testdaten sind klar nach Anforderung vom Entwickler definiert und werden dann analysiert durch eine implementierte Mathematische Rechnungsweise. In dieser Analyse-Phase wird über Testdaten herausgefunden was zu automatisieren und was im manuellen Test günstig ist, und zusätzlich werden die erstellten Testskripte nach vordefinierten Kritiken priorisiert. Alle analysierten Testfälle aus Testdaten werden in die nächste Phase spezifiziert. Die Testspezifikation bietet eine allgemeine Übersicht aller Testfälle in einem tabellarischen Format. In dieser Testspezifikation verlinkt extra noch eine „Testfallbeschreibung–Spezifikation“, die auf Anforderung basiert, und hält den Testvorgang schriftlich fest. Der Tester liest alle Information aus dieser Testfallbeschreibung-Spezifikation um den Automation-Test in eCATT zu realisieren. Nach dem Realisierungsschritt trägt der Tester die Ergebnisse in die Spezifikation-Tabelle. Am Ende jeder Aktivität steht ein fertiggestelltes Dokument, die eine Einführung bzw. Test-Ergebnisse enthält. Die Einführung steht für Nutzung zur Verfügung und die Ergebnisse werden für nächste Tester/Test gestartet.

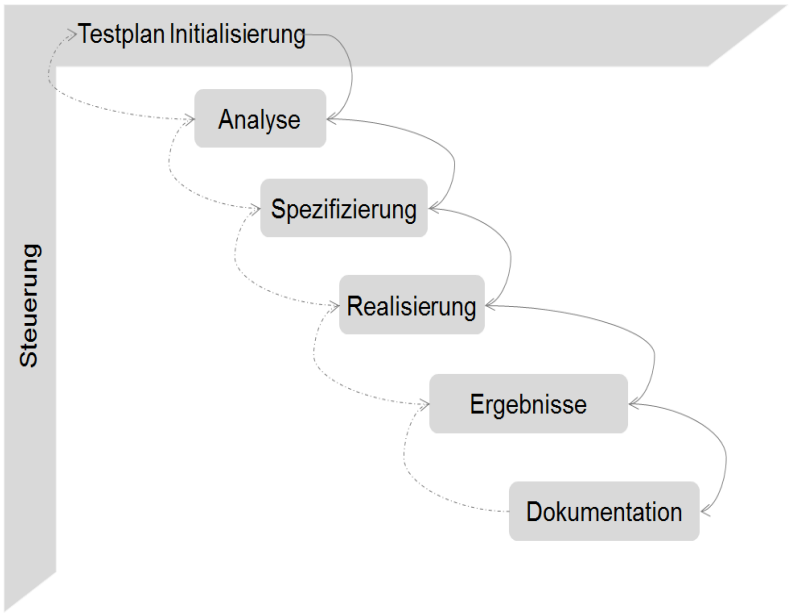


Abbildung 32: Vorgehensweise-Modell

Die Vorteile des erstellten Vorgehensweise-Modells:

- klare Abgrenzung der Phasen
- einfache Möglichkeiten der Planung und Kontrolle
- bei stabilen Anforderungen und klarer Abschätzung von Kosten und Umfang sehr effizientes Modell

4.3.1 Voraussetzung für den Einsatz von eCATT

eCATT (extended Computer Test Tool) ist ein Softwaretestautomatisierungstool, welches in das Standardisierungskonzept der SAP AG fällt. Generell sind Programmierkenntnisse für den Einsatz von eCATT nicht erforderlich. Zur Nutzung des kompletten Funktionsumfangs kann aber auf die SAP-eigene Programmiersprache ABAP zurückgegriffen werden.

Bevor die Arbeit mit eCATT beginnen kann, müssen zunächst zwei Schritte am System nachvollzogen werden, um eCATT einsetzen zu können:

Schritt 1: eCATT-Einrichtung: In jedem System, auf dem eCATT laufen soll, muss durch Aufrufen der Transaktion SE11 (ABAP Dictionary: Einstieg) die Berechtigung zum Ausführen von eCATT zugeteilt werden.

Schritt 2: Einrichtung SAPGUI: Es gibt in eCATT eine User-Interface-Ansteuerung namens „SAPGUI“. Mit dieser Ansteuerung lassen sich generell Transaktionen aufzeichnen. Sie gehört zu den mächtigsten Befehlen in eCATT und muss eingerichtet werden. Der Parameter für SAP GUI Scripting muss geprüft und entsprechend geändert werden.

4.3.2 Technische Voraussetzung

Neben Anforderungen an die Testorganisation existieren einige technische Voraussetzungen [HLT06] für den Einsatz von eCATT:

Voraussetzung 1: Einrichten der Truste-RFC-Verbindungen

Typischerweise besteht eine Testlandschaft aus mehreren Systemen und die Kommunikation zwischen Testsystem und zentralem Testverwaltungssystem erfolgt hierbei über die Remote-Function-Call-Schnittstelle (RFC). Die folgende Abbildung 33 Systemumgebung, zeigt die drei Verbindung in H42, H45, H5E System, H45 als zentralem Testverwaltungssystem. Gleichzeitig H42 entspricht Entwicklungssystem. D.h. Testskript wird in H42 entwickelt und nach Moderieren des Tests kann auch in H45 und H5E System durchgeführt. Dazu braucht die RFC-Verbindung für H5E Schulungssystem: Y_eCATT_RFC_H5E_liuhu10und und die RFC-Verbindung für H45 zentralem Testverwaltungssystem: Y_eCATT_RFC_H42.

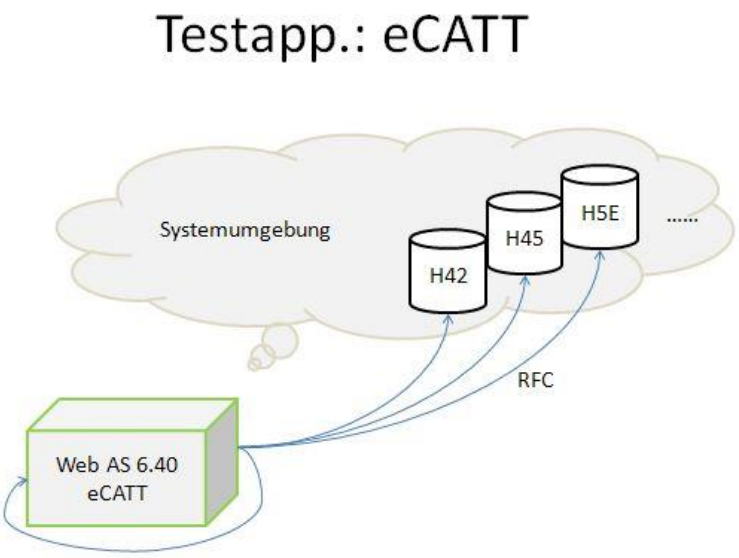


Abbildung 33: Systemumgebung

In einem Systemdatencontainer werden die unterschiedlichen RFC Verbindungen der Zielsysteme abgelegt (siehe Abbildung 34). Es ist die zentrale Verwaltungsstelle der Verbindungen. Die Testskripts wählen daraufhin einen Systemdatencontainer aus. Folglich können die Testfälle eine RFC Verbindung selektieren, welche im Systemdatencontainer ablegt worden ist. Durch diese Selektion baut der Testfall beim Ausführen des Testfalls eine Remote Verbindung zum Zielsystem mit den entsprechenden Mandaten auf [Naum12].

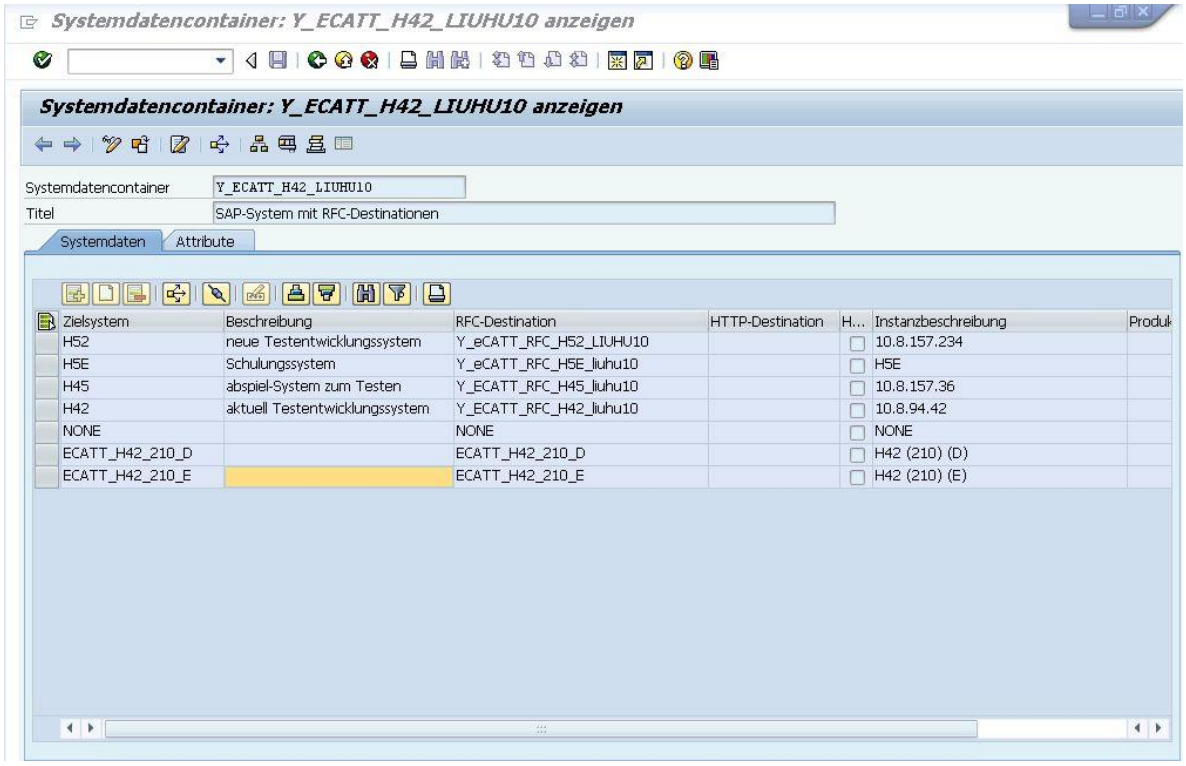


Abbildung 34: RFC-Verbindungen in H42

Voraussetzung 2: Support Package Level

Zunächst einmal müssen alle zu testenden Systeme bestimmte Mindestanforderungen an die installierten Support Package Level der Basis erfüllen, sofern Sie über ein Basisrelease älter als 6.20 verfügen [HLT06]:

Mandantenpflege: für jeden Mandanten, in dem ein automatisierter Test mittels eCATT laufen soll, muss die explizit erlaubt werden. Die Anpassung der Mandanten erfolgt in der Transaktion SCC4 [HLT06]: Das erste Testtool von SAP war CATT. Es konnte mit der Transaktion SCAT gestartet werden. Mit dem SAP-Basis-Release 6.20 wurde das Testtool eCATT eingeführt, die Starttransaktion lautet SECATT. Für eCATT-Basis Funktion sowie Testskript anlegen/ändern bzw. Migration von CATT nach eCATT können ab dem Basis-Release 6.40 /NetWeaver-Release NW04/ SAP ERP-Release 6.0 anwenden [Naum12]. Unter Einschränkungen beim Start von CATT und eCATT sollte Option „eCATT und CATT erlauben“ aktiviert werden.

Voraussetzung 3: SAPGUI-Skripting aktiveren

Eine Aufzeichnung mit Controls mittels des SAPGUI-Treibers muss sowohl im zentralen Testsystem als auch im allen Zielsystem SAP-GUI-Skripting aktiviert sein. Die Einstellung der Skripting-Berechtigung auf dem Server wird mittels der Transaktion RZ11 gepflegt. Der entsprechende Profilparameter lautet sapgui/user_scripting und muss auf ‚TRUE‘ gesetzt werden. Siehe folgende Abbildung 35 Profilparametereigenschaften.

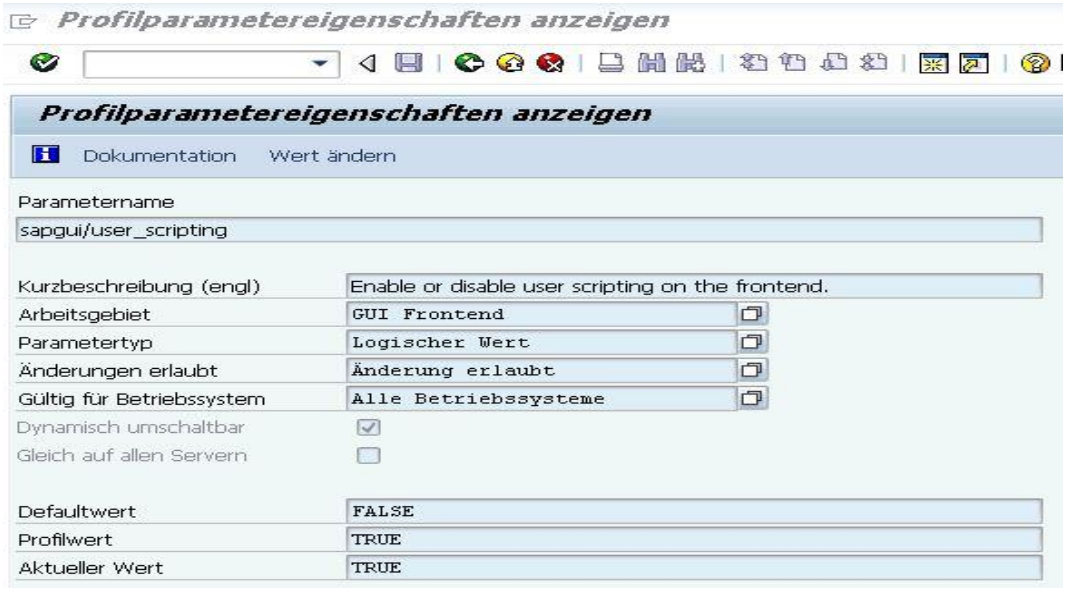


Abbildung 35: Profilparametereigenschaften

Voraussetzung 4: Parametervorbelegung über Transaktion SU3

Diese für den Anwender eine Möglichkeit kann beim Einsatz von eCATT zu Fehlern führen. Siehe folgende Abbildung 36.

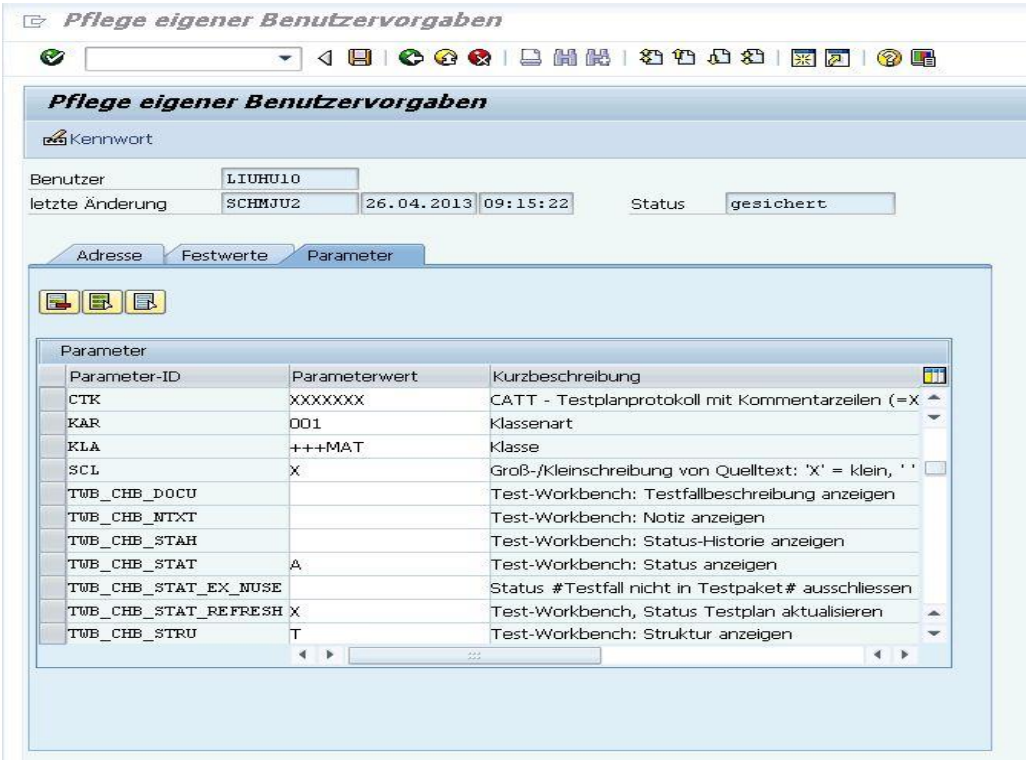


Abbildung 36:Pflege eigener Benutzervorgaben

4.3.3 Modularisierung und Ausführen von Testskripten mit eCATT

Ein Testfall wird durch eine Testkonfiguration abgedeckt. Eine Testkonfiguration besteht aus einem Skript, einem Systemdatencontainer und Testdaten. Die Testdaten werden nach inhaltlichen Gruppen geordnet in Testdatencontainern gespeichert und in der Testkonfiguration passend zu dem Skript ausgewählt und kombiniert. Dadurch wird die Redundanz der Testdaten auf ein Minimum reduziert, Übersichtlichkeit und Wartbarkeit bleiben erhalten [HLT06].

Eine wichtige Fähigkeit von eCATT-Skripten ist die Eigenschaft, andere Skripte aufzurufen. Hierdurch wird es möglich, einen Geschäftsprozess in eine Kombination von einzelnen Geschäftsvorfällen zu zerlegen und diese jeweils durch ein eigenes Skript abzudecken. Der REF-Befehl dient dem Aufruf anderer Skripte. Siehe Abbildung 37 Sequenz von eCATT-Skripten.

Abbildung 37: Sequenz von eCATT-Skripten

Die Kommandoschnittstelle des REF-Befehls besteht aus den Import und Exportparametern des aufgerufenen Skripts.

