Felix Fehrer, BSc

ENTWICKLUNGS- UND TESTDATENANALYSE MECHATRONISCHER KOMPONENTEN IM AUTOMOBILBAU

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Maschinenbau

eingereicht am

Institut für Fahrzeugtechnik

an der

Technischen Universität Graz

Beurteiler:

Dipl.-Ing. Dr. techn. Jürgen Fabian

Betreuer:

Dipl.-Ing. Markus Ernst, MLBT

Graz, März 2016

Danksagung

Die Masterarbeit stellte sich als die interessanteste Zeit in meinem ganzen Studium heraus, da ich während diesem Zeitraum sehr viel für mein bevorstehendes Berufsleben lernen konnte. Daher möchte ich mich an dieser Stelle bei all den Personen bedanken, die mich bei dieser Arbeit direkt und indirekt unterstützt haben.

Besonders bedanken möchte ich mich bei meinem Betreuer Herrn Ernst, der mich in jeder Projektphase unterstützt hat und mit ihm in den zahlreichen Diskussionsrunden produktive Lösungen entstanden sind. Für die Begutachtung der Arbeit und für organisatorische Betreuung möchte ich mich recht herzlich bei Herrn Fabian bedanken. Herr Fabian stand mir jederzeit für allfällige Fragen offen gegenüber und nahm sich trotz engem Terminplan immer für mich Zeit. Nicht zuletzt möchte ich mich bei Herrn Wolf bedanken, der mich bei meiner Arbeit softwaretechnisch unterstützt hat und während der Zusammenarbeit zu einem guten Freund geworden ist. Ohne ihn wäre die Arbeit nicht in der vorgegebenen Zeit umsetzbar gewesen.

Für die Betreuung seitens des Industriepartners MAGNA Powertrain standen mir Frau Mandic und Herr Pfeffer für Fragen stets zur Seite und dafür möchte mich ebenfalls recht herzlich bedanken.

Aus meinem familiären Umfeld möchte ich mich bei meinem Vater für die großzügige monetäre Zuwendung während des Studiums bedanken. Dadurch konnte ich mich voll und ganz auf mein Studium konzentrieren und dieses in respektabler Zeit abschließen. Letzten Dank möchte ich meiner Freundin Angelika widmen. Die Distanz von Studienort zu Heimatort stellte unsere Beziehung auf eine harte Probe, die wir nun nach vier Jahren erfolgreich gemeistert haben.

EIDESSTATTLICHE ERKLÄRUNG

AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und, die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Graz, am 29.03.2016 _____

Felix Fehrer

Kurzfassung

Durch die in den letzten Jahren steigende Implementierung mechatronischer Komponenten im Automobilbau gewinnt die automotive Software zusehends an Bedeutung. Bei der Erstellung von automotiver Software werden viele Entwicklungs- und Testdaten angehäuft, die im Laufe der Produktentstehung untersucht werden müssen. Aufgrund der Vielzahl an Daten treten in der Entwicklungs- oder Testphase oftmals Fehler auf, die es zu analysieren und beheben gilt. Die zahlreichen Analysetools am Markt liefern nur begrenzt aussagekräftige Ergebnisse bei der Untersuchung des Entwicklungsprozesses, da sehr wenig auf die in der Automobilindustrie verwendete Datenstruktur des V-Modells eingegangen wird. Unkonventionelle Analysetechniken, wie sie seitens des Industriepartners gefordert werden, können nur mithilfe eines maßgeschneiderten Analysetools ermöglicht werden. Dazu wird die Vorgehensweise des Industriepartners bei der Entwicklung eines neuen Produkts untersucht und alle zugehörigen Entwicklungs- und Testdaten erfasst. Zur Quantifizierung dieser Daten werden Kennzahlen, sogenannte Key Performance Indikatoren, mit dem Industriepartner vereinbart. Diese Kennzahlen sollen konkrete Fragestellungen im Entwicklungsprozess beantworten und bei jeder Projektphase eventuelle Soll-/Ist-Abweichungen aufzeigen. Die Analyse bei einer retrospektiven Betrachtungsweise verschiedener Zeiträume des Entwicklungsprozesses in Anlehnung an das V-Modell soll eine höhere Produktivität und einen effizienteren Projektablauf bewirken. Mit einem maßgeschneiderten Analysetool werden diese Kennzahlen in Diagrammen und Tabellen visualisiert, um die vereinbarten Fragestellungen an den Entwicklungsprozess beantworten zu können. Eine abschließende Untersuchung von Beispielprojekten unter Zuhilfenahme des Analysetools soll die praktische Handhabung der KPIs verdeutlichen.

Abstract

Due to the increasing implementation of mechatronic parts in automotive engineering in the last years, automotive software becomes more important. Programming automotive Software accumulates huge amounts of development- and test data, which have to be investigated during product development. Faults occur during the testing of automotive Software and these have to be analysed and resolved. There are numerous analysis tools on the market but they don't deliver desired results during the analysis of the process development. The tools don't consider data structures of the V-model which is used in the automotive industry. Unconventional techniques of data analysis, which are needed by the industrial partner, only can be done with a tailored analysis tool. Therefore, the development- and test data is gathered and the product developing process by the industry partner is investigated. For the quantification of this data so called Key Performance Indicators are used. With these metrics the development process related questions could be answered and deviations from the goal of every project phase should be identified. Higher productivity as well as a more structured project plan can be reached by using a retrospective analysis of various time periods in the development process. With this analysis tool, key performance indicators are illustrated in charts and tables to answer difficult questions during the development process. Finally, an investigation of elected projects with the support of the KPIs, used in the analysis tool, shows the benefit of involving the tool in projects.

Inhaltsverzeichnis

Abkürzungen

ALM	Application Lifecycle Management
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
ASW	Application Software
BSW	Basissoftware
CAN	Controller Area Network
CI	Change Issue
CLR	Common Language Runtime
CSV	Comma Separated Values
E/E Architecture	Electrical/Electronical Architecture
GB	Gigabyte
GUI	Graphical User Interface
HiL	Hardware in the Loop
KPI	Key Performance Indicator
LIN	Local Interconnect Network
MiL	Model in the Loop
MOST	Media Orientated Systems Transport
MR	Module Requirement
MSIL	Microsoft Intermediate Language
RC	Release Candidate
RTE	Runtime Environment
SCR	System Component Requirement
SiL	Software in the Loop
SQL	Structured Query Language
SR	System Release
SW	Software
SWC	Software Components
TC	Test Case
XML	Extended Markup Language

1 Einleitung

Der stetige Anstieg von Anforderungen seitens des Gesetzgebers oder der Kunden bezüglich Sicherheit, Komfort oder Effizienz im Automobil erfordert einen immer höheren Implementierungsgrad an mechatronischen Systemen, wie Elektrik und Elektronik (E/E), Mechanik und Software, [1]. Gerade die eingebettete Software in Zusammenspiel mit der zugehörigen Elektronik ist Hauptträger moderner Innovationen im Automobilsektor. Automotive Software ist mittlerweile der dominierende Faktor in der Automobilentwicklung geworden und steigt exponentiell seit den letzten 30 Jahren, wobei dieser Trend voraussichtlich noch weiter anhalten wird, [2]. Das Automobil ist mittlerweile eines der komplexesten Konsumgüter in der Gesellschaft geworden. Um den gestiegenen Ansprüchen nachzukommen, bedarf es einheitlicher Richtlinien des Entwicklungsprozesses. Standards wie AUTOSAR [3] gewinnen aufgrund steigender Implementierung von Softwarefunktionen im Kraftfahrzeug in den letzten Jahren immer mehr an Bedeutung und haben erheblichen Einfluss auf Methoden, Prozesse und Werkzeuge. Hohe Erwartungen an Zuverlässigkeit in rauen Umgebungsbedingungen, Verfügbarkeit und lange Produktlebenszyklen sind weitere Randbedingungen an die elektronischen Systeme im Automobilbau. Getrieben durch immer kürzer werdende Entwicklungszeiten sind geeignete Analysemethoden im Entwicklungsprozess unerlässlich, [1]. Ein organisierter Entwicklungsprozess ist jedoch noch keine Garantie für eine zufriedenstellende Produktentwicklung. Das Zusammenspiel von gut gestaltetem Entwicklungsprozess, richtig ausgewählten Techniken für die Tätigkeiten im Prozess (Sprachen, Methoden, Werkzeuge), sowie geschulte Mitarbeiter, die den Entwicklungsprozess verstehen und ihn mit den Techniken umsetzen können, sind Indizien für hohe Software-Qualität, [4]. Um einen Entwicklungsprozess beurteilen und verbessern zu können, muss dieser systematisch analysiert werden. Jene Menge an Entwicklungs- und Testdaten, welche im Zuge der Produktentwicklung entstehen, können zu einer Evaluierung herangezogen werden. Die Analyse dieser Daten ermöglicht potentielle Missstände in der Prozessplanung und -durchführung aufzuzeigen, diese Mängel zu beheben und bei zukünftigen Projekten zu vermeiden.

1.1 Zielsetzung

Ziel dieser Arbeit ist es, den Entwicklungsprozess mit der Analyse von Entwicklungs- und Testdaten übersichtlicher und transparenter darzustellen. Da mit steigender Komplexität und Quantität von Softwarefunktionen auch die Fehleranfälligkeit zunimmt sollen mit geeigneten Maßzahlen Abweichungen vom Entwicklungsprozess und den dazugehörigen Entwicklungs- und Testdaten untersucht werden. Dazu wird das vom Industriepartner verwendete Application Lifecycle Plattform PTC MKS [5] als Ausgangssituation herangezogen. In dieser Plattform sind die nötigen Informationen enthalten, um eine Analyse des Entwicklungsprozesses durchführen zu können. Mit dem maßgeschneiderten Analysetool soll die gesamte Projektabwicklung effizienter gestaltet werden und somit Entwicklungszeit und Kosten gesenkt werden. Die vorliegende Masterarbeit stellt Ausschnitte aus der Mitentwicklung eines maßgeschneiderten Analysetools dar, vgl. [6]. Mit übersichtlichen und nachvollziehbaren Grafiken und Tabellen sollen folgende Ziele verwirklicht werden:

- Der aktuelle Projektstatus soll anhand von Arbeitspaketen erörtert werden.
- Der Trend der Planungselemente hinsichtlich deren Erfüllungsgrads soll einsehbar sein, um Abweichungen vom Ziel leicht erkennbar zu machen.

- Ein Soll-/Ist-Vergleich von abgeschlossenen zu noch zu bearbeitenden Arbeitspaketen, Anforderungen und Testfälle soll durchführbar sein.

- Der Bearbeitungsstatus aller am Entwicklungsprozess beteiligten Elemente sollen über die Projektdauer offengelegt werden.

- Die Qualität der Vernetzung der Planungselemente untereinander soll dargestellt werden.

- Projekte sollen anhand der daran beteiligten Arbeitspakete, Anforderungen und Testfälle miteinander verglichen werden.

Alle diese Punkte sollen miteinander in Interaktion stehen, um somit Aussagen über den ganzen Entwicklungsprozess verschiedener Projekte treffen zu können.

1.2 Gliederung der Masterarbeit

Die Arbeit gliedert sich in einen Theorieteil, welcher all jene Punkte zusammenfasst, die für die Durchführung der Arbeit benötigt werden und für die Funktionsweise des Tools wichtig sind. Im praktischen Abschnitt wird auf das Analysetool mit dessen Key Performance Indikatoren eingegangen.

In Kapitel 2 werden die theoretischen Grundlagen erläutert. Es wird auf die Anforderungen, die Funktionsweise und den Aufbau von automotiver Software eingegangen. Ein Grundverständnis des Entwicklungsprozesses des Industriepartners wird mit Erläuterung des V-Modells geschaffen. Die erforderlichen Grundlagen zur Erstellung des Analysetools werden in den Kapiteln 2.4 und 2.5 beschrieben. Den abschließenden Teil der theoretischen Erörterungen bildet die Untersuchung der am Markt verfügbaren Analysetools mit dem Kapitel 2.6.