Die einfachste Organisationsform ist der konsequente Verzicht auf Modularisierung. D.h. ein Skript wird pro Testfall erstellt. Bei der Verwendung einer Sequenz von Skripten wird der Ablauf des Testfalls in eine Sequenz kleinerer Skripte zerlegt. In dieser Diplomarbeit werde die einfachste Organisationsform nehmen, da jedes Testskript als ein Zustand im Testautomation-Modell betrachtet wird. Es lohnt sich nicht so bei der Erstellung eines Skriptes zu investieren, weil es später in Testautomation-Modell (Abbildung 30) sowieso mit anderem Testskript kombiniert wird. Vor allem zusätzlich Import-/Exportparameter zu pflegen, sind immer ein großer Aufwand und fehleranfällig. Wir lassen die Parametrisierung nur bei Testkonfiguration-Schritt. Die konkrete Ausführen Schritt von Testskript wird in Kapitel 4.3.4 anhand ein Workflow am Bespiel der HELLA vorgestellt.

4.3.4 Workflow am Beispiel der HELLA KGaA Vorstellung

Diese Workflow Vorstellung wird gemäß dem Vorgehensweise-Modell an das konkrete Beispiel schritt für schritt durchgeführt.

Workflow für Dokumentinhalt-abhängige Merkmale Prüfung

Schritt 1: Analyse durch Formel: p (20 w+15 f) > p (5 w + 5 f) + 240 + (p - 1) 10

Entscheidung über automatischen Testfall: ein Testfall wird in Mircosoft Excel-Tabelle beschrieben siehe Abbildung 4.2.4: Eine Testfall-Beschreibung über Merkmale im PDM System

Testfall-Dokument Bei Funktion „Dokument anlegen“, Transaktionscode „CV01N“, Dokumentart D10 als eine Eingabe, sollte die folgende Merkmale unter den Zusatzdaten angezeigt:

- Berechtigungsgruppe
- Vertraulichkeitsstufe
- Dokumentinhalt
- Mastermaterial
- PEP-Phase aus Materialstamm
- Musterstand HELLA
- Basisvariante
- CAD-Erstellungssystem
- Version/Release (CAD-System)

- CAD-Laufumgebung
- CAD-Baugruppe
- LIN
- Freigegeben/abgelehnt von

Dieser Testfall wird in drei Systeme jeweils H42, H45 und H5E getestet. D.h. P = 3 gemäß obere mathematische Forme **p (20 w + 15 f) > p (5 w + 5 f) + 240 + (p - 1) 10**

- die entsprechen manuell-Zeitaufwand: **p (20 w + 15 f)**
- die entsprechen Testautomatisierung-Zeitaufwand: **p (5 w + 5 f) + 240 + (p - 1) 10**

Siehe Analyse-Ergebnisse P = 3 in untere Abbildung 38 Automatisierungstest vs. Manuell Test.

Abbildung 38 zeigt: Einmal Ausführung dieser manuellen Testfall über Anzeigen von Dokumentart D10 erfordert in ein Testsystem H42 etwa 20 min und in drei Testsysteme H42, H45 und H5E etwa 60 min. Diese Zeitspanne bezieht sich auf die erfolgreiche Ausführung ohne Auftreten von Fehlern. Sollte ein Fehler auftreten, erhöht sich die für diesen Testfall erforderlich Zeit sofort. Und Einmal Automatische Ausführung dieser Testfall braucht man 275 min in ein Testsystem, diese Zeit für die Entwicklung ist die Durchschnittszeit einer Person, die bisher wenig Erfahrung mit Testautomatisierung hat. Für einen erfahrenen Testautomatisierer erfolgt die Entwicklung wesentlich schneller. Man sieht einen hohen zeitlichen Vorteil nach erhöhter Wiederholungsanzahl der Testfall bzw. Testsystem-Anzahl, wenn die Testfälle automatisiert ausgeführt werden. Also dieser testautomatisierte Test rentiert sich bereits nach ca.4 Wiederholung.

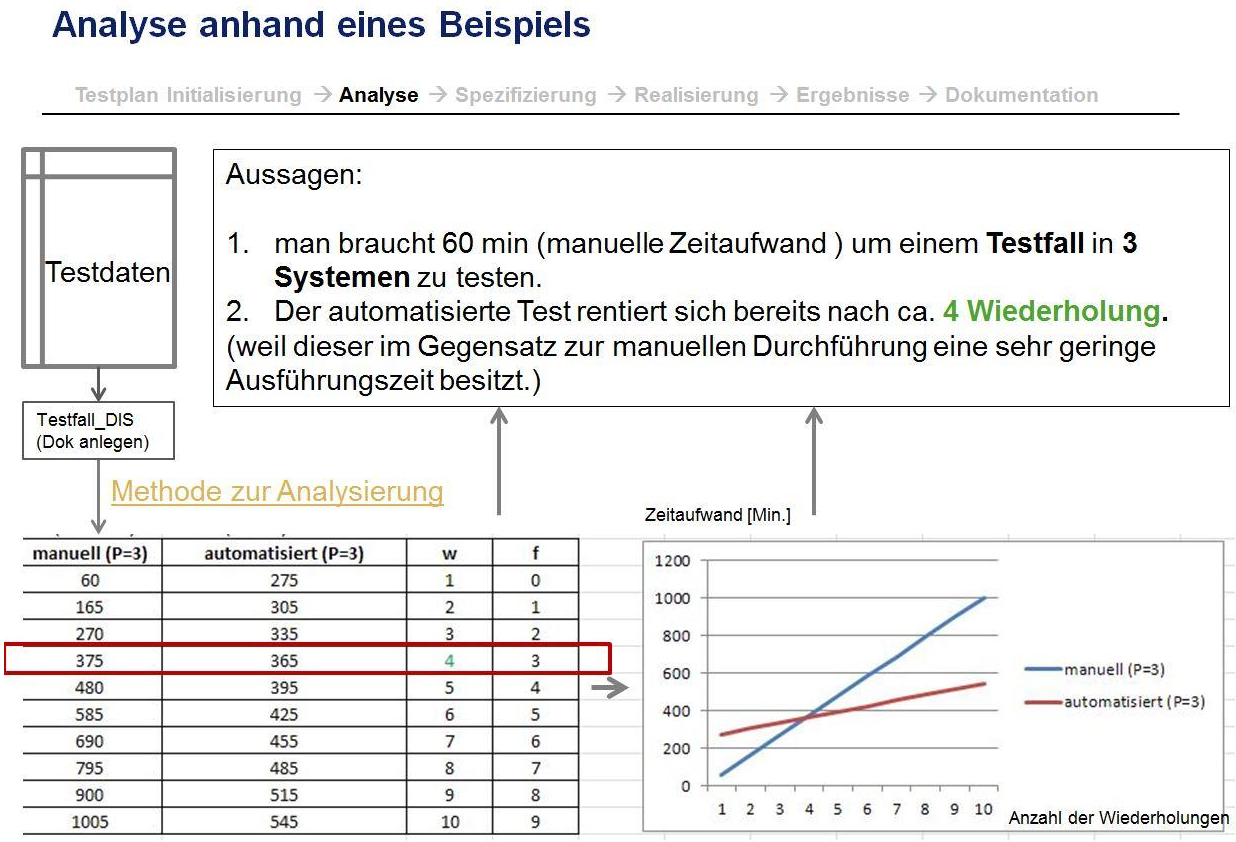


Abbildung 38: Automatisierungstest vs. Manuell Test

d.h. dieser Testfall wird in MS Excel Tabelle min 4-mal wiederholt getestet und wird in nächste Schritt als automatisierbares Testskript in eCATT aufgezeichnet.

Schritt 2: eCATT-Testskript erstellen:

ein eCATT-Skript besteht aus Attributen, Parametern und Kommandos. Die Attribute eines Testskripts umfassen neben Verwaltungsinformationen wie Titel, Paket, Verantwortlicher und Komponente auch Schlagwörter und Versionierungsinformationen, die vom System zur Unterstützung der Skriptverwaltung verwendet werden. Oberen automatischen Testfall nennt man hier als Testskript: Y_TS_CV01N_MERKMALANZEIGEN.

In das Feld „Komponente“ der „Kopfdaten“ wird die betroffene Komponente des SAP-Systems eingetragen. Generell kann man auch in die Komponente „BC-TWB-TST-ECA“ für eCATT oder „SAP“ für Anwendungskomponenten eintragen, wenn man die betroffene Komponente nicht genau kennt. In diesem Feld kann man nach der betroffenen Komponente aber auch über die F4-Hilfe suchen. Nach Betätigung der F4-Hilfe erscheint folgende Auswahlübersicht: die entsprechende Allgemeine Daten unter Attribut siehe folgende Abbildung 39.

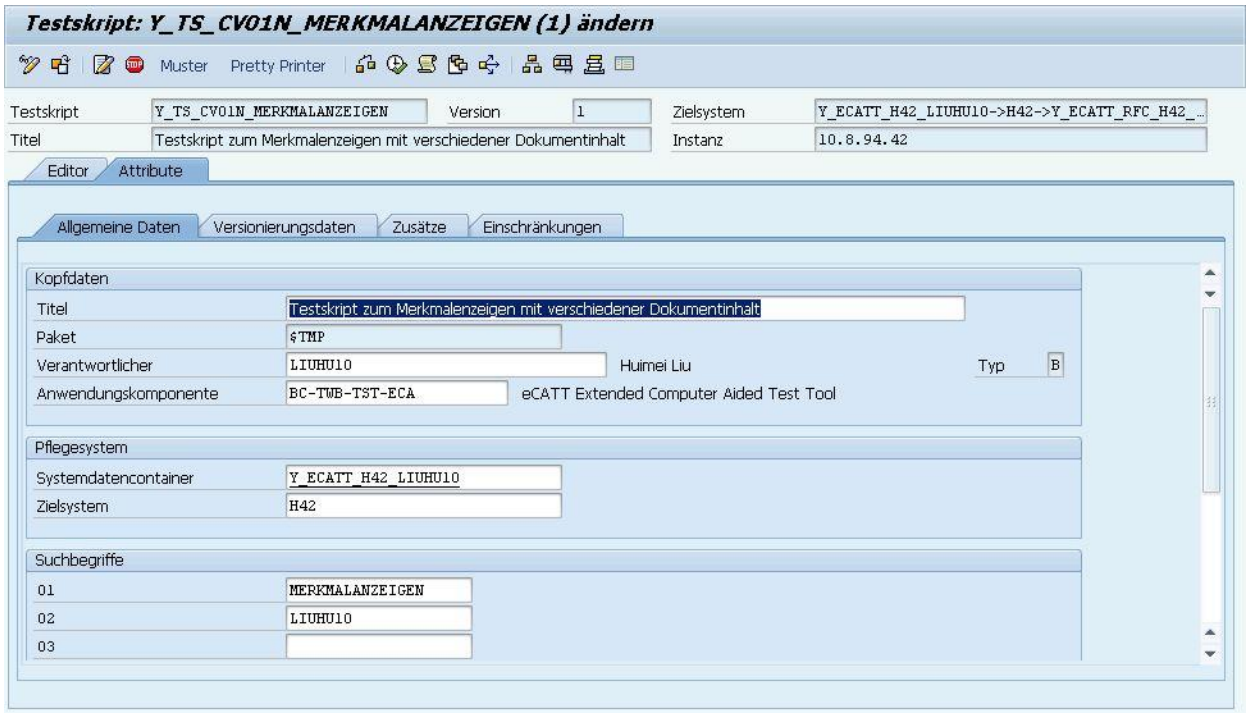


Abbildung 39: Die Allgemeinen Daten eines Testskripts

eCATT führt zum Aufbau modularer Testfälle die Teilobjekte Testskript, Testkonfiguration, Testdatencontainer und Systemdatencontainer ein [SAP13a].

Testskript besteht aus einem Skripttext, der den Ablauf des Tests algorithmisch beschreibt. Siehe Bereich 1 in Abbildung 40: Testskript-Screenshot. Das Testskript hat weiterhin eine Parameterschnittstelle mit Import und Exportparametern sowie lokalen Variablen. Im Bereich 2 befindet sich eine Eingabemöglichkeit, mit der die Parameter eines Skripts erstellt werden können. Wenn man eine Kommandoschnittstelle bearbeiteten möchte, wird rechts davon zusätzlich der Strukturedитor geöffnet, die durch Doppelklick auf den Namen einer Kommandoschnittstelle im Kommandoeditor aktiviert wird. Im Bereich 3 Skripteditor wird die Skriptlogik erstellt und entsprechende Testvorgänge aufgezeichnet. In dem Fall wird Transaktion CV01N mit der SAPGUI-Befehl aufgezeichnet. Der SAPGUI-Befehl dient der Ansteuerung des SAPGUI-Treibers. Er hat die Form:

SAPGUI (<Transaktionscode>, <Kommandoschnittstelle>, [<Zielsystem>])

Allgemein zum Aufbau des eCATT-Skript siehe [Naum12].

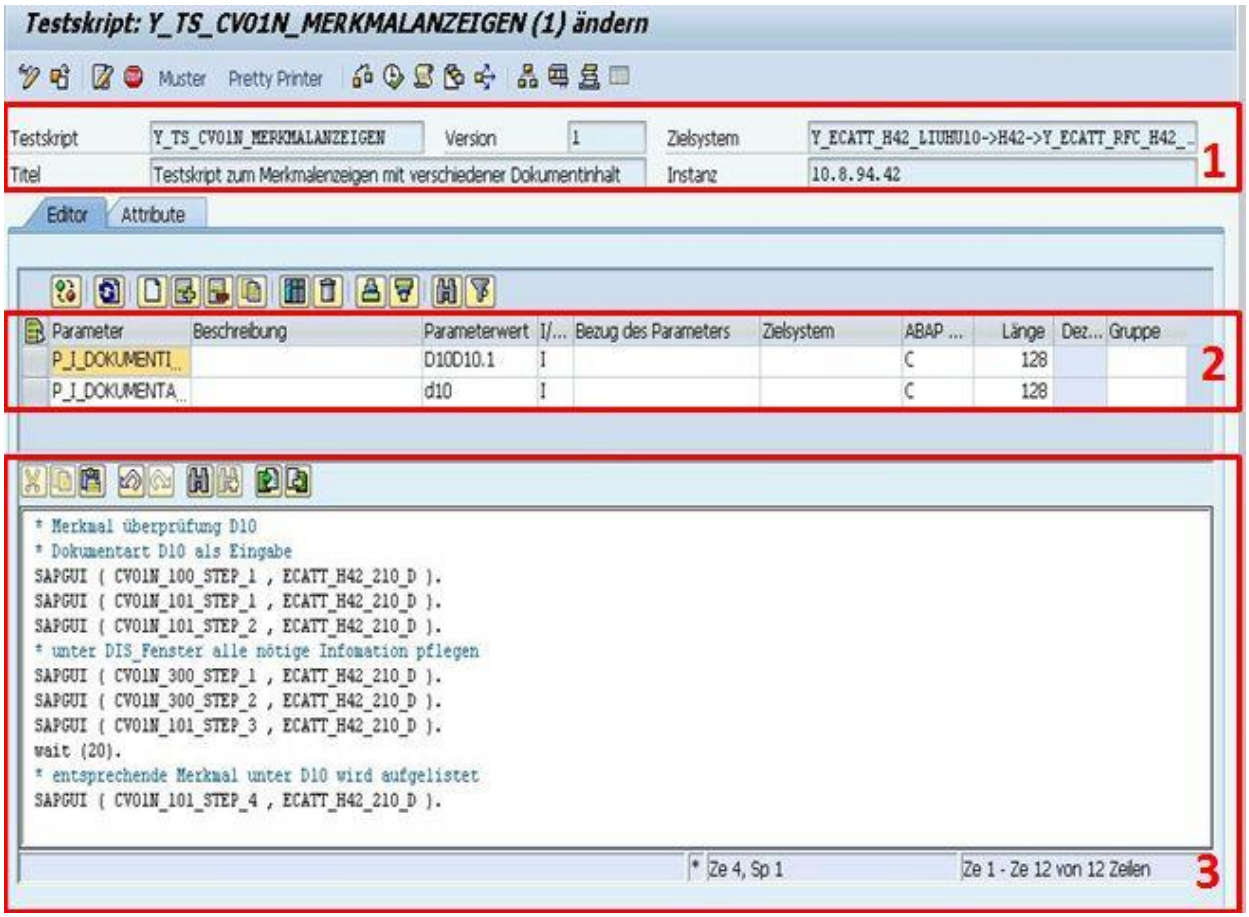


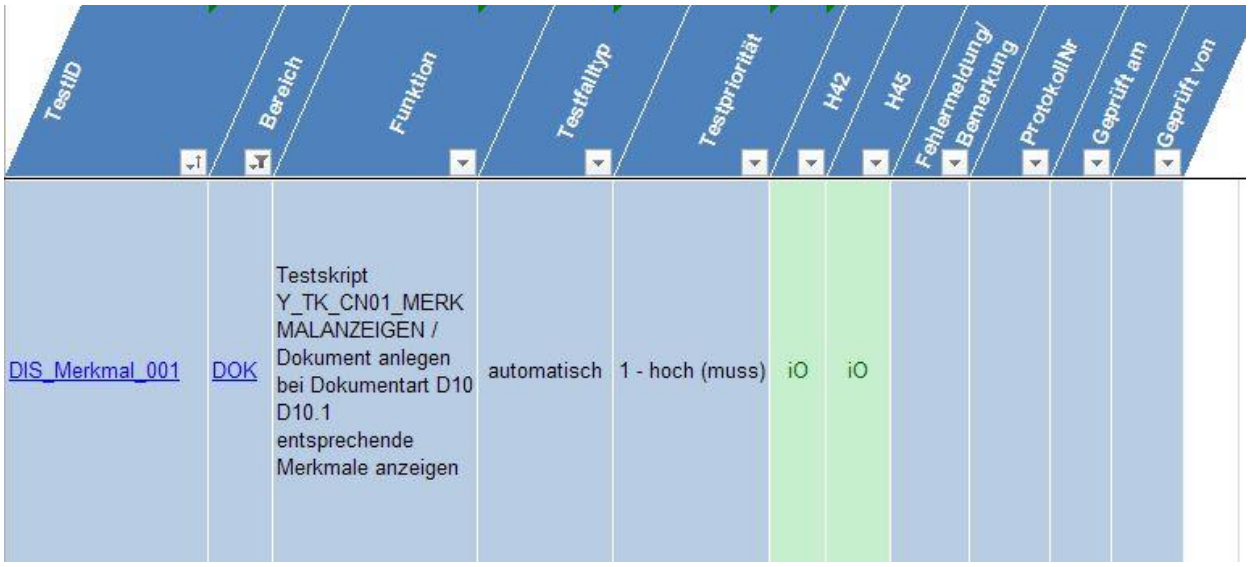
Abbildung 40: Testskript-Screenshot

Schritt 3: Priorisierung der Testskripts

Die Entwicklung eines Testskripts muss auch geplant werden. Daher wird die Bewertung der Testskripts hier in drei Stufen unterteilt. Die Priorisierungsverfahren siehe Kapitel 4.2.3. Die erste Stufe wird für Testfälle vergeben, bei denen die Durchführung von absoluter Notwendigkeit ist. Die Priorisierung ist somit hoch und eine Ausführung muss stattfinden. Die zweite Stufe deckt die Testaktivitäten ab, welche eine relevante Bewertung erhalten, jedoch eine etwas geringere Bedeutsamkeit im Hinblick auf die Geschäftsabläufe besitzen. Es findet daher eine mittlere Priorisierung statt. In der Letzten Stufe findet man die Testfälle wieder, die keine hohe Relevanz aufweisen und nur bei freien Kapazitäten durchgeführt werden. Die Bewertung ist sehr gering und die Durchführung kann dennoch in der Regel stattfinden, jedoch ist es nicht zwingend verlangt [Meie12]. Über eine Formulierung der Kriterien zur Bewertung kann man flexibel nach Testanforderung abstimmen.

Schritt 4: Spezifizierung

Das obere Analyse-Ergebnis über die Entscheidung eines automatischen Testfalls sollte in die entsprechende Spezifikation-Tabelle eingetragen. Durch das Spezifizieren von Testfällen zu Beginn der Anforderungsanalyse kann eine umfangreiche Beschreibung mit allen notwendigen Informationen der Testfälle vom Fachpersonal erfolgen. Dadurch wird eine inhaltlich konkrete Beschreibung der Testfälle erlangt. Siehe Abbildung 41 Spezifikation für Merkmal Prüfung.



TestID	Bereich	Funktion	Testfalltyp	Testpriorität	H42	H45	Fehlermeldung/ Bemerkung	ProtokollNr	Geprüft am	Geprüft von
DIS_Merkmal_001	DOK	Testskript Y_TK_CN01_MERKMALANZEIGEN / Dokument anlegen bei Dokumentart D10 D10.1 entsprechende Merkmale anzeigen	automatisch	1 - hoch (muss)	iO	iO				

Abbildung 41: Spezifikation für Merkmal Prüfung von Dokumentart D10 und Dokumentinhalt D10.1

Diese Testspezifikation bietet eine allgemeine Übersicht aller analysierten Testfällen in einem tabellarischen Format. Darunter müssen die folgenden Felder nach Analyse-Schritt gepflegt werden:

- TestID (wodurch eine konkrete Testfall-Beschreibung aus gegebener Original Testdaten aus MS-Excel Tabelle erfolgen kann, diese detaillierte Beschreibung wird in einem separaten Dokument verlinkt)

- Bereich (DOK, MAT, PreBOM…)

- Funktion (eine kurze Funktionsbeschreibung)

- Testfalltyp (automatisch/manuell)

- Testpriorität (1-hoch, 2-mittel, 3-klein)

Die restlichen 6 Spalten sollten auf jeden Fall nach dem Ausführungsschritt berücksichtigt werden. Sowie die Statuswerte der einzelnen Systeme, die entsprechende Fehlermeldung und ProtokollNr nach Durchführung, von wem wird es geprüft und dazu geprüfte Datum.

TestID: In der Testfall-Beschreibung findet man zusätzlich zu den bereits oben genannten Informationen die Quelle und die eCATT Komponenten für diesen Testfall. Die Quelle gibt Auskunft über den Antragsteller bzw. dem dazu gehörigen Change Request. Siehe Abbildung 42 Testfall-Beschreibung-DIS_Merkmal_001. Ins Detail siehe Appendix E. Darüber hinaus wird die Testanforderung, der Testvorgang und die dafür benötigen Testobjekte tiefgehend erläutert. Daneben werden die Vorgänger- und Nachfolge Testfälle ausgeführt, so dass auch die Verzweigungen zwischen diversen Testfällen verdeutlicht werden. Außerdem werden die Eingabe-und Ausgabenwerte mit aufgeführt, wodurch eine Simple und Schnelle Kontrolle stattfinden kann. Dieser Schritt erfolgt genau Information aus Testautomation-Modell. Letztlich gibt es noch einen Bereich für sonstige Informationen.

Testfallbeschreibung

„DIS_Merkmal_001“

Umfeld*:	DOK				**Nr.:**	001	
Typ*:	automatisch				**Quelle*:**	DIS Testfälle.xls von Herrn Borst	
Status*:	**H42**	**H45**	**H5E**	**H51**	**Priorität*:**	hoch (1)	
eCATT*:	Y_TS_CV01N_MERKMALANZEIGEN						

Testanforderung*

1. Irgendeinen Dokumentinhalt manuell eingeben, dabei das "d" klein schreiben, nach Bestätigung mit Enter muss das "d" in ein großes "D" umgeschrieben werden.
2. Wenn ein Bestimmte Dokumentart-und Inhalt bei Dokument anlegen ausgewählt wird, soll das System folgende Merkmale unter den Zusatzdaten angezeigt.

Ziel*

1. das "d" in ein großes "D" werden nach Bestätigung mit Enter umgeschrieben.

2. die angezeigte Merkmale von D10 und D10.1:

- Berechtigungsgruppe
- Vertraulichkeitsstufe
- Dokumentinhalt
- Mastermaterial
- PEP-Phase aus Materialstamm: automatisch übernehmen
- Basisvariante
- CAD-Erstellungssystem
- Version/Release (CAD-System)
- CAD-Laufumgebung
- CAD-Baugruppe
- LIN
- Musterstand Hella
- Freigegeben/Abgelehnt von.
- Bes.Merkmal(e) (HI-QE1-40-10)

Testvorgang	**Testobjekte**
- Ruf Transaktion CV01N an, so dass die neue Dokumentation angelegt wird	Dokumentinhalt „d10“ eingeben

Vorbedingungen	Nachbedingungen
-	-
Vorgängertestfälle	**Nachfolgetestfälle**
- Prüfe, ob die Transaktion CV01N aufrufbar ist - Prüfe, ob ein Dokumentart D10 im Testdaten existiert	- Prüfe, ob die entsprechenden Werte von den Dokumentinhalt-abhängigen Merkmalen richtig aufgelistet werden.
Eingaben	**Ausgaben**
-	-
Sonstiges (z.B.: Screenshot)	

* Pflichtfelder

Anmerkung:

Testfall-spezifikation muss mit Testskript Dokumentation gut unterscheiden können.

- Testfall-Spezifikation : ziel Testfälle einheitlich erstellen
- Testskript Dokumentation: die Beschreibung der Testskript Ablauf
  Die zwei sollten auch kein groß unterschied sein, da Testskript ist inhaltlich Abbildung von Testfall.

In Gelbfarbe-Einträgen sind nicht nötig. Das entspricht „Testablauf" und „Testergebnisse"

Abbildung 42: Testfall-Beschreibung-DIS_Merkmal_001

Schritt 5: Durchführung: Automation-Modell

Für übersichtlichen Automatisierungsplan zur Testdurchführung wird ein Automation-Modell hier erstellt. Der obere Testfall wird in Original Testdaten MS-Excel Tabelle (aus Abbildung 4.2.4) über 17 Dokumentarte erforderlich (auch in andere 2 Testsystemen) durchgeführt. D.h. dieses entsprechende dazu erstellte Testskript wird 17 Mals aufgerufen um die alle Dokumentart-/Inhalt abhängige Merkmale zu prüfen. Jedes eCATT-skript ist parametrisierbar bezüglich der zu verwendenden Stamm- und Bewegungsdaten als auch bezüglich der SAP Systeminstanzen die durch die Testfälle getestet werden sollen. Dieser Testfall sollte jederzeit wiederholt ausführbar sein. Der entsprechende Test-Workflow wird im folgenden erstellten Automation-Modell abgebildet. Siehe Abbildung 43 Workflow für Dokumentart -und Inhalt abhängige Merkmale prüfen.

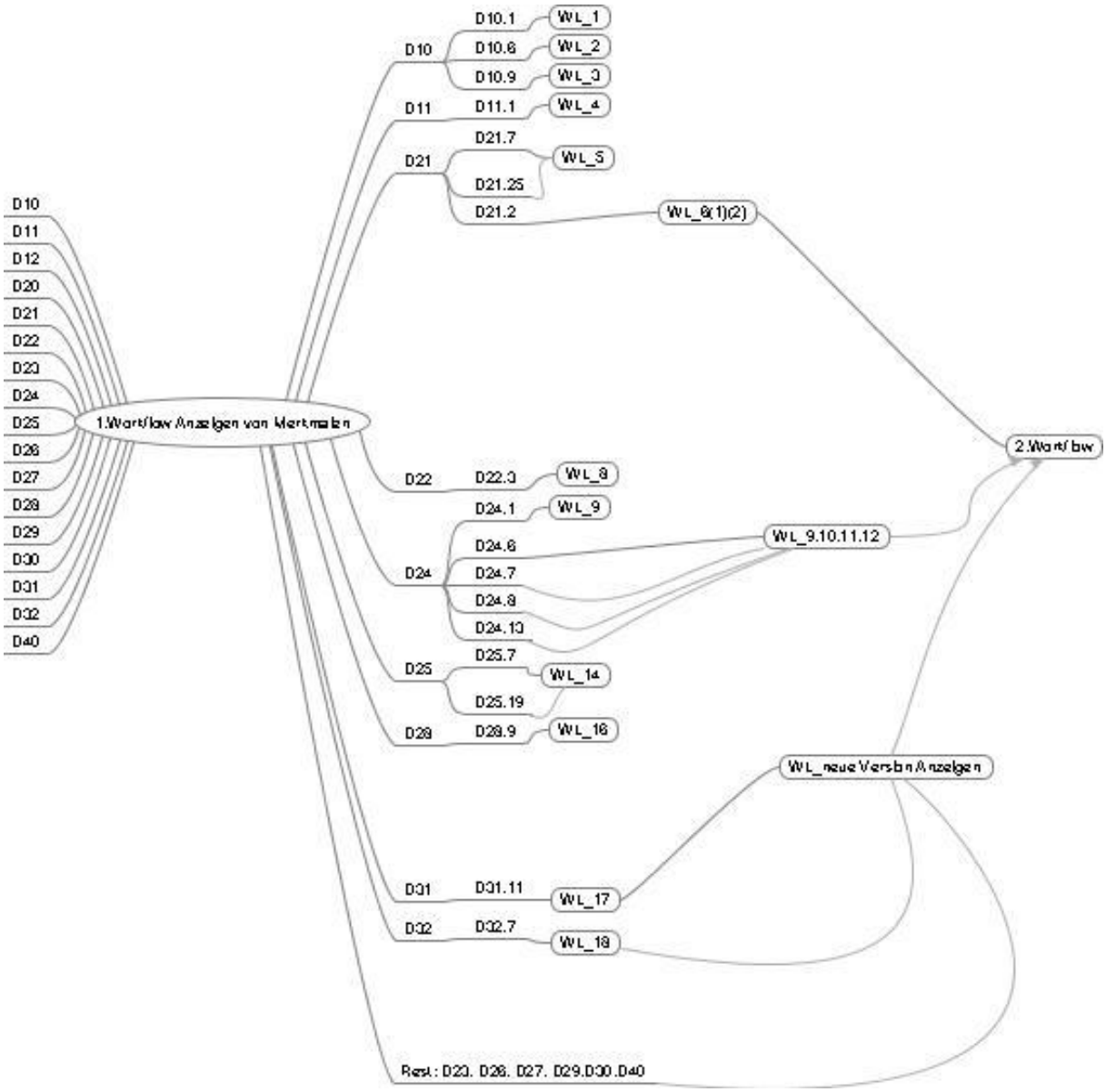


Abbildung 43: Workflow für Dokumentart -/ Inhalt abhängige Merkmale prüfen

Oben 1.Workflow-Modell Anzeigen von Merkmalen als Start Testskript-Zustand, zu testete Dokumentart-/Inhalt als Übergangseingaben. Die anderen Zustand „WL_Nummer“ sind alle Ausgangszustände, die alle nach Durchführung in eine Testfall-Beschreibung mit Testergebnissen und ProtokollNr. ausführlich eingetragen werden. Also die Abbildung 41 Spezifikation für Merkmal Prüfung, wird berücksichtigt und vollständig ausgeführt. Ein Workflow heißt hier ein Testablauf über eine Funktion (entspricht ein Transaktionscode) sowie die beim Dokument anlegen.

Bei Dokumentart D10 Ausführungsergebnis siehe Abbildung 44 Protokoll von Dokumentart D10 in Testsystem H42.

Ins Detail siehe Appendix A: Testskript Y_TS_CV01N_MERKMALANZEIGEN für Überprüfung der Merkmale von Dokumentarten D10

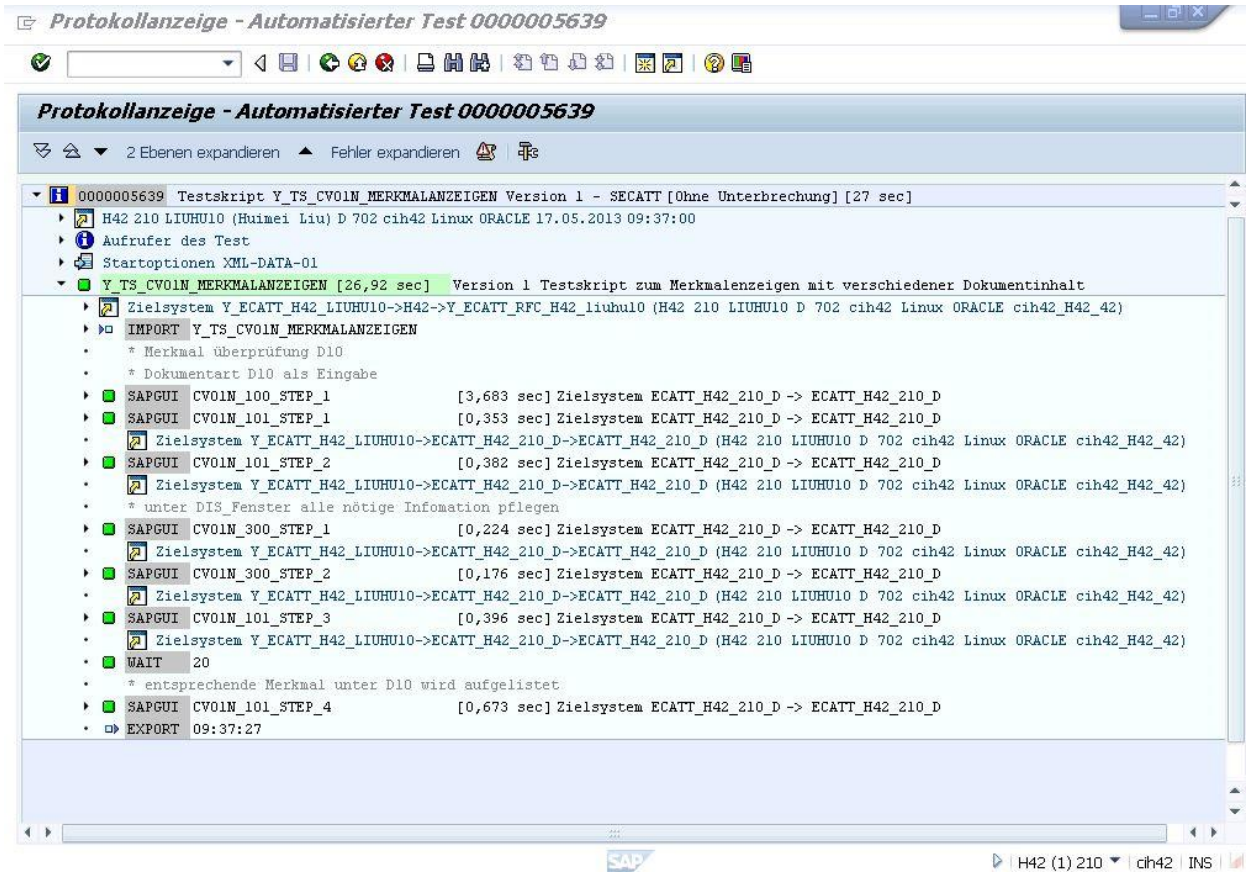


Abbildung 44: Protokoll von Dokumentart D10 in Testsystem H42

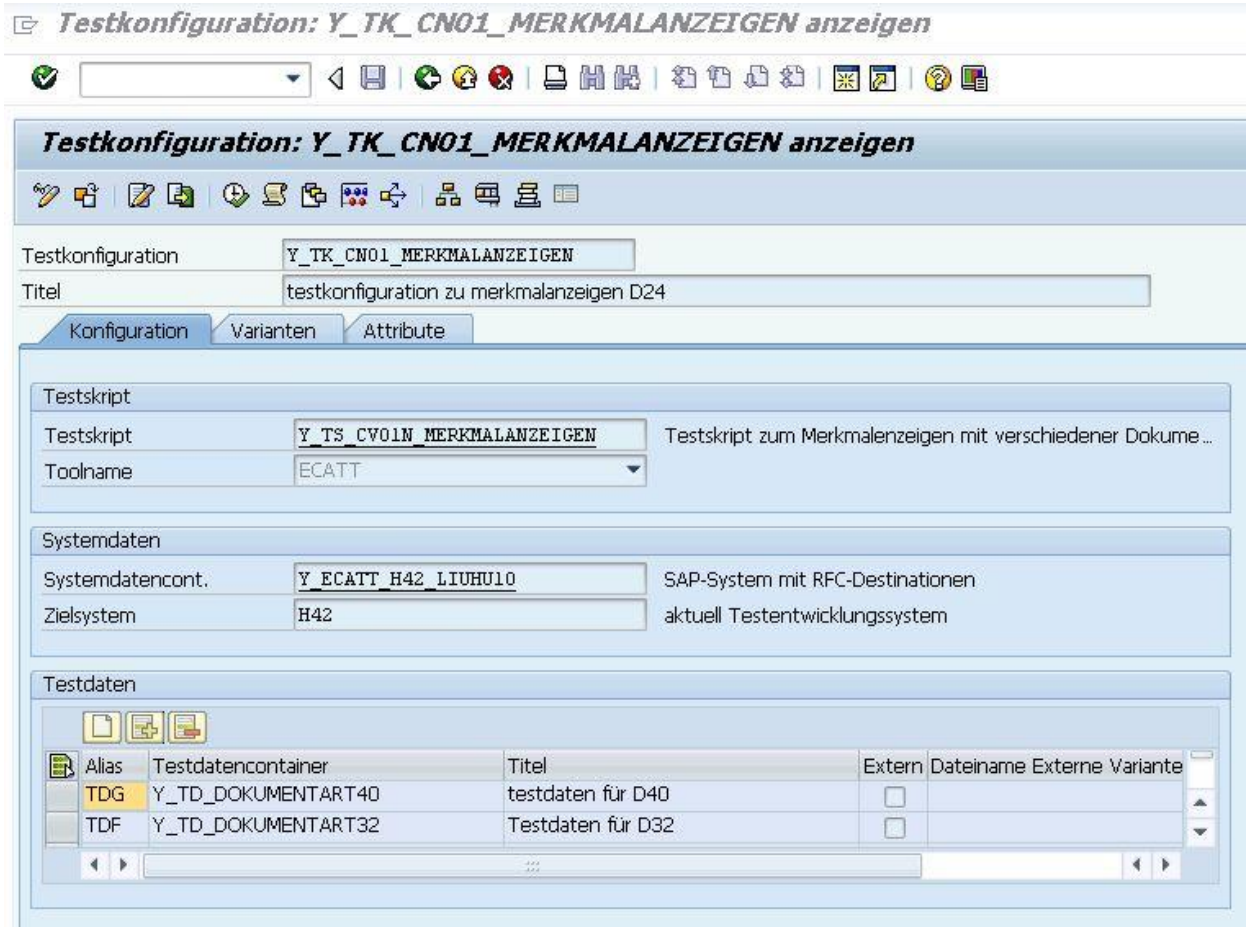


Abbildung 45: Testkonfiguration Y_TK_CN01_MERKMALANZEIGEN

Dazu die letzte Zustände für 2.Workflow: für Merkmal Prüfung bei Status-Wechseln siehe Appendix E.