Im praktischen Teil der Arbeit wird der Entwicklungsprozess des Industriepartners mit dessen Datenstruktur im Application Lifecycle Management Tool PTC MKS untersucht, um die Projektabhandlung der Firma MAGNA Powertrain GmbH & Co. KG zu verstehen. Nach Erfassen der Datenstruktur wird der softwaretechnische Aufbau des Analysetools erklärt und die mit dem Industriepartner erarbeiteten Key Performance Indikatoren beschrieben.

Abschließend erfolgt eine Analyse der Daten in Form von Fallbeispielen aus verschiedenen Projekten. Damit sollen mögliche Vorgehensweisen beim Gebrauch des Analysetools aufgezeigt werden. Die in der Masterarbeit verwendeten Beispielprojekte enthalten Entwicklungs- und Testdatensätze, um die Funktionalität des KPI Tools vollständig zu testen.

2 Theoretische Grundlagen automotiver Software

Die nachfolgenden Kapitel enthalten die benötigten theoretischen Grundlagen, die zur Analyse von Entwicklungs- und Testdaten mechatronischer Komponenten erforderlich sind.

2.1 Einleitung und Stand der Technik

Die Umsetzung von aktuellen Abgasvorschriften oder die Einhaltung von Sicherheitsstandards im Kraftfahrzeug sind mittlerweile ohne softwaretechnische Unterstützung nicht mehr möglich. Ein modernes Kraftfahrzeug besitzt inzwischen 80 ECUs (Electronic control units) und bis zu 100 Millionen Zeilen an Code (LOC). Diese ECUs sind Teil der ganzen Softwareinfrastruktur und kommunizieren mit Bussystemen wie dem Controller Area Network (CAN), MOST (media orientated systems transport), FlexRay oder LIN (local interconnect network) miteinander. Diese Systeme verhalten sich hinsichtlich Übertragungsrate im Vergleich zum Ethernet, welches im PC vorrangig genutzt wird, langsam haben aber höchste Ansprüche bezogen auf Echtzeitfähigkeit und Ausfallswahrscheinlichkeit. Die Netzwerke kommen je nach Anforderungsbereich im KFZ im Antriebsstrang, Chassis, Exterieur/Interieur oder Infotainment zum Einsatz. In den nachfolgenden Absätzen wird auf die Eigenschaften automotiver Software eingegangen und in puncto Anforderungen, Funktionsweise oder Aufbau beschrieben, [7].

2.1.1 Anforderungen an die Software im KFZ

Die wichtigsten Unterschiede von automotiver Software zu herkömmlicher Software, wie beispielsweise PC-Software, werden in folgenden Punkten zusammengefasst, [7]:

- *Zuverlässigkeit:* Automotive Software muss während des ganzen Lebenszyklus im KFZ ausfallssicher sein.
- *Funktionale Sicherheit:* Sicherheitskritische Systeme wie das Anti-Blockier System oder ESC (Elektronisches Stabilitätsprogramm) müssen unter allen Umständen ausfallssicher sein und über Notlaufeigenschaften verfügen.
- *Echtzeitfähigkeit:* Die Einhaltung vorgegebener maximaler Verzögerungszeiten im Micro- oder Millisekunden-Bereich erfordert optimierte Betriebssysteme und spezielles Softwaredesign.
- *Minimaler Ressourcenverbrauch:* Zusätzlicher Speicherbedarf und Rechenleistung erhöhen die Kosten drastisch, da Hardware im Automobilbau millionenfach produziert wird.
- *Robuster Aufbau:* Automotive Software stellt hohe Anforderungen an die elektromagnetische Verträglichkeit.
- *Mechatronische Closed-Loop Regelung:* Reaktionen des Automobils auf Eingriffe durch die Software müssen gemessen und einer Regelung unterzogen werden.

Wie man aus den oben erwähnten Punkten entnehmen kann, gibt es eine große Bandbreite an Anforderungen an die Software im Kraftfahrzeug. Die meisten Systeme müssen „echtzeitfähig“ arbeiten, darunter versteht man, dass die Reaktion der Regelung mit dem physikalischen Prozess Schritt halten muss. Die Software im Automobil wird für den bestimmten Anwendungsbereich erstellt und in das Gesamtsystem eingebunden (Embedded Software). Die Funktionen werden auf der ganzen Welt weiterentwickelt und verbessert, daher haben die Ersatzteile auch einen Lebenszyklus von bis zu 30 Jahren. Je nach Einsatzgebiet unterschieden

sich die typischen Merkmale der Software. So ist die Software für den Antriebsstrang sehr umfangreich, während bei den Fahrwerksanwendungen das Echtzeitverhalten eine zentrale Rolle einnimmt. Bei dem Sicherheits- und Komfortbereich hingegen steht der Ressourcenverbrauch im Mittelpunkt. Die Software ist auf Grund von Applikationsparameter und Kennfelder immer an das Zielsystem anpassbar. Meistens wird die Software für das Kraftfahrzeug in einem Verbund entwickelt, welcher sich aus interdisziplinäre Zusammenarbeit und verteilte Entwicklung zusammensetzt. Ein Beispiel für die interdisziplinäre Zusammenarbeit ist die Verbindung zwischen der Antriebs- und der Elektronikentwicklung. Die verteilte Entwicklung findet zwischen Zulieferer und Fahrzeughersteller statt, [8].

2.1.2 Funktionsweise der Software im KFZ

Die meisten Regelungen im Kraftfahrzeug müssen echtzeitfähig sein, daher müssen sie innerhalb einer vorgegebenen Zeitspanne auf Anforderungen reagieren. Damit die Echtzeitfähigkeit gewährleistet ist, werden bestimmte Programmteile im Steuergerät bevorzugt, [8].