5 Ergebnisse, Zusammenfassung und zukünftige Arbeit

Zur Veranschaulichung und Prüfung der eingesetzten und weiterentwickelten Verfahren wurden die Ergebnisse durch untere Koordinaten mit dem durchgeführten Testzeitaufwand-Wert aus dem Live-System generiert. In diesem Kapitel werden die Grundzüge vom Testprozess zur Automatisierung zusammenfasst und die relevanten Entwicklungsprozesse für zukünftige Arbeiten geschrieben.

5.1 Ergebnisse

Der obere durchgeführte Workflow für Merkmalsprüfung ist beim manuellen Vorgehen sehr mühsam und langwierig, da es große Menge Dokumentart und Dokumentinhalt gibt. (siehe Appendix F). Ein immer identischer Testvorgang wird getestet. Daher ist die Merkmalsprüfung gut für die Automatisierung geeignet.

Die Ergebnisse zu den Vergleichen der Test-Aufwand werden in folgender Abbildung dargestellt.

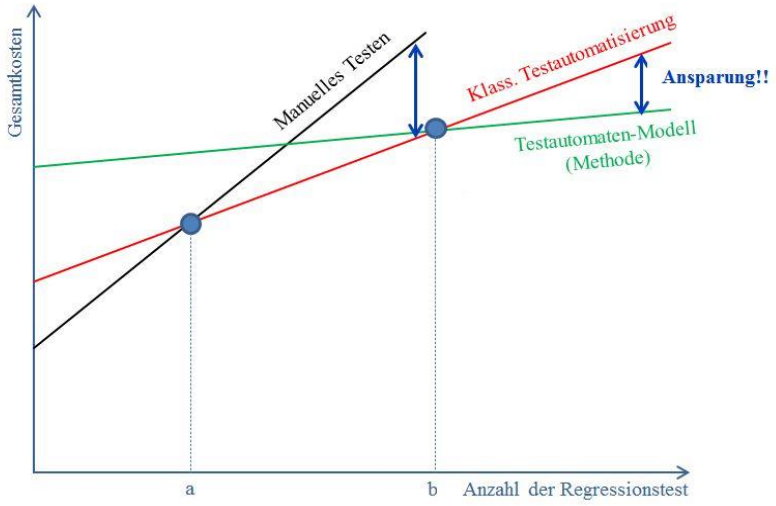


Abbildung 46: Statische Analyse Überblick

Die Abbildung 46 Statische Analyse Überblick zeigt, das Verhältnis von Gesamtkosten (Y-Achse) zu Anzahl der Releases/Regressionstest (X-Achse). Die drei Geraden auf der Abbildung zeigen diesen Zusammenhang für drei Test-Konzept, das manuelle Teste, die klass. Testautomatisierung und das in dieser Arbeit Modellbasiert Testen.

Die schwarze Gerade steht für das rein manuelle Testen, sie hat die geringsten Initialkosten, die gesamtkosten steigen jedoch durch dem hohen Aufwand der manuellen Ausführung. d.h. je mehr Testfälle man durchführt bzw. in je mehr Testsystemen, umso mehr Zeit- und Arbeitsaufwand ist notwendig.

Die zweite Gerade steht für die klassische Testautomatisierung. Der anfänglich höhe Entwicklungsaufwand für die Testautomatisierung steigert die Initialkosten gegenüber den manuellen Test. Zum Beispiel, durch die Testskriptgewinnung bzw. Integration des Testsystems mit Testtools. Nach dem Schnittpunkt der schwarzen und roten Gerade rentiert sich die klassische Testautomatisierung bei steigender Anzahl an Regressionstests, da die wiederholbaren Testskripts häufig durchgeführt wurden. Die grüne Gerade steht für das modellbasierte Testen. Das Testskript wird einheitlich spezifiziert und priorisiert, und mithilfe eines Testautomatisationsmodells werden die Testprozesse effizient durchgeführt. Diese Spezifizierung erfordert ein hohe Erstanpassung, daher sind die Anfangskosten am höchsten, jedoch ist der Anstieg der Kosten im Vergleich zu dem manuellen Testen, sowie zur klassischen Testautomatisierung geringer, da die Testwiederholungen automatisch ablaufen und mit jeder Wiederholung genauer werden. Ebenfalls hat das modellbasierte Testen die geringsten Endgesamtkosten. Der zweite blaue Kreis beschreibt den Schnittpunkt der Geraden des modellbasierten Testens mit der Gerade des klassischen Testautomatisierens.

Bevor man mit einer Durchführung einer Testautomatisierung beginnt, ist eine Analyse wichtig. Das heißt, wann wird es sich lohnen nach dem Vergleich der Gesamtkosten zwischen manuellem und automatisiertem Testen zu unterscheiden.

5.2 Zusammenfassung

Zur Entwicklung einer Vorgehensweise zur Automatisierung wurden im Rahmen dieser Diplomarbeit zwei Grundzüge erfasst. Dadurch werden Vor-und Nachteile zusammengefasst.

Grundzüge 1: Testautomatisierung 1.0 mit eCATT

Die folgende Abbildung 47 Testautomatisierung 1.0 mit eCATT, beschreibt den Arbeitsablauf mit Testautomatisierung 1.0. Der Testvorgang zur Automatisierung wird per Capture/Replay mit der simplen Aufzeichnung der Testschritte moduliert und anschließender Parametrisierung/Konfigurieren durch ein Testtool eCATT.

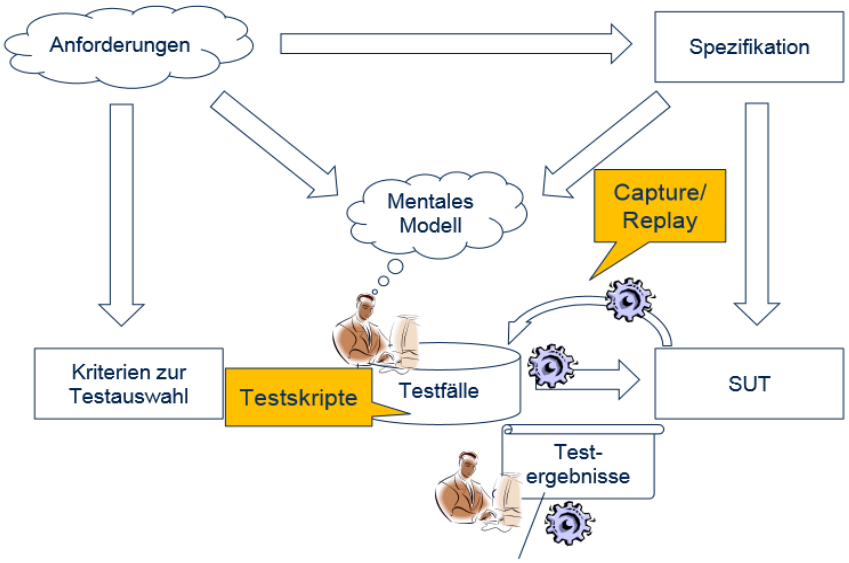


Abbildung 47: Testautomatisierung 1.0 [Guel10] mit eCATT

Grundzüge 2: Testautomatisierung 2.0 (Testautomaten-Modell)

In dieser Diplomarbeit wird mit Hilfe eines Modells organisiert, also Modellbasierter Test zum Automatisierung Test. Siehe Abbildung 48 Testautomatisierung 2.0.

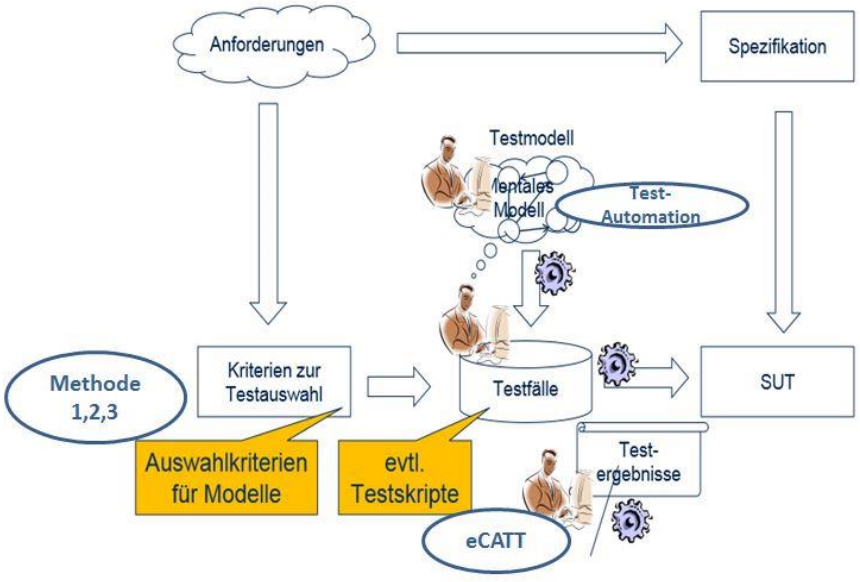


Abbildung 48: Testautomatisierung 2.0 (Erweiterung von [Guel10])

Über detaillierte Kriterien für Modelle zur Testauswahl und evtl. Testskripte für Modelle wird drei Methode zur Analyse der Testdaten implementiert. Tests sind immer nur Stichproben, und der Testaufwand ist deshalb nach Risiko und Prioritäten zu steuern.

Vor- und Nachteile unter verschiedenen Testkonzepten zur Testautomatisierung.

Aktivität	Manuelles Testen	Testautomatisierung 1.0	Testautomatisierung 2.0
Erstellung	(-) Nicht immer systematisch (-) Aufwendig und teuer (-) Fehleranfällig	(-) Nicht immer systematisch (-) Aufwendig und teuer (-) Fehleranfällig	(+) Systematisch (+) Schnell und Wiederholbar (+) Präzise (+) Bessere Abdeckung
Durchführung	(-) Aufwendig und teuer (-) Fehleranfällig	(+) Schnell (+) Präzise (-) Nicht wiederholbar bei Änderungen	(+) Schnell (+) Präzise (+) Anpassbar bei Änderungen
Auswertung	(-) Aufwendig und teuer (-) Fehleranfällig	(~) Schnell (~) Präzise	(~) Schnell (~) Präzise
Controlling	(-) Überdeckung/Nachverfolgbarkeit schwierig	(-) Überdeckung/Nachverfolgbarkeit schwierig	(+) Überdeckungsanalyse möglich (+) Nachverfolgbarkeit möglich (+) Frühe Fehlererkennung

Tabelle 10: Vorteile und Nachteile von Einsatz Modellbasiert Testkonzept [Guel10]

5.3 Zukünftige Arbeit

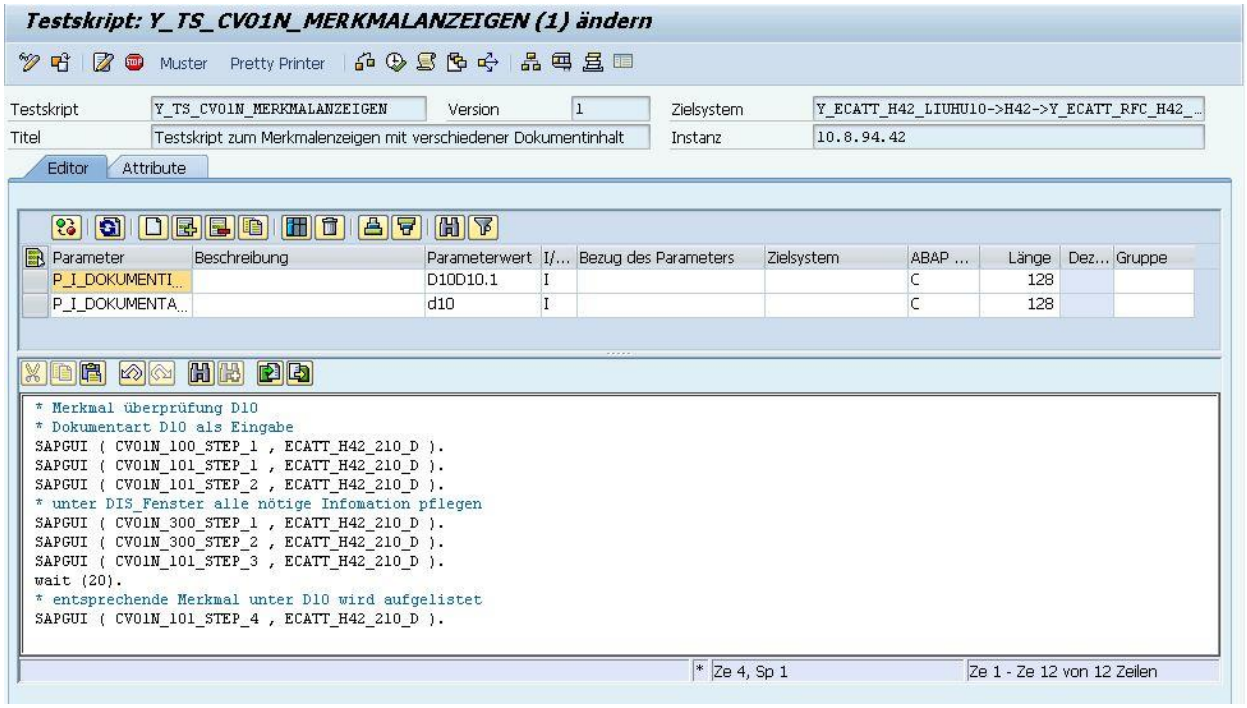
Die im Rahmen dieser Diplomarbeit entwickelte Vorgehensweise zur Testautomatisierung mit Hilfe des Testtool eCATT kann auch für andere Testzwecke verwendet werden, aber einzelne Phasen laufen in der Theorie nacheinander ab, in der Praxis sind jedoch Rückschritte oft unvermeidlich. Eine mögliche Lösung für dieses Abfolge-Problem ist in der Weiterentwicklungsphase von neuen Testprozessen zu finden.

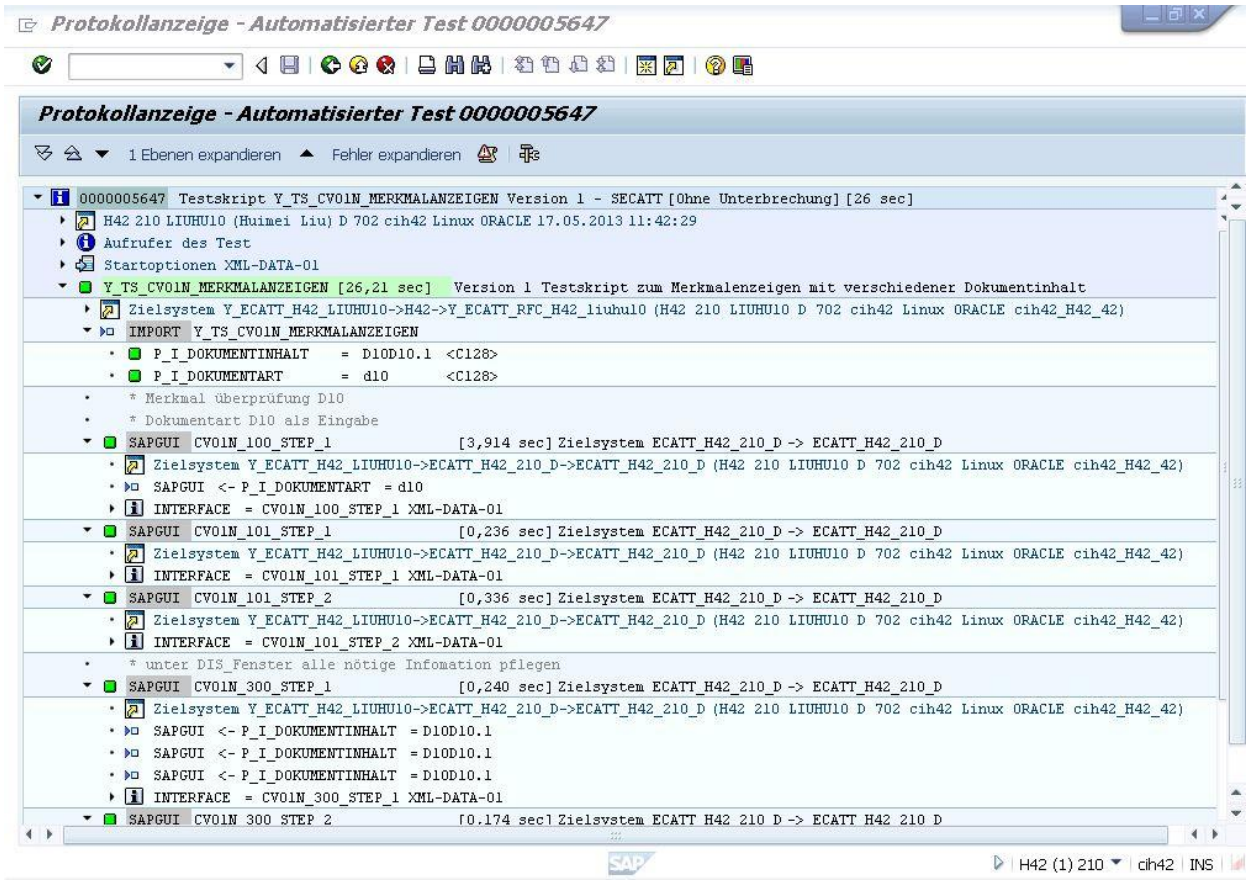
Die Testskripts werden nach einer bestimmten Reihenfolge durchgeführt, dies ist ein wichtiger Schritt um eine effiziente Testautomatisierung durchzuführen. Eigen definierte Priorisierungskriterien sollten auf einem gut überlegten Wege stattfinden, dadurch erhält man ein besseres Verständnis der Kundenanforderungen, wie zum Beispiel in der Test-Entwicklungsphase in welche richtig weiter aufgebaut werden soll und in welche nicht.