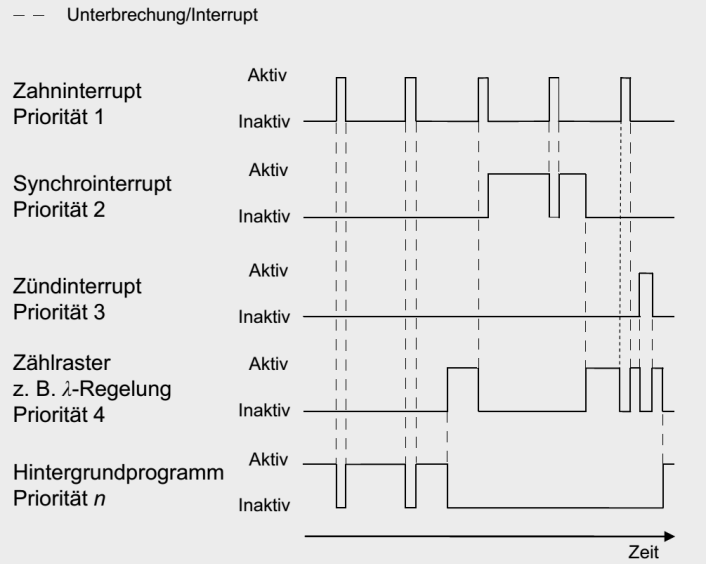


Abbildung 2.1: Prinzip der Verteilung der Rechenleistung durch Interrupt-Steuerung, [8]

Funktionsprinzip der Interrupt-Steuerung: Der Mikrocontroller im Steuergerät bekommt aus dem Programmspeicher viele Befehlscodes, welche der Controller nacheinander ausführt. Die Dauer für die Durchführung eines Befehls hängt sowohl vom eingesetzten Mikrocontroller als auch von der Taktfrequenz ab. Damit Funktionen mit hoher Priorität zuerst abgearbeitet werden, benötigt man eine Softwarestruktur, [8].

Ablauf eines Interrupts: Falls ein Ereignis von hoher Dringlichkeit eintritt, kann das laufende Programm mit der Interrupt Steuerung des Mikrocontrollers unterbrochen werden. Denn das Programm kommt in die Interrupt Routine und arbeitet diese ab. Sobald die Routine beendet wurde, fährt das unterbrochene Programm wieder fort, Abbildung 2.1. Unter einem Interrupt versteht man eine Unterbrechung, welche durch ein Signal von außen ausgelöst wird, [8].

2.1.3 Aufbau von Software im KFZ

Die Software besteht aus vielen Programmteilen. Dabei unterschiedet man zwischen den „wahrnehmbaren Funktionen“ der Software, der Anwendungssoftware und einer Basissoftware. Das Zusammenspiel dieser Softwaren wird in der Softwarearchitektur bestimmt.

Es gibt

- die *statistische Sicht,* welche hierarchisch die Funktionsgruppen, Signale und die Verteilung der Ressourcen beschreibt,
- die *funktionale Sicht,* die den Signalverlauf durch die verschiedenen Funktionen festlegt
- und die *dynamische (=zeitabhängige) Sicht,* welche das Zeitverhalten bei der Abarbeitung der verschiedenen Programmteile betrachtet, [8].

Die Software im Kraftfahrzeug wird in verschiedenen Gruppen aufgeteilt. Die großen Gruppen (Fahrdynamik-, Getriebe- oder Motorsteuerung) lassen sich weiter unterteilen (Luftsystem, Kraftstoffsystem, Drehzahlregelung) bis hin zu einzelnen Sensorauswertungen oder Einspritzmengenberechnungen. Eine komplette Neuentwicklung einer großen Funktion findet sehr selten statt, denn Ziel ist es, möglichst viele schon entwickelte und getestete Teile von Softwarekomponenten wiederzuverwenden. Man unterscheidet zwischen plattformunabhängigen Teilen, welche über viele Fahrzeuge hinweg eingesetzt werden können und fahrzeugspezifischen Teilen, welche für ein spezielles Fahrzeug verwendet werden, [8].

Die Softwareentwicklung beginnt mit der Analyse und der Spezifikation der Software-Architektur, die seitens vorgegebener Konventionen verschiedener Automobilhersteller getroffen wurde. Es kommt oft zu einer Zuordnung der Softwarekomponenten zu unterschiedlichen Schichten, sodass ein Schichtenmodell entsteht. Innerhalb einer Ebene können Softwarekomponenten beliebig aufeinander zugreifen. Ebenen mit einem höheren Abstraktionsniveau können auf Schichten mit niedrigerem Niveau zugreifen, umgekehrt ist dies jedoch kaum bis gar nicht möglich. So greifen, wie in Abbildung 2.2 ersichtlich, Softwarekomponenten höherer Abstraktionsebenen, wie beispielweise „Anzeigeobjekte“ auf deren Treiber zu, die in niedrigeren Ebenen der Basissoftware angesiedelt sind. Die Schichten werden nach ihrem Abstraktionsniveau gegliedert. Sobald von einem höheren Niveau auf alle Niedrigeren zugegriffen werden kann, spricht man von einem *Schichtenmodell mit strikter Ordnung.* Kann man hingegen immer nur auf die nächstniedrigere Schicht zugreifen, wird dies als *Schichtenmodell mit linearer Ordnung* bezeichnet. Ein Beispiel dieser Art ist das 7-Schichtenmodell nach ISO/OSI [9] das jenem von AUTOSAR.

ähnelt, vergleiche Kapitel 2.1.4. Auf Grund der Abstraktionsebenen wird die Erstellung, Wartung und Wiederverwendung der Softwarekomponenten vereinfacht. Als Beispiel dient der Entwurf für die Softwarearchitektur für das Kombiinstrument, siehe Abbildung 2.2, [1].