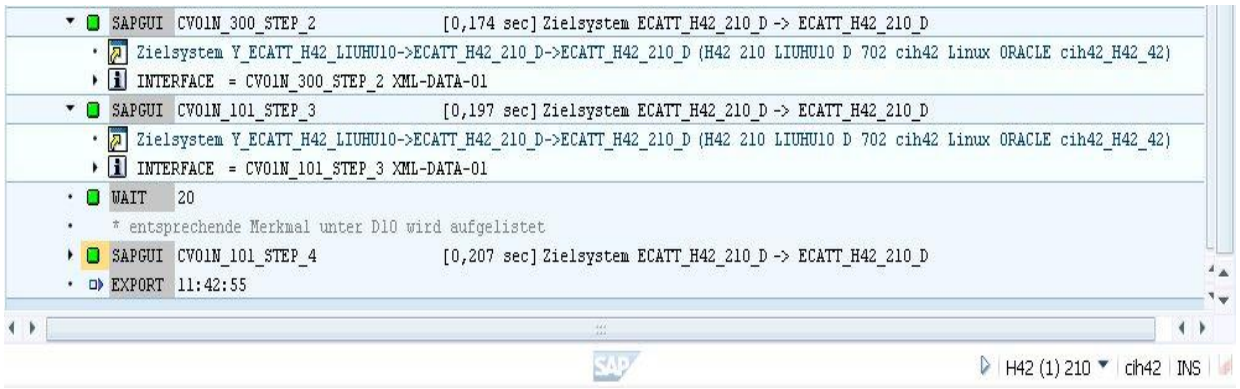
eCATT wird von der SAP AG auch in Zukunft weiter entwickelt werden. D.h., dass noch weitere Funktionalitäten hinzukommen werden, die nicht selbst programmiert werden müssen. Testfälle, Testfallkataloge, Testpakete und insbesondere eCATT-Skripte können für zukünftige Projekte wiederverwendet werden, gerade für Regressionstests in Releasewechsel-Projekten ist die eine interessante Option.

Appendix A

Testskript Y_TS_CV01N_MERKMALANZEIGEN

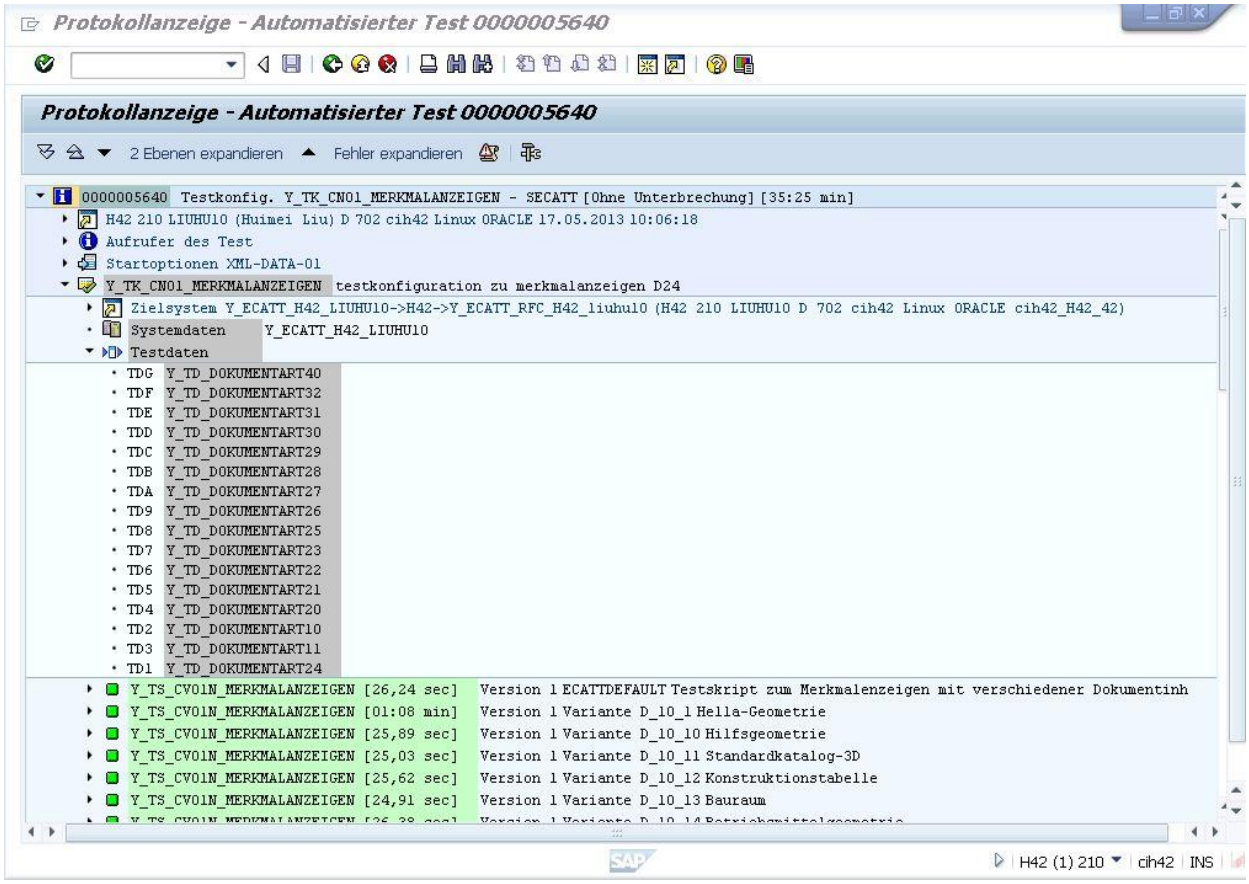


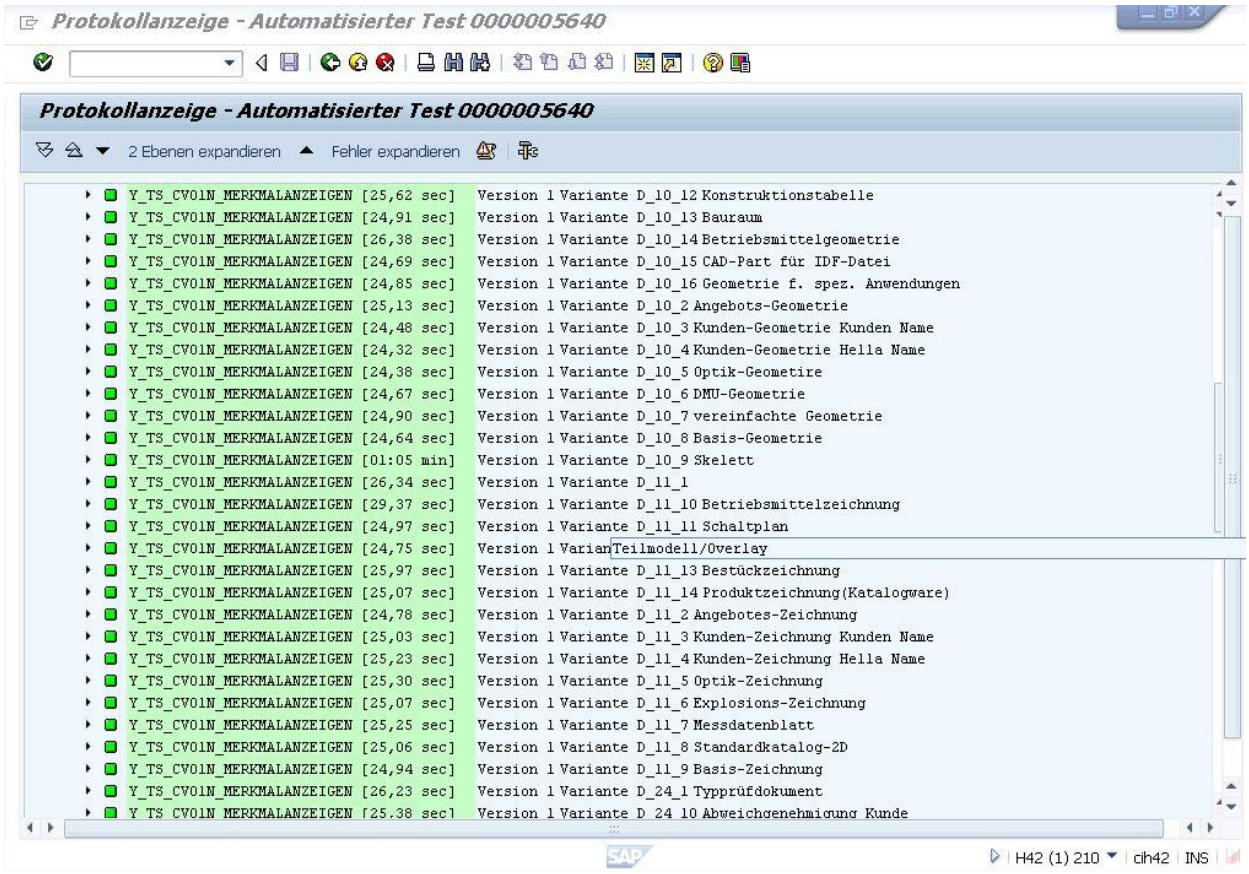


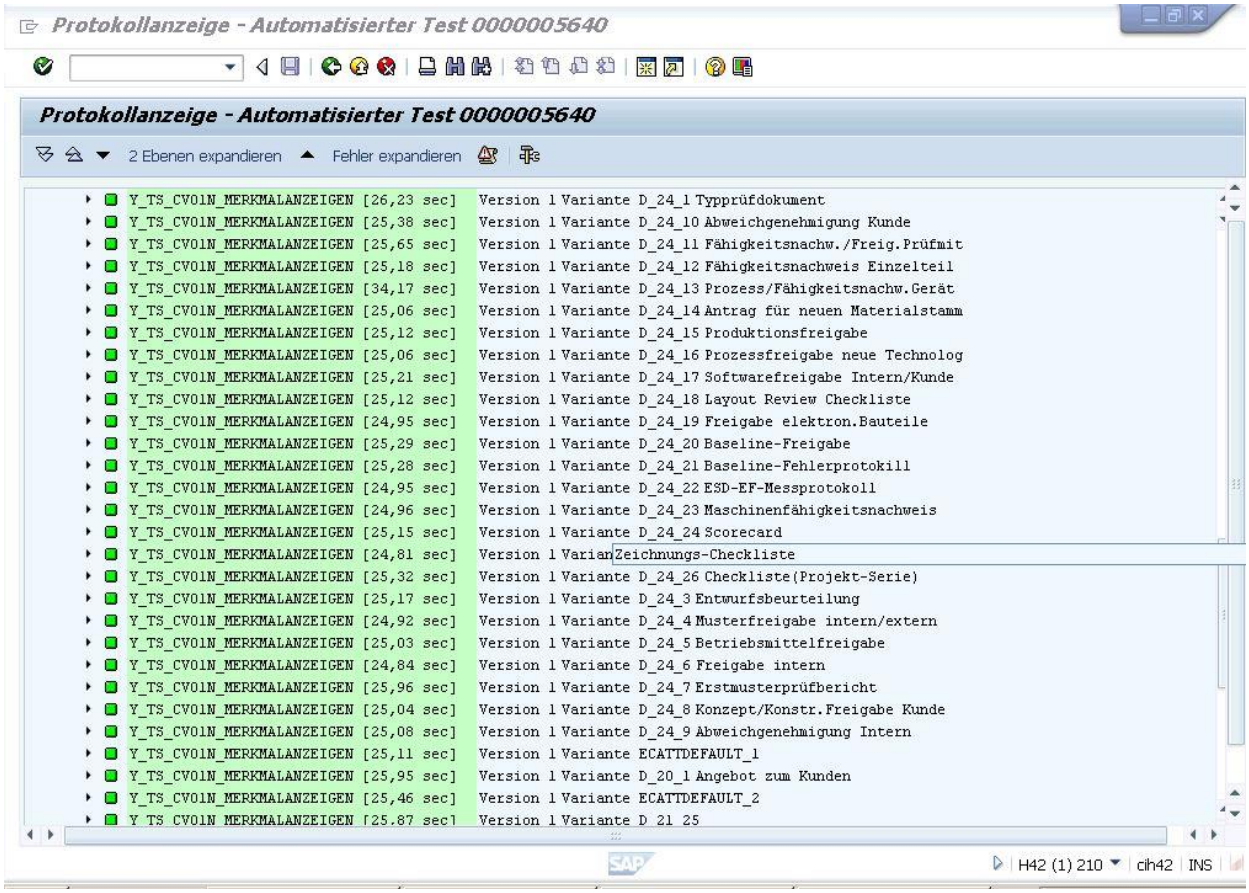

SAPGUI CV01N_300_STEP_2 [0,174 sec] Zielsystem ECATT_H42_210_D -> ECATT_H42_210_D
Zielsystem Y_ECATT_H42_LIUHU10->ECATT_H42_210_D->ECATT_H42_210_D (H42 210 LIUHU10 D 702 cih42 Linux ORACLE cih42_H42_42)
INTERFACE = CV01N_300_STEP_2 XML-DATA-01
SAPGUI CV01N_101_STEP_3 [0,197 sec] Zielsystem ECATT_H42_210_D -> ECATT_H42_210_D
Zielsystem Y_ECATT_H42_LIUHU10->ECATT_H42_210_D->ECATT_H42_210_D (H42 210 LIUHU10 D 702 cih42 Linux ORACLE cih42_H42_42)
INTERFACE = CV01N_101_STEP_3 XML-DATA-01
WAIT 20
* entsprechende Merkmal unter D10 wird aufgelistet
SAPGUI CV01N_101_STEP_4 [0,207 sec] Zielsystem ECATT_H42_210_D -> ECATT_H42_210_D
EXPORT 11:42:55
H42 (1) 210
cih42
INS