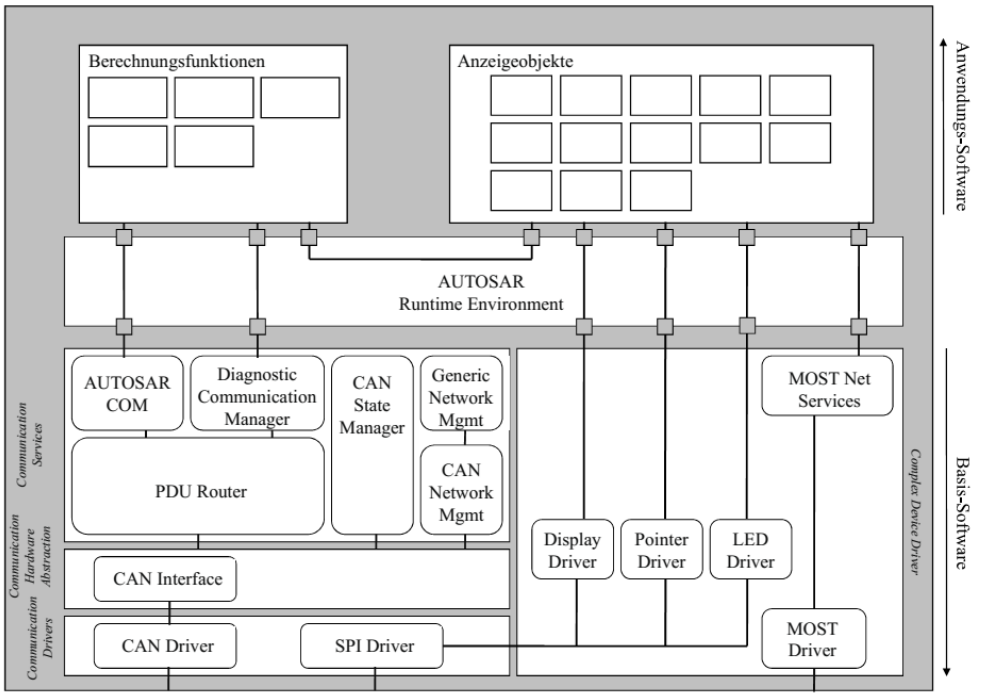


Abbildung 2.2: Softwarearchitektur des Kombi-Instruments, [1]

2.1.4 AUTOSAR

Zum Austausch verschiedener Softwareteile wurden einige Standards entwickelt, wie zum Beispiel der AUTOSAR- Standard.

AUTOSAR steht für Automotive Open System Architecture und wurde im Herbst 2003 gegründet. Der Standard ist eine Entwicklungspartnerschaft aus Fahrzeugherstellern, Steuergeräteherstellern, Herstellern von Entwicklungswerkzeugen, Steuergeräte-Basis Software und Mikrocontrollern. Auf Grund der Einführung einer einheitlichen Softwarearchitektur ist das Ziel von AUTOSAR die Erleichterung des Austausches von Softwareelementen auf verschiedenen Steuergeräten, [3].

Der AUTOSAR-Standard wurde von mehreren Automobilherstellern vereinbart, um sicherzustellen, dass Softwarekomponenten wiederverwendet, ausgetauscht, skaliert und integriert werden. Besonders wichtig für den AUTOSAR-Standard ist die Teilung in die steuergerätspezifische Basis-Software (BSW), die steuergeräteunabhängige Anwendungs-Software (ASW) und deren Verbindung über den virtuellen Funktionsbus (VFB), siehe Abbildung 2.3. Des Weiteren sind die funktionalen Softwarekomponenten (SWC) strikt voneinander und von der BSW getrennt. Diese SWC enthalten spezifische Regelalgorithmen (Runnable Entities) und kommunizieren über die AUTOSAR-Schnittstelle mit den übrigen Funktionen. Diese Schnittstellen (API) sind in SWC-XML-Beschreibungen bestimmt. XML steht für Extended Markup Language und ist eine Aufzeichnungssprache, [8].

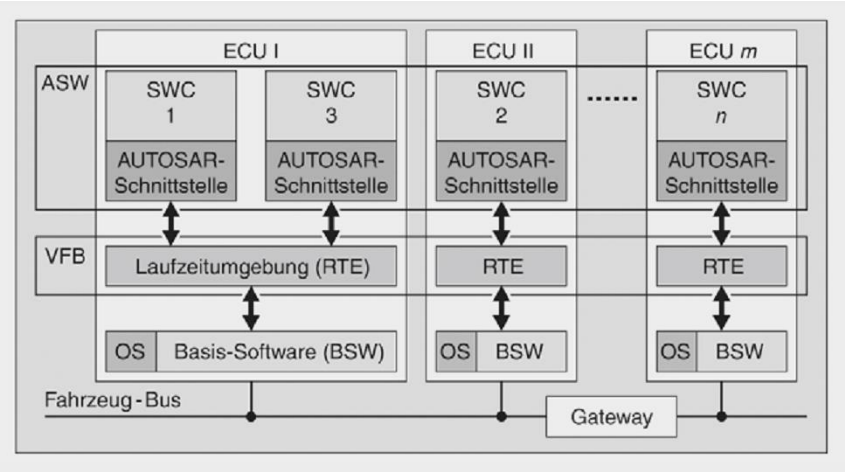


Abbildung 2.3 AUTOSAR-Architektur, [8]

AUTOSAR definiert lediglich die Interfaces und das Datenformat. Es verbindet die verschiedenen Softwarekomponenten miteinander und organisiert den Informationsaustausch dieser. Der Designer muss sich so keine Gedanken um die Software Infrastruktur machen und kann sich auf die Applikation konzentrieren. AUTOSAR verlinkt die Softwarekomponenten miteinander und weist diesen eine bestimmten ECU zu, siehe Abbildung 2.4, [10].

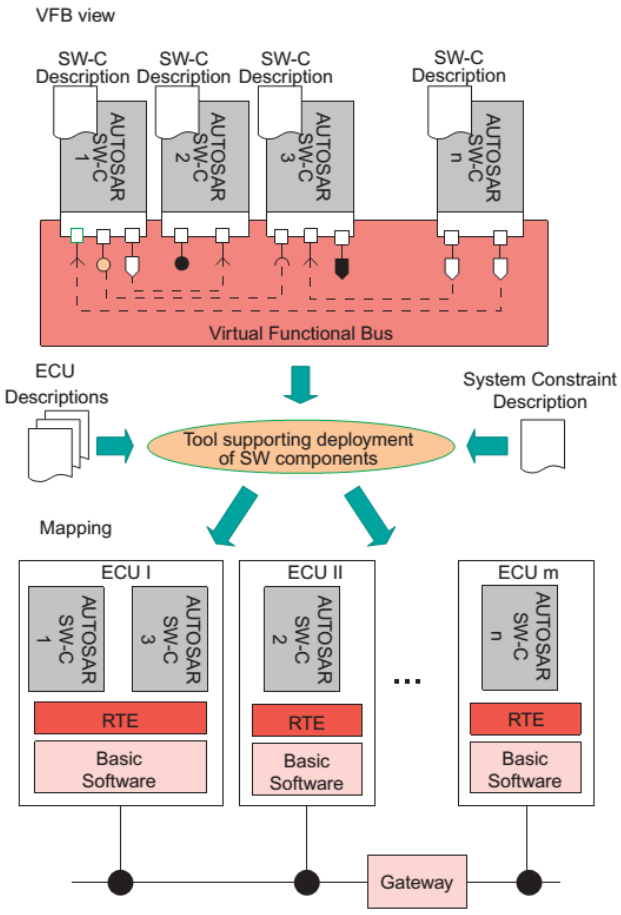


Abbildung 2.4: Funktionsweise des Virtual Bus, [10]

Hierfür wird in vier Schritten wie folgt vorgegangen, [10]:

1. Beschreibung der Softwarekomponenten hinsichtlich Software, Hardware und System
2. Verteilung der SWCs anhand der Beschreibung in Punkt 1.
3. ECU Konfiguration (RTE und BSW).
4. Erstellen eines ausführbaren Codes.

Auf Grund der AUTOSAR Methodik wird die Integration der Anwendungssoftware in einem Steuergerät wesentlich erleichtert. So entfällt eine manuelle Anpassung der Software. Nachteile ergeben sich hinsichtlich erhöhten Speicherbedarfs und höherer Rechenleistung aufgrund des geforderten AUTOSAR Standards was sich in zusätzlichen Kosten in der preisgetriebenen Automobilindustrie wiederspiegelt, [3], [10], [11].

2.1.5 ASIL [12]

Diese Norm behandelt alle sicherheitsbezogenen Systeme die in der E/E-Architektur in PKWs mit höchstzulässigem Gesamtgewicht von 3500kg verankert sind. ISO 26262 [9] bezieht sich auf mögliche Ausfälle oder Fehler, ausgelöst von E/E sicherheitsrelevanten Systemen bzw. Interaktion mit ihnen. Hierfür werden sogenannte Automotive Safety Integrity Levels (kurz ASIL) eingeführt, in denen Ausfallswahrscheinlichkeiten oder Risiken definiert sind. Es wird je nach Sicherheitsrisiko von ASIL A (niedrigste Stufe) bis ASIL D (höchste Stufe) unterschieden, [12], [13].

2.1.6 In the Loop Testsysteme

Bei den In-the-Loop Testsystemen handelt es sich um Testverfahren, bei denen ein elektronisches System über Schnittstellen entweder mit einer realen Umgebung, wie zum Beispiel mit Sensoren oder Aktoren, oder mit einem virtuellen Kontext, beispielsweise mit mathematischen Modellen, verbunden wird. Dabei wird die Reaktion des Systems analysiert und wieder an das System zurückgespielt. Man kann sowohl einzelne Steuergeräte, als auch ganze Steuergerätenetzwerke prüfen. Die Vorteile der In-the-Loop-Testverfahren sind die Erzielung höherer Flexibilität, größerer Testtiefe und Reproduzierbarkeit der Testfälle. Ein weiterer Vorteil gegenüber Tests am Prüfstand oder im Fahrzeug ist die Vorgabe von Betriebszuständen ohne Einschränkungen, wie zum Beispiel: bei einem Motorsteuergerät im vollständigen Last-Drehzahl-Bereich. Die meisten Tests lassen sich bereits automatisieren, wodurch möglichst viele Fehlerarten und deren Kombinationen exakt getestet und dokumentiert werden können. In-the-Loop-Testsysteme werden zur Validierung und zur Weiterbildung von Soft- und Hardware angewendet. Je nach Art des Gerätes, welches geprüft wird unterscheidet man zwischen folgenden Testsystemen, [8]:

- Model-in-the-Loop (MiL): Bei Model-in-the-Loop wird das Funktionsmodell der Software, das auf einem Entwicklungsrechner läuft, getestet.
- Software-in-the-Loop (SiL): Bei Software-in-the-Loop wird der Softwarecode, welcher auf einem Entwicklungsrechner läuft, getestet.
- Function-in-the-Loop (FiL): Bei Function-in-the-Loop wird wieder der Softwarecode getestet, doch im Gegensatz zum SiL läuft dieser auf der Zielhardware (also auf dem Steuergerät).
- Hardware-in-the-Loop (HiL): Bei Hardware-in-the-Loop wird das gesamte Steuergerät über die I/O Schnittstelle getestet.

2.2 Entwicklungsprozess gemäß V-Modell [14]

Das V-Modell wurde vom Bundesministerium für Verteidigung entwickelt und dient als Hilfsmittel, um die Erfolgswahrscheinlichkeit von Projekten zu verbessern. Seit der Einführung im Jahre 1992 wurde das V-Modell ständig erweitert unter den Namen V-Modell 97 bis hin zur aktuellen Version, dem V-Modell XT, welches im Februar 2005 veröffentlicht wurde, und kostenfrei zur Verfügung steht. Das V-Modell XT ist in den Bundesministerien Deutschlands für deren Durchführung von Entwicklungsprojekten verbindlich vorgeschrieben, [15].