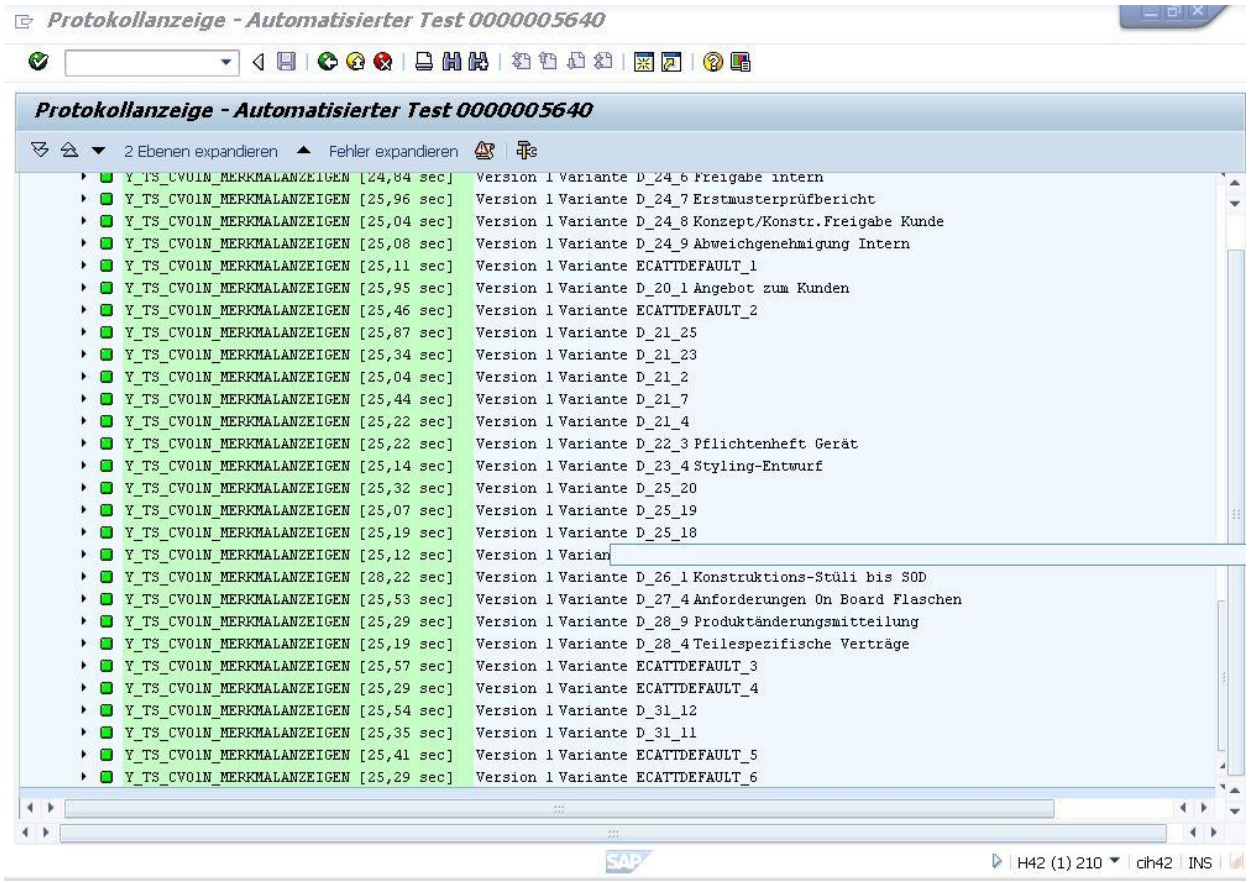
Appendix B

Testkonfiguration Y_TK_CN01_MERKMALANZEIGEN




Protokollanzeige - Automatisierter Test 0000005640
Protokollanzeige - Automatisierter Test 0000005640
2 Ebenen expandieren
Fehler expandieren
Y_TS_CV01N_MERKMALANZEIGEN [25,62 sec] Version 1 Variante D_10_12 Konstruktionstabelle
Y_TS_CV01N_MERKMALANZEIGEN [24,91 sec] Version 1 Variante D_10_13 Bauraum
Y_TS_CV01N_MERKMALANZEIGEN [26,38 sec] Version 1 Variante D_10_14 Betriebsmittelgeometrie
Y_TS_CV01N_MERKMALANZEIGEN [24,69 sec] Version 1 Variante D_10_15 CAD-Part für IDF-Datei
Y_TS_CV01N_MERKMALANZEIGEN [24,85 sec] Version 1 Variante D_10_16 Geometrie f. spez. Anwendungen
Y_TS_CV01N_MERKMALANZEIGEN [25,13 sec] Version 1 Variante D_10_2 Angebots-Geometrie
Y_TS_CV01N_MERKMALANZEIGEN [24,48 sec] Version 1 Variante D_10_3 Kunden-Geometrie Kunden Name
Y_TS_CV01N_MERKMALANZEIGEN [24,32 sec] Version 1 Variante D_10_4 Kunden-Geometrie Hella Name
Y_TS_CV01N_MERKMALANZEIGEN [24,38 sec] Version 1 Variante D_10_5 Optik-Geometire
Y_TS_CV01N_MERKMALANZEIGEN [24,67 sec] Version 1 Variante D_10_6 DMU-Geometrie
Y_TS_CV01N_MERKMALANZEIGEN [24,90 sec] Version 1 Variante D_10_7 vereinfachte Geometrie
Y_TS_CV01N_MERKMALANZEIGEN [24,64 sec] Version 1 Variante D_10_8 Basis-Geometrie
Y_TS_CV01N_MERKMALANZEIGEN [01:05 min] Version 1 Variante D_10_9 Skelett
Y_TS_CV01N_MERKMALANZEIGEN [26,34 sec] Version 1 Variante D_11_1
Y_TS_CV01N_MERKMALANZEIGEN [29,37 sec] Version 1 Variante D_11_10 Betriebsmittelzeichnung
Y_TS_CV01N_MERKMALANZEIGEN [24,97 sec] Version 1 Variante D_11_11 Schaltplan
Y_TS_CV01N_MERKMALANZEIGEN [24,75 sec] Version 1 Varian Teilmodell/Overlay
Y_TS_CV01N_MERKMALANZEIGEN [25,97 sec] Version 1 Variante D_11_13 Bestückzeichnung
Y_TS_CV01N_MERKMALANZEIGEN [25,07 sec] Version 1 Variante D_11_14 Produktzeichnung(Katalogware)
Y_TS_CV01N_MERKMALANZEIGEN [24,78 sec] Version 1 Variante D_11_2 Angebotes-Zeichnung
Y_TS_CV01N_MERKMALANZEIGEN [25,03 sec] Version 1 Variante D_11_3 Kunden-Zeichnung Kunden Name
Y_TS_CV01N_MERKMALANZEIGEN [25,23 sec] Version 1 Variante D_11_4 Kunden-Zeichnung Hella Name
Y_TS_CV01N_MERKMALANZEIGEN [25,30 sec] Version 1 Variante D_11_5 Optik-Zeichnung
Y_TS_CV01N_MERKMALANZEIGEN [25,07 sec] Version 1 Variante D_11_6 Explosions-Zeichnung
Y_TS_CV01N_MERKMALANZEIGEN [25,25 sec] Version 1 Variante D_11_7 Messdatenblatt
Y_TS_CV01N_MERKMALANZEIGEN [25,06 sec] Version 1 Variante D_11_8 Standardkatalog-2D
Y_TS_CV01N_MERKMALANZEIGEN [24,94 sec] Version 1 Variante D_11_9 Basis-Zeichnung
Y_TS_CV01N_MERKMALANZEIGEN [26,23 sec] Version 1 Variante D_24_1 Typprüfdokument
Y_TS_CV01N_MERKMALANZEIGEN [25,38 sec] Version 1 Variante D_24_10 Abweichgenehmigung Kunde
H42 (1) 210
cih42
INS


Protokollanzeige - Automatisierter Test 0000005640
Protokollanzeige - Automatisierter Test 0000005640
2 Ebenen expandieren
Fehler expandieren
Y_TS_CV01N_MERKMALANZEIGEN [26,23 sec] Version 1 Variante D_24_1 Typprüfdokument
Y_TS_CV01N_MERKMALANZEIGEN [25,38 sec] Version 1 Variante D_24_10 Abweichgenehmigung Kunde
Y_TS_CV01N_MERKMALANZEIGEN [25,65 sec] Version 1 Variante D_24_11 Fähigkeitsnachw./Freig.Prüfmit
Y_TS_CV01N_MERKMALANZEIGEN [25,18 sec] Version 1 Variante D_24_12 Fähigkeitsnachweis Einzelteil
Y_TS_CV01N_MERKMALANZEIGEN [34,17 sec] Version 1 Variante D_24_13 Prozess/Fähigkeitsnachw.Gerät
Y_TS_CV01N_MERKMALANZEIGEN [25,06 sec] Version 1 Variante D_24_14 Antrag für neuen Materialstamm
Y_TS_CV01N_MERKMALANZEIGEN [25,12 sec] Version 1 Variante D_24_15 Produktionsfreigabe
Y_TS_CV01N_MERKMALANZEIGEN [25,06 sec] Version 1 Variante D_24_16 Prozessfreigabe neue Technolog
Y_TS_CV01N_MERKMALANZEIGEN [25,21 sec] Version 1 Variante D_24_17 Softwarefreigabe Intern/Kunde
Y_TS_CV01N_MERKMALANZEIGEN [25,12 sec] Version 1 Variante D_24_18 Layout Review Checkliste
Y_TS_CV01N_MERKMALANZEIGEN [24,95 sec] Version 1 Variante D_24_19 Freigabe elektron.Bauteile
Y_TS_CV01N_MERKMALANZEIGEN [25,29 sec] Version 1 Variante D_24_20 Baseline-Freigabe
Y_TS_CV01N_MERKMALANZEIGEN [25,28 sec] Version 1 Variante D_24_21 Baseline-Fehlerprotokill
Y_TS_CV01N_MERKMALANZEIGEN [24,95 sec] Version 1 Variante D_24_22 ESD-EF-Messprotokoll
Y_TS_CV01N_MERKMALANZEIGEN [24,96 sec] Version 1 Variante D_24_23 Maschinenfähigkeitsnachweis
Y_TS_CV01N_MERKMALANZEIGEN [25,15 sec] Version 1 Variante D_24_24 Scorecard
Y_TS_CV01N_MERKMALANZEIGEN [24,81 sec] Version 1 Varian Zeichnungs-Checkliste
Y_TS_CV01N_MERKMALANZEIGEN [25,32 sec] Version 1 Variante D_24_26 Checkliste(Projekt-Serie)
Y_TS_CV01N_MERKMALANZEIGEN [25,17 sec] Version 1 Variante D_24_3 Entwurfsbeurteilung
Y_TS_CV01N_MERKMALANZEIGEN [24,92 sec] Version 1 Variante D_24_4 Musterfreigabe intern/extern
Y_TS_CV01N_MERKMALANZEIGEN [25,03 sec] Version 1 Variante D_24_5 Betriebsmittelfreigabe
Y_TS_CV01N_MERKMALANZEIGEN [24,84 sec] Version 1 Variante D_24_6 Freigabe intern
Y_TS_CV01N_MERKMALANZEIGEN [25,96 sec] Version 1 Variante D_24_7 Erstmusterprüfbericht
Y_TS_CV01N_MERKMALANZEIGEN [25,04 sec] Version 1 Variante D_24_8 Konzept/Konstr.Freigabe Kunde
Y_TS_CV01N_MERKMALANZEIGEN [25,08 sec] Version 1 Variante D_24_9 Abweichgenehmigung Intern
Y_TS_CV01N_MERKMALANZEIGEN [25,11 sec] Version 1 Variante ECATTDEFAULT_1
Y_TS_CV01N_MERKMALANZEIGEN [25,95 sec] Version 1 Variante D_20_1 Angebot zum Kunden
Y_TS_CV01N_MERKMALANZEIGEN [25,46 sec] Version 1 Variante ECATTDEFAULT_2
Y_TS_CV01N_MERKMALANZEIGEN [25,87 sec] Version 1 Variante D_21_25
H42 (1) 210
cih42
INS


Protokollanzeige - Automatisierter Test 0000005640
Protokollanzeige - Automatisierter Test 0000005640
2 Ebenen expandieren
Fehler expandieren
Y_TS_CV01N_MERKMALANZEIGEN [24,84 sec] Version 1 Variante D_24_6 Freigabe intern
Y_TS_CV01N_MERKMALANZEIGEN [25,96 sec] Version 1 Variante D_24_7 Erstmusterprüfbericht
Y_TS_CV01N_MERKMALANZEIGEN [25,04 sec] Version 1 Variante D_24_8 Konzept/Konstr.Freigabe Kunde
Y_TS_CV01N_MERKMALANZEIGEN [25,08 sec] Version 1 Variante D_24_9 Abweichgenehmigung Intern
Y_TS_CV01N_MERKMALANZEIGEN [25,11 sec] Version 1 Variante ECATTDEFAULT_1
Y_TS_CV01N_MERKMALANZEIGEN [25,95 sec] Version 1 Variante D_20_1 Angebot zum Kunden
Y_TS_CV01N_MERKMALANZEIGEN [25,46 sec] Version 1 Variante ECATTDEFAULT_2
Y_TS_CV01N_MERKMALANZEIGEN [25,87 sec] Version 1 Variante D_21_25
Y_TS_CV01N_MERKMALANZEIGEN [25,34 sec] Version 1 Variante D_21_23
Y_TS_CV01N_MERKMALANZEIGEN [25,04 sec] Version 1 Variante D_21_2
Y_TS_CV01N_MERKMALANZEIGEN [25,44 sec] Version 1 Variante D_21_7
Y_TS_CV01N_MERKMALANZEIGEN [25,22 sec] Version 1 Variante D_21_4
Y_TS_CV01N_MERKMALANZEIGEN [25,22 sec] Version 1 Variante D_22_3 Pflichtenheft Gerät
Y_TS_CV01N_MERKMALANZEIGEN [25,14 sec] Version 1 Variante D_23_4 Styling-Entwurf
Y_TS_CV01N_MERKMALANZEIGEN [25,32 sec] Version 1 Variante D_25_20
Y_TS_CV01N_MERKMALANZEIGEN [25,07 sec] Version 1 Variante D_25_19
Y_TS_CV01N_MERKMALANZEIGEN [25,19 sec] Version 1 Variante D_25_18
Y_TS_CV01N_MERKMALANZEIGEN [25,12 sec] Version 1 Varian
Y_TS_CV01N_MERKMALANZEIGEN [28,22 sec] Version 1 Variante D_26_1 Konstruktions-Stüli bis SOD
Y_TS_CV01N_MERKMALANZEIGEN [25,53 sec] Version 1 Variante D_27_4 Anforderungen On Board Flaschen
Y_TS_CV01N_MERKMALANZEIGEN [25,29 sec] Version 1 Variante D_28_9 Produktänderungsmitteilung
Y_TS_CV01N_MERKMALANZEIGEN [25,19 sec] Version 1 Variante D_28_4 Teilespezifische Verträge
Y_TS_CV01N_MERKMALANZEIGEN [25,57 sec] Version 1 Variante ECATTDEFAULT_3
Y_TS_CV01N_MERKMALANZEIGEN [25,29 sec] Version 1 Variante ECATTDEFAULT_4
Y_TS_CV01N_MERKMALANZEIGEN [25,54 sec] Version 1 Variante D_31_12
Y_TS_CV01N_MERKMALANZEIGEN [25,35 sec] Version 1 Variante D_31_11
Y_TS_CV01N_MERKMALANZEIGEN [25,41 sec] Version 1 Variante ECATTDEFAULT_5
Y_TS_CV01N_MERKMALANZEIGEN [25,29 sec] Version 1 Variante ECATTDEFAULT_6
H42 (1) 210
cih42
INS

Appendix C

Spezifikation von Testskript Y_TS_CV01N_MERKMALANZEIGEN

<table>
<tr><th colspan="7">Testfallbeschreibung

„DIS_Merkmal_001“</th></tr>
<tr><td>Umfeld*:</td><td colspan="4">DOK</td><td>Nr.:</td><td>001</td></tr>
<tr><td>Typ*:</td><td colspan="4">automatisch</td><td>Quelle*:</td><td>DIS Testfälle.xls von Herrn Borst</td></tr>
<tr><td>Status*:</td><td>H42</td><td>H45</td><td>H5E</td><td>H51</td><td>Priorität*:</td><td>hoch (1)</td></tr>
<tr><td>eCATT*:</td><td colspan="5">Y_TS_CV01N_MERKMALANZEIGEN</td></tr>
<tr><td colspan="7">Testanforderung*</td></tr>
<tr><td colspan="7">1. Dokumentinhalt D10 manuell eingeben, dabei das "d" klein schreiben, nach Bestätigung mit Enter muss das "d" in ein großes "D" umgeschrieben werden.
2. Wenn ein Dokumentart-und Inhalt D10 und D10.1 bei Dokument anlegen ausgewählt wird, soll das System folgende Merkmale unter den Zusatzdaten angezeigt.</td></tr>
<tr><td colspan="7">Ziel*</td></tr>
<tr><td colspan="7">1. das "d" in ein großes "D" werden nach Bestätigung mit Enter umgeschrieben.

2. die angezeigte Merkmale von D10 und D10.1:

- Berechtigungsgruppe

- Vertraulichkeitsstufe

- Dokumentinhalt

- Mastermaterial

- PEP-Phase aus Materialstamm: automatisch übernehmen

- Basisvariante</td></tr>
</table>

- CAD-Erstellungssystem

- Version/Release (CAD-System)

- CAD-Laufumgebung

- CAD-Baugruppe

- LIN

- Musterstand HELLA

- Freigegeben/Abgelehnt von.

- Bes.Merkmal(e) (HI-QE1-40-10)

Testvorgang	**Testobjekte**
- Ruf Transaktion CV01N an, so dass die neue Dokumentation angelegt wird	Dokumentinhalt „d10“ eingeben
Vorbedingungen	**Nachbedingungen**
- Berechtigung für Dokument anlegen eingerichtet	-
Vorgängertestfälle	**Nachfolgetestfälle**
- Prüfe, ob die Transaktion CV01N aufrufbar ist - Prüfe, ob ein Dokumentart D10 im Testdaten existiert	- Prüfe, ob die entsprechenden Werte von den Dokumentinhalt-abhängigen Merkmalen richtig aufgelistet werden.
Eingaben	**Ausgaben**
-	-

Sonstiges (z.B.: Screenshot)

* Pflichtfelder

Anmerkung:

Testfall-Spezifikation muss mit Testskript Dokumentation gut unterscheiden werden können.

- Testfall-Spezifikation : Ziel Testfälle einheitlich erstellen
- Testskript Dokumentation: die Beschreibung der Testskript Ablauf
  Die zwei sollten auch nicht groß unterschiedlich sein, da das Testskript inhaltlich eineAbbildung von Testfall ist.