Unter dem V-Modell versteht man die Prozessveranschaulichung der verschiedenen Phasen der Produktentstehung, welche von der Analyse der Anforderungen über die Entwicklung, die Implementierung und den Test bis hin zu dem Systemeinsatz reicht. [8] Der Vorteil des V-Modells ist, dass sowohl Software- also auch Hardwareentwicklung gleichermaßen berücksichtigt wird. Auch wird die Qualitätsprüfung in der Systemherstellung integriert. Bevorzugt wird das V-Modell bei Systemen mit hohen Zulässigkeits- und Sicherheitsanforderungen eingesetzt, da bei diesen Systemen genaue Prüfschritte vorgeschrieben sind, [1], [14].
Für die Fahrzeugentwicklung werden zunächst Komponenten hergestellt, die in Experimentalfahrzeugen untersucht werden. Basierend auf diesen Ergebnissen werden Prototypen entwickelt. In der Folge beschleunigen weitere Messungen die Entwicklung von Vorserien- und Serienfahrzeugen. Dieser Prozess hält solange an, bis alle Anforderungen oder Qualitätsziele erfüllt sind. Dieser Vorgang wird auch oft als Protoptypen- oder Spiralenmodell bezeichnet. Auf Grund der zunehmenden Anzahl der Iterationen nehmen sowohl das Entwicklungsrisiko als auch der Aufwand pro Iteration ab, [1].
Ein Beispiel für die Vorgehensweise bei einer automotiven Produktentwicklung nach dem V-Modell liefert Abbildung 2.5. Dabei werden im linken Ast die Anforderungen und Spezifikationen vereinbart und im rechten Ast die implementierten Anforderungen getestet. Im Sinne des V-Modells ist in der Abbildung ebenfalls die immer detailreicher werdende Aufspaltung der Anforderungen und Spezifikationen nach unteren Ebenen erkenntlich. Die Masterarbeit beschäftigt sich vorwiegend mit der untersten Ebene, der Komponentenentwicklung.

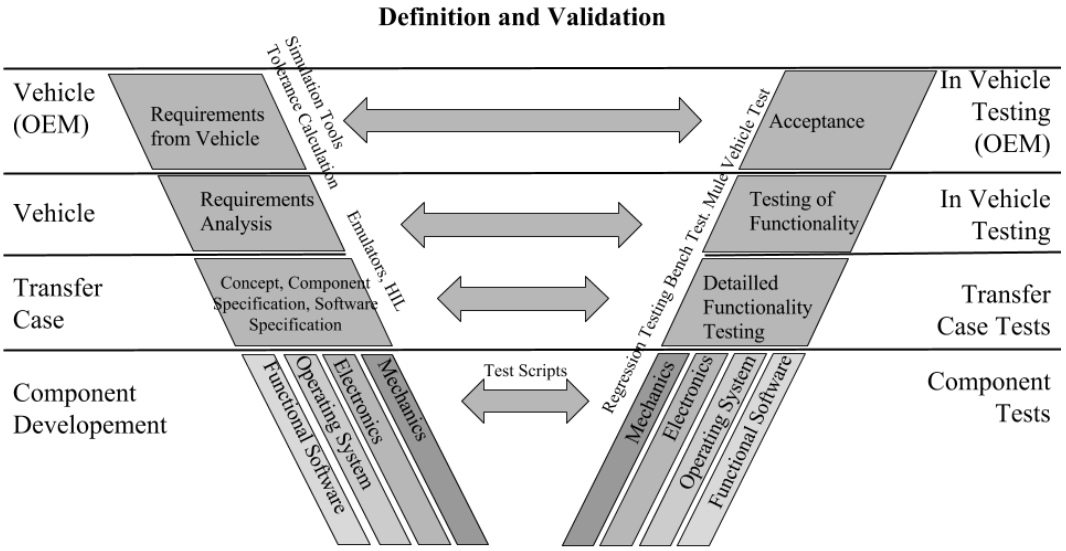


Abbildung 2.5: V-Modell Übersicht, vgl. [14]

Ziele des V-Modells:

Durch die Vorgabe standardisierter Vorgehensweisen sowie eindeutig definierter Rollen und durch die Beschreibung von Zwischenergebnissen wird die Projekttransparenz erhöht. Somit werden frühzeitig Abweichungen vom Entwicklungsprozess erkannt und Risiken früher aufgedeckt. Die Folge ist eine *Minimierung des Projektrisikos*. Ein weiteres Ziel ist die *Verbesserung und Gewährleistung der Qualität,* denn aufgrund der vielen Zwischenergebnisse beim V-Modell wird garantiert, dass die zu liefernden Ergebnisse vollständig und von gewünschter Qualität sind. Mit Hilfe des V-Modells kommt es zu einer *Eindämmung der Projekt- und Systemlebenszykluskosten,* denn es lassen sich sowohl der Aufwand als auch die Entwicklungen, die Herstellung, der Betrieb, die Pflege und die Wartung eines Systems genau steuern und überwachen. Des Weiteren sind die Ergebnisse einheitlich und besser nachvollziehbar. Außerdem bringt dieses Modell eine *kontinuierliche Verbesserung der Projektfähigkeit* durch die ständige Entwicklung neuer Prototypen. Ein weiteres Ziel ist die *Verbesserung der Kommunikation* zwischen allen Projektbeteiligten durch die standardisierte Beschreibung aller nötigen Bestandteile. Dadurch können Missverständnisse zwischen den einzelnen Projektbeteiligten, wie zum Beispiel Nutzer, Auftraggeber, Auftragnehmer oder Entwickler, reduziert werden, [1], [15].