In Gelbfarbe-Einträgen sind nicht nötig. Das entspricht „Testablauf“ und „Testergebnisse“

Appendix D

Dokumentart und Dokumentinhalt

D10	Geometrien	D21	Qualitätsdokumente	D23	Daten vom Kunden	D25	Prozessplanung	D30	Projektmanagement
D10.1	Hella-Geometrie	D21.1	Produkt-FMEA	D23.3	Anfrage vom Kunden	D25.1	Produktgespräch	D30.1	Änderungsdokumentation/LoP-Exp
D10.2	Angebots-Geometrie	D21.2	Status-Report f.Kund.,z.B.APQP	D23.4	Styling-Entwurf	D25.2	Ablaufplan	D30.2	Auflassung/Stückzahlplanung
D10.3	Kunden-Geometrie Kunden Name	D21.3	Internes Prozessaudit Full Run	D23.5	wichtiger Schriftv./ Protokoll	D25.3	Fertigungs- / Montagelayout	D30.3	Kapazitätsplanung Projektteam
D10.4	Kunden-Geometrie Hella Name	D21.4	Control Plan	D23.6	Allg. Beding. & Vereinbarungen	D25.4	Zusammenbauuntersuchung	D30.4	Maßnahmenplan/LoP intern
D10.5	Optik-Geometrie	D21.5	PMR Reporting	D23.7	Logistik Vereinbarung Kunde	D25.5	Arbeitsbeschreibung (Produkt)	D30.5	Patentdokumente
D10.6	DMU-Geometrie	D21.6	DVP&R,Testplan,Produktreifegrd	D23.8	Verpackungsvereinbarung	D25.6	Ergebnisdokum.Ablaufsimulation	D30.6	Projektauftrag/ Zielvereinbar.
D10.7	vereinfachte Geometrie	D21.7	Testreport / Prüfbericht	D23.9	Gewährleistungsvereinbarung	D25.7	Verpackungsdatenblatt	D30.7	Projektterminplan
D10.8	Basis-Geometrie	D21.9	Fertigungsflussdiagramm	D23.10	EDI Vereinbarung	D25.8	Vorgaben Prozessdaten	D30.8	Risikoanalyse
D10.9	Skelett	D21.10	Prozessreifegrad	D23.11	Kapazitätsvereinbarung	D25.9	Wartung/Reinigung AP/Montlinie	D30.9	Anforderungsprofil
D10.10	Hilfsgeometrie	D21.11	Internes Produktaudit	**D27**	**Software**	D25.10	Schulungs-/Personaleinsatzplan	D30.10	Vertraulichkeitsvereinbarung
D10.11	Standardkatalog-3D	D21.12	Externes Prozessaudit/R@R	D27.1	HMF / BIN	D25.11	Kapazitätsplan. Betriebsmittel	D30.11	Projekthandbuch/ Kick-Off
D10.12	Konstruktionstabelle	D21.13	Prozess-FMEA	D27.2	HMF (MKS/PDM-Schnittst.)	D25.12	Bedarfsmitteilung Einkauf	D30.12	Aufwandsermittlung
				D27.3	Software (MKS/PDM-Schnittst.)	D25.13	Funktionsprüfung Betriebsm.	D30.13	Arbeitspaketbeschreibung
D10.13	Bauraum	D21.14	Design Review Dokumente	D27.4	Anforderungen On Board Flaschen	D25.14	Betriebsanleitung Betriebsm.	D30.14	Releaseplanung S/H
D10.14	Betriebsmittelgeometrie	D21.15	Dokumentationsplan DmbA	D27.5	Prüfsoftware	D25.15	Verschleißteilliste	D30.15	Konfigurationsplan
D10.15	CAD-Part für IDF-Datei	D21.16	Fehlerbaum / FTA	**D24**	**Freigabedokumente**	D25.16	Kalibrieranweis./ -vorschrift	D30.16	Design to Cost
D10.16	Geometrie f.spez.Anwendungen	D21.17	Lebenslauf Einzelteil	D24.1	Typprüfdokument	D25.17	Bestückdaten	D30.17	Interner Vertrag
D11	**Zeichnungen**	D21.18	Lebenslauf Gerät	D24.3	Entwurfsbeurteilung	D25.18	Herstelldaten Leiterplatte	**D31**	**Non-CAD-Simulationen**
D11.1	Hella-Zeichnung	D21.19	Musterdokumentation	D24.4	Musterfreigabe intern/ extern	D25.19	Schablonendaten Leiterplatte	D31.1	Simulation Survey
D11.2	Angebots-Zeichnung	D21.20	Normenübersicht/Normenbaum	D24.5	Betriebsmittelfreigabe	D25.20	Prüf-und Programmierdaten LP	D31.2	CAL Steuerdaten
D11.3	Kunden-Zeichnung Kunden Name	D21.21	Prüfanweisung	D24.6	Freigabe intern	**D26**	**Stücklisten**	D31.3	CAL Geometrie-Rohdaten
D11.4	Kunden-Zeichnung Hella Name	D21.22	Prüfergebnisse Endprüfung TtM	D24.7	Erstmusterprüfbericht	D26.1	Konstruktions-Stüli bis SOD	D31.4	CAL Simulation
D11.5	Optik-Zeichnung	D21.23	Prüfprotokoll / Messdaten	D24.8	Konzept/Konstr. Freigabe Kunde	D26.2	Betriebsmittelstückliste	D31.5	Strömungssimulation / CFD
D11.6	Explosions-Zeichnung	D21.24	Prüfung Einzelteil	D24.9	Abweichgenehmigung Intern	**D28**	**Lieferanten Daten**	D31.6	Mechanik Simulation / FEM
D11.7	Messdatenblatt	D21.25	Qualitätsvereinbarung m. Kunde	D24.10	Abweichgenehmigung Kunde	D28.1	Angebot vom Lieferanten	D31.7	Spritzgießsimulation
D11.8	Standardkatalog-2D	D21.26	Requalifizierungsprüfung Gerät	D24.11	Fähigkeitsnachw./Freig.Prüfmit	D28.2	Mat-Spezifikation Lief.bezogen	D31.8	Toleranzanalyse
D11.9	Basis-Zeichnung	D21.27	Schwachstellenanalyse Labor	D24.12	Fähigkeitsnachweis Einzelteil	D28.3	Sourc.Com.Prot./Angeb.Auswert.	D31.9	Materialfluss Simulation
D11.10	Betriebsmittelzeichnung	D21.28	SPC-Daten (procella/qs-stat)	D24.13	Prozess/Fähigkeitsnachw. Gerät	D28.4	Teilespezifische Verträge	D31.10	Verhaltenssimulation
D11.11	Schaltplan	D21.29	Testspezifikation	D24.14	Antrag für neuen Materialstamm	D28.5	Projektstatus Lieferant	D31.11	Simulations Eingangsdaten
D11.12	Teilmodell / Overlay	D21.30	Zusätz.kundenspez. Q-Dokumente	D24.15	Produktionsfreigabe	D28.6	Technisches Datenblatt	D31.12	Konzeptberechnung
D11.13	Bestückzeichnung	D21.31	Herstellbarkeitsanalyse	D24.16	Prozessfreigabe neue Technolog	D28.7	8D-Rep.Lief./Korrekturmaßnahme	D31.13	ISO Messdaten
D11.14	Produktzeichnung (Katalogware)	D21.32	Prüfauftrag	D24.17	Softwarefreigabe Intern/Kunde	D28.8	R@R/Audit bei Lieferanten	**D32**	**Logistik**
D12	**Externe CAD-Daten**	D21.33	Lessons Learned	D24.18	Layout Review Checkliste	D28.9	Produktänderungsmitteilung	D32.1	Logistik Kalkulation
D12.1	Geometrie v. Kunden	D21.34	8D-Report an Kunden (Muster)	D24.19	Freigabe elektron. Bauteile	D28.10	Notfallplan	D32.2	Machbarkeit/Risiken Logistik
D12.2	Zeichnung v. Kunden	D21.35	Liste besonderer Merkmale	D24.20	Baseline-Freigabe	D28.11	Kapazitätsinformationen	D32.3	Packmittel-Umlaufm.-Datenblatt
D12.5	Geometrie v. Lieferanten	D21.36	Funktionale Sicherheit	D24.21	Baseline-Fehlerprotokoll	D28.12	Spezielle Vereinbarung Mat.	D32.4	Checkliste Logistikqualität
D12.6	Zeichnung v. Lieferanten	D21.37	Werkstoffdaten	D24.22	ESD-EF-Messprotokoll	**D29**	**Hardware**	D32.5	Checkl Prozesse zu Lieferanten
D12.7	Aufbereitete Geom. v. Kunden	**D22**	**Produkt Konzept**	D24.23	Maschinenfähigkeitsnachweis	D29.1	Blockschaltbild	D32.6	Checkl. int. Logistikabläufe
D12.8	DMU-Geometrie v. Kunden	D22.1	Lastenheft Gerät	D24.24	Scorecard	D29.2	Leiterplattenlayout	D32.7	Logistics-Planning-Level
D20	**Vertrauliche Dokum.**	D22.2	Lastenheft Einzelteil&Software	D24.25	Zeichnungs-Checkliste	D29.3	Systemarchitektur	D32.8	Vorkalkulation Verpackung
D20.1	Angebot zum Kunden	D22.3	Pflichtenheft Gerät	D24.26	Checkliste (Projekt-Serie)	D29.4	Hardwarebeschreibung	D32.9	Druckdaten
D20.2	Kalkulation (P2R)/Vorkalkulat.	D22.4	Pflichtenheft Betriebsmittel					**D40**	**Produktnutzung**
D20.3	Vertrag mit Kunde (LOI etc.)	D22.5	Pflichtenheft Einzelteil&Softw					D40.1	Montageanleitung
D20.4	Bestellung	D22.6	Lastenheft Betriebsmittel					D40.2	Bedienungsanleitung
D20.5	Rahmenauftrag	D22.7	Lastenheft Prüfmittel					**D97**	**Bilder / Fotos**
D20.6	Kostenlebenslauf	D22.8	Pflichtenheft Prüfmittel					D97.1	Verpackungsmittel-Foto
D20.7	SEK-Rechnungen	D22.9	Lastenheft Licht					D97.2	Geräte/Komponenten-Foto
D20.8	SEK-Dokumentation	D22.10	Pflichtenheft Licht					D97.3	Betriebsmittel-Foto
D20.9	Pre-Nominationletter	D22.11	Konzeptbewertung					D97.4	Ersatzteil-Foto
D29.10	Third Party Agreement								

Appendix E

nicht automatisierbare Tests am Beispiel

Weitere Workflow: für Merkmal Prüfung bei Status-Wechseln

Original Testdaten:

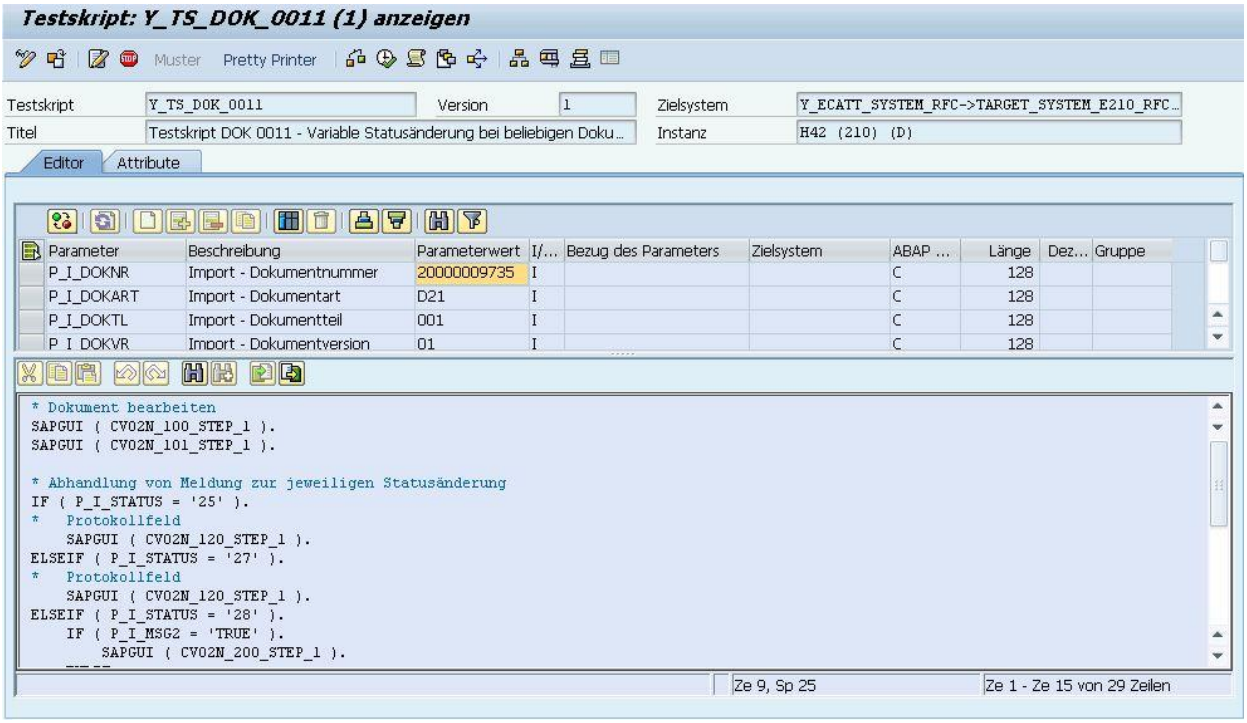
System:	**Objekt:**
Mandant:	**Hardware:**
UserID:	**Betriebssystem:**
Datum:	**CAD System:**
Name:	**DIS des Protokolls:**

Funktion	Test erforderlich	Trans-aktion	Hinweise / Ablauf	Erwartetes Ergebnis [Fehlermeldungen / Hinweismeldungen]	iO / niO
Einchecken - Auschecken					
Einchecken (Status 20 - 21)			DIS unter Dokumentart D21 anlegen und speichern	DIS wird im Status 20 angelegt.	
			Status auf 21 ändern	Nach dem Speichern wird die Datei automatisch in der Datenbank abgelegt (Schloss zu)	
				Die Datei wird am lokalen Ablageort gelöscht	
			Datei in der Anwendung geöffnet lassen	Die Datei kann nicht gelöscht werden und es kommt die Fehlermeldung: Datei ... kann auf dem Frontend nicht gelöscht werden.	
			Datei mit dem Button unterhalb der Originale ablegen	Die Datei wird abgelegt (Schloss zu), der Status aber nicht umgesetzt	
Zwischenstand erzeugen (Status 95)			Status auf 95 ändern und mit Enter bestätigen	Es öffnet sich ein Protokollfeld, welches gefüllt werden muss (z.B. 1. Überarbeitung). Ein Überspringen mit Enter ist nicht möglich.	
			DIS speichern	Nach dem Speichern hat der DIS wieder den Status 21	
			Statusprotokoll anzeigen	Zwischenstand wird angezeigt	
Auschecken (Status 21 - 20)			Original selektieren und mit dem Button "Bleistift" unterhalb der Originale in Arbeit nehmen	Zielverzeichnis wird vorgeschlagen	
				Die Datei wird im Zielverzeichnis abgelegt und automatisch in der Applikation geöffnet.	
				Der DIS wurde automatisch auf Status 20 gesetzt und das Schloss ist offen	
Auschecken mit anderem Sachbearbeiter			Original selektieren und mit dem Button "Bleistift" unterhalb der Originale in Arbeit nehmen	Der DIS wurde automatisch auf Status 20 gesetzt, das Schloss ist offen und der Sachbearbeiter wurde geändert.	
Einchecken (Status 20 - 21)			Datei ändern, speichern und wieder einchecken	Auf dem Reiter "Originale" taucht der Zwischenstand unterhalb der Originaldatei auf (kleines Dreieck öffnet die Struktur) - grüner Punkt: aktuelle Datei - gelbes Dreieck: Zwischenstand	
Automatische Erzeugung des neutralen Format (Status 21 - 25)			Status von 21 auf 25 ändern	Es öffnet sich ein Protokollfeld, welches mit Enter übersprungen werden kann.	
				Nach einiger Zeit (ca. 30 sec.) wird das neutrale Dokument (z.B. PDF) erzeugt und als 2. Original dem DIS hinzugefügt (während der Erzeugung ist der DIS gesperrt).	
				Der Eintrag aus dem Protokollfeld (z.B. 1. Review) wird im Statusprotokoll	

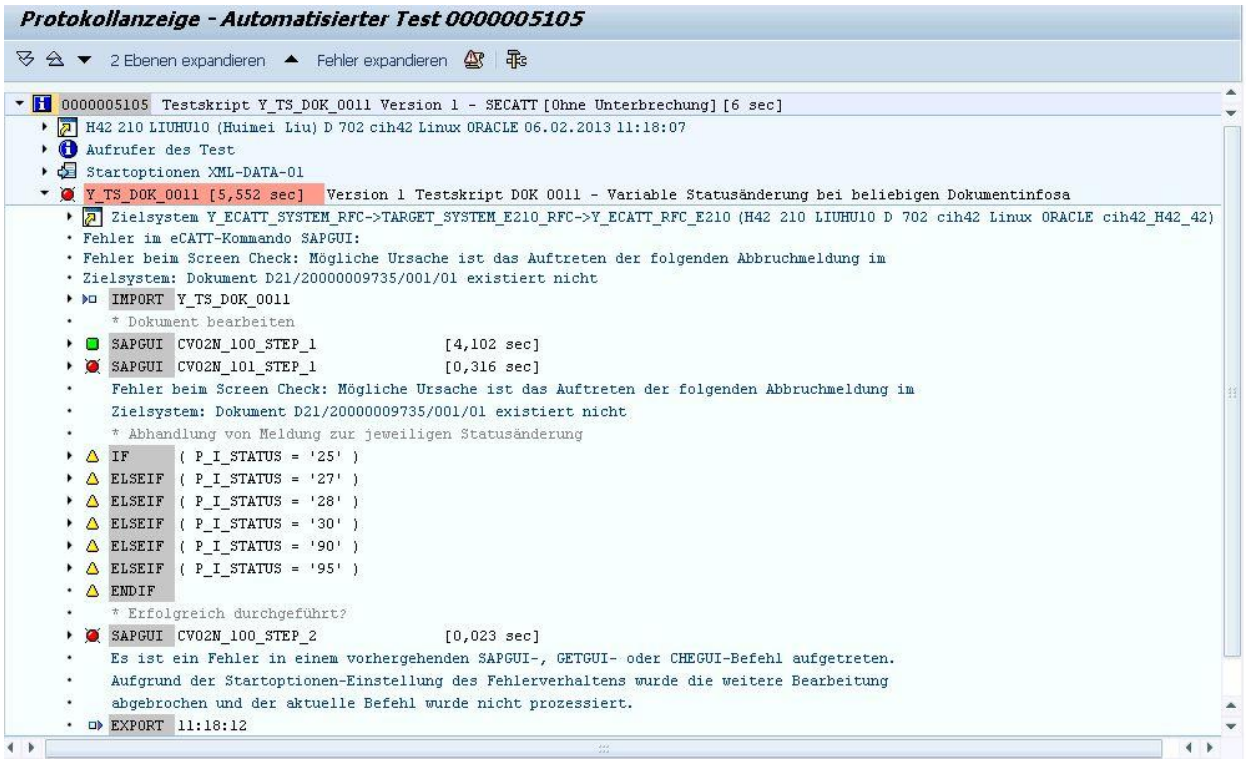
Bei dem Workflow ist der Testtyp abhängig. Es konnte nicht automatisch durchgeführt werden, jedoch ist es auch durch die Parametrisierungsmethode effizient.

TestID	Bereich	Funktion	Testtyp	Testpriorität	H42	H45	Fehlermeldung/ Bemerkung	Protokoll Nr	Geprüft am	Geprüft ...
DIS_Merkmal_001	DOK	Testskript Y_TK_CN01_MERKMALANZEIGEN / Dokument anlegen bei Dokumentart D10 D10.1 entsprechende Merkmale anzeigen	automatisch	1 - hoch (muss)	iO	iO				
DIS_Merkmal_002	DOK	**Testdaten mit Konfiguration /** Workflow bzgl. Dokument anlegen bei beliebigen Dokumentart bzw.Dokumentinhalt, Dokumentinhalt-abhängige Anzeigen von Merkmale / **enth. Testskripte** Y_TK_CN01_MERKMALANZEIGEN, DXXDXX.ALL	automatisch	1 - hoch (muss)	iO	iO		0000005232	27.02.2013	liuhu10
DIS_Merkmal_003	DOK	Testskript Y_TS_CV01_MERKMALANZEIGEN (2)/ Wertlisten Anzeigen von Merkmal "CAD-Baugruppe" und "Basisvariante", D10.1	abhängig	1 - hoch (muss)	iO	iO		00000052781	28.02.2013	liuhu10
DIS_Merkmal_004	DOK	Testskript Y_TS_CV01_ML2/ Wertlisten Anzeigen von Merkmal "Simlationstyp", D10.6, D10.9, D31.11	manuell	1 - hoch (muss)	iO	iO	TS-Fehlermeldung wegen Page-Change --> deshalb diese Fall ist nicht automatisierbar	0000005274	28.02.2013	liuhu10
DIS_Merkmal_005	DOK	Testskript Y_TS_CV01_ML4/ Wertlisten Anzeigen von Merkmal "Maßzeichnung" und "Maßstab (nur CATIA)", D11.1	manuell	1 - hoch (muss)	iO	iO	auch wegen Page-Change Fehler während der Durchführung des Testskriptes	0000005288	28.02.2013	liuhu10
DIS_Merkmal_006	DOK	Testskript Y_TS_CV01_ML5/Wertlisten Anzeigen von Merkmal "Berichttyp" und "Gegenstand", D21.7, D21.25	abhängig	1 - hoch (muss)	iO	iO	Schreibfehler vonTestfall in Excel: Anbauanlage (Anbaulage)	0000005290	28.02.2013	liuhu10
DIS_MerkMal_007	DOK	Testskript Y_TS_CV01_ML6/Wertlisten Anzeigen von Merkmal, MaterialStatus entspicht PEP-Phase, D21.2	automatisch	1 - hoch (muss)	iO	iO	betrachte version 2	0000005299	28.02.2013	liuhu10
DIS_Merkmal_008	DOK	Testskript Y_TS_CV01_ML8/Wertlisten Anzeigen von Merkmal "Langzeitversorgungskonzept", D22.3	abhängig		iO	iO		0000005300	28.02.2013	liuhu10

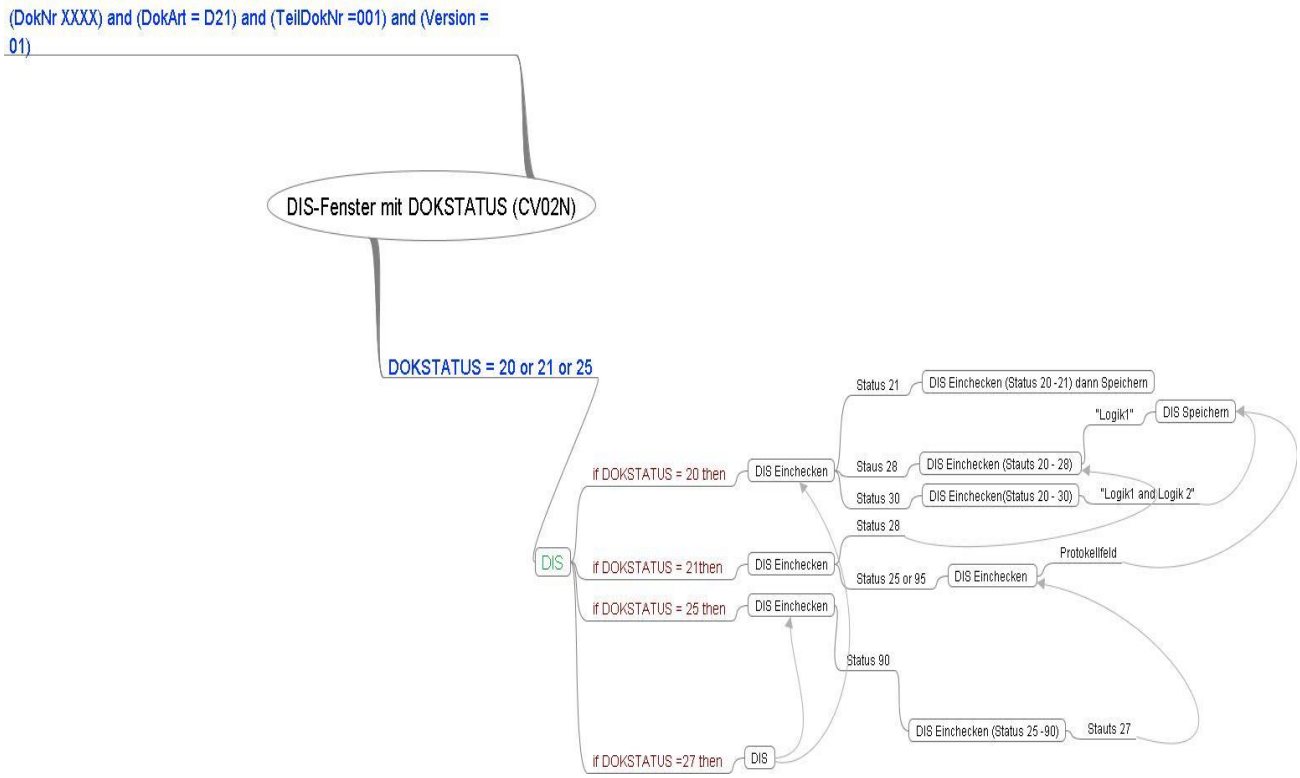
Testskript: Y_TS_DOK_0011(1)



Nicht durchführbar → nicht automatisierbar



Automatison-Modell



Appendix F

Reguläre Sprache und Testautomat*

Reguläre Ausdrücke Definition:

Reguläre Sprachen [Leip] lassen sich über reguläre Ausdrücke definieren. Sie wird als die Sprache bezeichnet, die durch einen Ausdruck E repräsentiert wird, mit L(E).

Def.: Die Menge der regulären Ausdrücke (über dem Alphabet Σ) ist die kleinste Menge, die

die folgenden Bedingungen erfüllt:

Ausdruck

1. ε und ∅ sind reguläre Ausdrücke.

Dazu die Sprache : L(ε) = {ε} und L(∅) = ∅.

2. a ∈ Σ ist regulärer Ausdruck.

Dazu die Sprache : L(a) = {a}

Wenn E und F reguläre Ausdrücke sind, dann

3. (E+F) und (EF) sind reg. Ausdrücke.

Dazu die Sprache : L((E+F)) = L(E) ∪ L(F)

L((EF)) = L(E)L(F)

4. E* ist regulärer Ausdruck.

Dazu die Sprache : L(E*) = (L(E))*

Bindungsstärken: * > Konkatenation > +

Beispiele über Alphabet {0,1}:

(01)*, ((01)* + (00)*), ((11)1)

äußere Klammern können entfallen, ebenso solche, die wegen Assoziativität von + und

Konkatenation nicht notwendig sind:

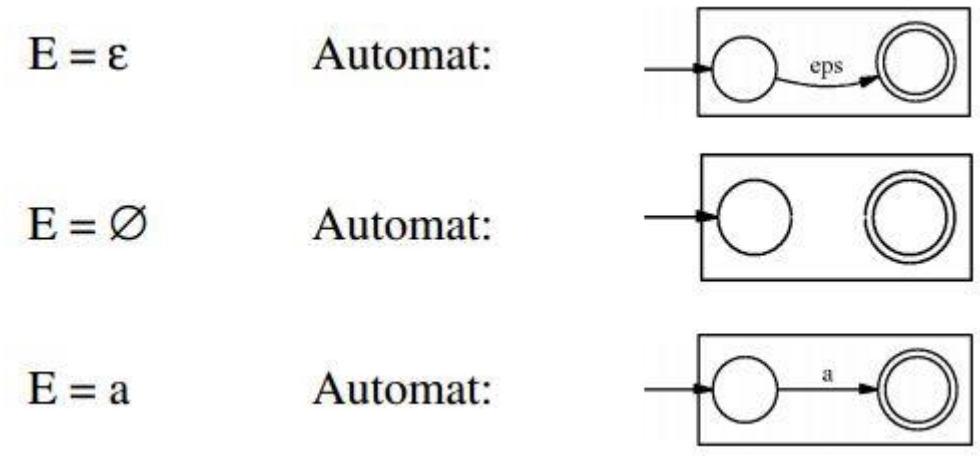
(E1+E2)+E3 = E1+(E2+E3) deshalb einfach E1+E2+E3

(E1E2)E3 = E1(E2E3) deshalb einfach E1E2E3

aber: (01)* ≠ 01* * bindet stärker

(01)+0 ≠ 0(1+0) {01,0} versus {01,00} Wichtig: ∅ S=S∅ =∅

Reguläre Sprachen und endliche Automaten



Satz: Sei L eine reguläre Sprache. Es gibt einen endlichen Automaten, der L akzeptiert.

Beispiel: Ausführliche Vorstellung der Workflow/Testszenario (Siehe Abbildung 30 Testautomation-Modell)

Sie A ein Automat, gegeben durch Testskript T = {Testskript_1, Testskript_2, Testskript_3,…}, Startzustand Testskript_1, Finalzustände F = {Testskript_1, Testskript_2, Testskript_3}

Die Testworkflow (Sprache) wird daraus konstruiert, die Ergebnisse von einem Zustand T_i in einen Zustand T_j bringen. Diese Workflow durch reguläre Ausdrücke zu charakterisieren, folgt, dass die Workflow des Automaten regulär ist. Die Workflow des Automaten ist dann nämlich die Vereinigung der Workflows, die vom Startzustand in die verschiedenen Finalzustände bringen. Jeden Durchlauf bis Finalzustand bildet ein Testszenario ab. Eventuell existieren es verschiedene Pfade von T_i nach T_j und die nicht alle auf einmal betrachten müssen, schränken die möglichen Wege durch den Automaten zunächst ein und erweitern sie dann, bis keine Einschränkungen mehr vorliegen oder neue Entwicklungsschritt (in zusätzliche Quadrat) vorkommen, die Einschränkungen (der unterschiedliche Testszenario) können durch Spezifikationstabelle berücksichtigt werden.

Beim Lesen eines Leeren Workflow bleibt der Automat im selben Zustand. Der Testautomat dient zu einer übersichtlich Darstellung der automatisierbaren Testprozesse in unterschiedlichen Workflows, damit die wiederholten Testskripts aktiv aufgerufen werden.

Literaturverzeichnis

[Carl13] Carl Hanser Verlag GmbH & Co. KG :“ *Kundenanforderung Kano Modell*“, http://www.qz-online.de/qualitaets-management/qm-basics/artikel/kundenanforderungen-kano-modell-168360.html, 2013

[Gada10] Yvonne Gadaschewski, Nicolas Druzba:“ *Methodische Grundlagen der Testautomatisierung*", http://winfwiki.wi fom.de/index.php/Methodische_Grundlagen_der_Testautomatisierung#Allgemein, FOM Essen, 2010

[Gaer12] Alexander Gärtner:“ *Evaluierung der Testautomatisierung mit SAP Solution Manager 7.1*“, Bachelorarbeit, http://www.diplomarbeiten24.de/vorschau/192069.html, Hochschule Karlsruhe - Technik und Wirtschaft (Informatik und Wirtschaftsinformatik), 2012

[Glin02] Martin Glinz:“ *Software Engineering I*”, Vorlesungsskript, WS 2001/2002”, https://homepages.fhv.at/hv/Semester4/OOAD/glinz.pdf, 2002

[Goel06] Olaf Göllner:“ *Testen / Capture & Replay*”, http://www.se.uni-hannover.de/pub/File/kurz-und-gut/ss2006-seminar-werkSoft/Kurz_Gut-Olaf_Goellner_Testen.pdf, Uni Hannover, 2006

[Guel10] B. Güldali:“ *IT_Arbeitskreis_-_Testautomatisierung“,* http://is.uni-paderborn.de/fileadmin/Informatik/AG-Engels/Personen/Baris_Gueldali/Vortraege/IT_Arbeitskreis_-_Testautomatisierung.pdf, Paderborn, 2010

[Hell10a] HELLA KGaA Hueck & Co.:” *Was ist PDM*“, http://www.HELLA.com/produktion/Intranet/WebSite/Intranet_de/ZentraleBereiche/InformationsManagement/SAP/Module/Produktdatenmgmt/Was_ist_PDM/Was_ist_PDM.jsp, 2010

[Hell10b] HELLA-Praxishandbuch:“ *eCATT – Softwaretestautomatisierung im SAP Bereich*“, Dokumentation, 2010

[Hell12a] HELLA KGaA Hueck & CO.:“ HELLA-interne-Zeitschrift“, *Daten aus 2012 HELLA-interne Dokument*, Lippstadt, 2012

[Hell12b] HELLA KGaA Hueck & CO.:“ *PLM-System”,* http://www.HELLA.com/intranet-plm/30.html?rdeLocaleAttr=de, 2012

[HLT06] Helfen, Markus, Lauer Michael und Trauthwein, Hans Martin**:“** *SAP-Lösungen testen (SAP PRESS)*“, ISBN: 3-89842-721-8, 1.Auflage, Bonn, 2006

[Hoop03] James Hoopes:“ *False Prophets: The Gurus Who Created Modern Management and Why Their Ideas Are Bad for Business Today*, Top-down und Bottom-up in der Management-Theorie“, ISBN 9780738207988, 2003

[Inno13] Firma Innobis:“ *Solution Manager für Testmanagement - Versteckt aber vorhanden*“, http://www.e3cms.de/index.php?id=3270, 2013

[Jose13] SICON Joself Schöttner – Industire - Consultant, http://www.pdm-infoshop.de/pdmplm/, 2013

[Koch03] Ramona Koch:“ *Einführung ins Testen*“, Ausarbeitung im Rahmen der Veranstaltung „*Software Engineering*“, 2003

[Kuhr13] Marco Kuhrmann:“ *Wasserfallmodell, Enzyklopädie der Wirtschaftsinfomatik online-Lexikon*“, http://www.enzyklopaedie-der-wirtschaftsinformatik.de /lexikon/is-management/Systementwicklung/Vorgehensmodell/Wasserfallmodell/, 2013

[Lang10] Carsten Langenhan:“ *Automatisierte Software-Tests in SAP-Umfeld“, SO_MAGAZIN*, http://www.iqnite-conferences.com/iqnite-de/Presse/sq-magazin-2010-03.pdf, Ausgabe 14.März 2010

[Leip] Universität Leipzig:“ *Reguläre Sprachen.*“, Informatik Papers, http://www.informatik.uni-leipzig.de/~brewka/papers/3.RegulaereSprachen.pdf, Leipzig

[Meie12] Alexander Meier:“ *Testautomation im SAP PLM Kontext*“, Testspezifikation, Bachelorarbeit, FH Dortmund, 2012

[Meis12] Jens Meißner:“ *Evolution im Softwaretesting*“, Rockwell Automation, Vortrag, Karlsruhe, 2012

[Micr10] Microsoft Corporation:“ *Günstiger und Schneller Testen Durch Komfortable Testautomatisierung*“, http://download.microsoft.com/download/4/5/1/4519B5E0-7E1F-4D70-A015-A60225B46E0F/GuenstigerUndSchnellerTestenDurchKomfortableTestautomatisierung.pdf, Microsoft Visual Studio 2010

[Maye13] Robert Mayerhofer:“*Praxis Handbuch SAP Business One*“, ISBN 978-3-8362-1839-9, 2013

[Naum12] Jacqueline Naumann:“ *Praxisbuch eCATT*“, ISBN 978-3-8362-1151-2. Galileo Press Bonn, 2012

[PiMu03] Piesche, M. und Muhr, W.:“ *Testautomatisierung am Beispiel von “Rational TeamTest“”,* TüvIT Sommerforum, www.tuvit.de, September 2003

[PKS02] Pol, M., Koomen, T und Spillner, A.:“ *Management und Optimierung des Testprozesses: Ein praktischer Leitfaden für erfolgreiches Testen von Software mit TPI und TMap“,* 2.Deutsche Auflage, ISBN: 3-89864-156-2, 2002

[SAP13a] SAP-Help:” *SAP – Bibliothek - eCATT: Extended Computer Aided Test Tool*”, Übersicht der Prozessverlauf“, http://help.sap.com/saphelp_erp60_sp/helpdata/de/6f/b1923fa5e93c17e10000000a114084/content.htm, 2013

[SAP13b] SAP-Help:" *eCATT: Extended Computer Aided Test Tool (BC-TWB-TST-ECA)*", http://help.sap.com/saphelp_nw04/helpdata/de/1b/e81c3b84e65e7be10000000a11402f/content.htm, 2013

[SAP13c] SAP:" *The Best-Run Businesses Run SAP, SAP Netweaver, Komponenten und Werkzeug, SAP Solution Manager*", http://www.sap.com/germany/plattform/netweaver/components/solutionmanager/index.epx, 2013

[SAP13d] SAP:" *Test Automation with SAP Solution Manager and HP QTP*", http://www.sap.com/australia/campaign/ES-Newsletter Q4/images/test_automation.gif, 2013

[SAP13e] SAP Dokumentation:" *Testautomatisierung mit eCATT*", https://help.sap.com/saphelp_aif20/helpdata/de/88/d667d8546d443ebe0f328240830669/content.htm, 2013

[SAP12] SAP Standard Press, 2012

[SBMB11] Seidl, Richard, Baumgartner, Manfred und Bucsics, Thomas:" *Basiswissen Testautomatisierung: Konzepte, Methoden und Techniken*." Dpunkt Verlag, 2011

[Schm] Alexander Schmitz:" *Automatisierung des Testens objektorientierter Frameworks",* Diplomarbeit, Universität Dortmund Software Technologie, http://ebus.informatik.uni-leipzig.de/www/media/lehre/diplomarbeiten/da-schmitz-pdf.pdf

[SHSM13] SAP-Help. solution management:" *Der Test Organizer*", http://help.sap.com/saphelp_sm32/helpdata/de/a2/157235d0fa8742e10000009b38f889/frameset.htm, 2013

[SpLi09] Andreas Spillner, Tilo Linz:" *Basiswissen Softwaretest Aus- und Weiterbildung zum Certified Tester Foundation Level nach ISTQB-Standard*", ISBN 3-89864-358-1, 2009

[StEs13] Marco Stark, David Espin:" *Modellbasiertes Test*", Fundamentaler_Testprozess_ISTQB.png, Hochschule FOM Essen, 2013

[Wien94] Lauren Ruth Wiener:" *Digitales Verhängnis - Gefahren der Abhängigkeit von Computern und Programmen"*, ISBN: 3-89319-672-2, Bonn, 1994

[Wiki12b] Wikipedia, Softwarequalität:" *Die Qualitätskriterien für Software nach ISO 9126*", http://de.wikipedia.org/wiki/Softwarequalit%C3%A4t, 2013

[Zamb98] Zambelich, K.:" *Totally Data - Driven Automated Testing – A White Paper*.", http://sqa-test.com, 1998

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift:

Stuttgart, den 28.05.2013