Das V-Modell XT bietet für drei unterschiedliche Projekttypen eine Richtlinie für die Organisation und Durchführung von Projekten, [15]:

1. **Systementwicklungsprojekt eines Auftraggebers:**
   Dieser Projekttyp schildert die Erstellung einer Ausschreibung und wählt basierend auf den Angeboten einen Auftragnehmer aus. Der Auftragnehmer produziert und liefert das vom Auftraggeber gewünschte System.
2. **Systementwicklungsprojekt eines Auftragnehmers:**
   In diesem Fall wird während des Projekts ein Angebot erstellt und bei einem Vertragsabschluss ein System produziert. Danach wird das System zur Abnahme an den Auftraggeber geliefert.
3. **Einführung und Pflege eines organisationsspezifischen Vorgehensmodells:**
   Dieser Projekttyp hat die Aufgabe, das V-Modell zu analysieren und im Falle einer Komplikation Verbesserungsvorschläge auszuarbeiten.

Da es unterschiedliche Projekttypen gibt, muss zunächst die Bestimmung des Projekttyps erfolgen. Dieser fordert zwingend *Vorgehensbausteine* und *Projektdurchführungsstrategien*. Ludewig und Lichter [4] beschreiben vier obligatorische Vorgehensbausteine, wie

- Projektmanagement,
- Qualitätssicherung,
- Problem- und Änderungsmanagement sowie
- Konfigurationsmanagement.

Weitere Vorgehensbausteine können projektspezifisch hinzugefügt werden. Diesen Vorgang bezeichnet man auch als *Tailoring*. Bei diesem Prozess werden die Abhängigkeiten der einzelnen Bausteine ersichtlich. Vorgehensbausteine beinhalten *Produkte* (WAS), *Aktivitäten* (WIE) und *Rollen* (WER), die für die Erfüllung einer bestimmten Aufgabenstellung notwendig sind. In Abbildung 2.6 sind die Abhängigkeiten illustriert. Die Projektdurchführungsstrategie (WANN) regelt die zeitliche Abfolge der Projektphasen um eine zuverlässige Planung und Steuerung des Projektes zu ermöglichen. Es kann zu Beginn des Projektes in allen Projektabschnitten gestartet werden. Für die Beendigung eines nächstfolgenden Projektabschnittes ist jedoch der Abschluss des vorherigen Abschnittes erforderlich, welcher durch einen *Entscheidungspunkt* gekennzeichnet ist. Zum erfolgreichen Abschluss eines

Entscheidungspunktes müssen ein oder mehrere Produkte abgeschlossen sein, wie Abbildung 2.7 verdeutlicht, [4], [15].

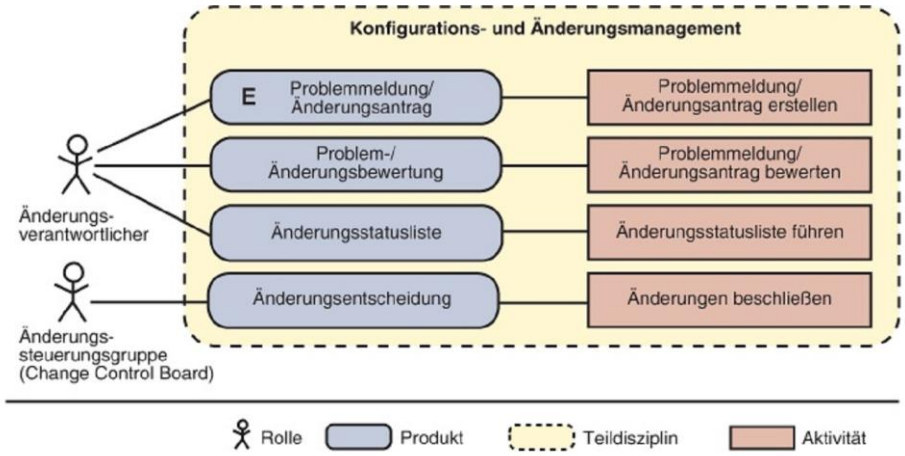


Abbildung 2.6: Vorgehensbaustein, [4]

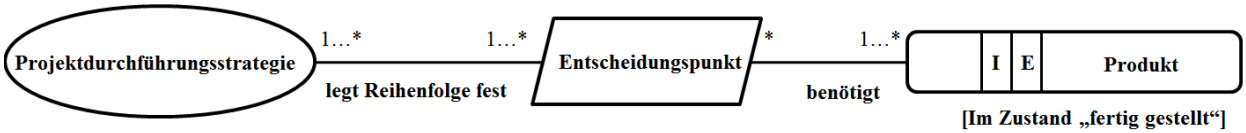


Abbildung 2.7: Projektdurchführungsstrategie und Entscheidungspunkt, vgl. [15]

2.3 Business Intelligence Schichtenmodell

Die IT-basierte Unterstützung des Managements, kurz „Business Intelligence“, ist kein neuer Begriff und wurde schon in den 60er Jahren von H.P. Luhn beschrieben, [16]. Es gab immer wieder Versuche, die Führungsebene mit Hilfe von Informationssystemen zu unterstützen. Die Definition von „Business Intelligence“ ist sehr weitläufig und wird laut Kemper et al. [17] wie folgt definiert:

> *„Business Intelligence (BI) bezeichnet einen integrierten, unternehmensspezifischen, IT-basierten Gesamtansatz zur betrieblichen Entscheidungsunterstützung.“*

Dabei bezieht sich das Wort „Intelligence“ nicht auf die deutsche Bedeutung des Wortes Intelligenz, sondern soll die gewonnenen Erkenntnisse und Informationen aus der Analyse von Daten beschreiben. Eine Business Intelligence Lösung ist oft in Form von Schichten aufgebaut, vergleiche Abbildung 2.8, [18]:

- *Quellen*: Diese können aus externen oder internen Systemen kommen und umfassen Daten unterschiedlichster Stuktur. Interne Quellen können beispielsweise Application Lifecycle Tools wie das vom Industriepartner verwendete System PTC MKS sein oder Meilensteinlisten in Tabellenform, die mit dem Management vereinbarte Ziele enthalten.
- Bei der *Datenversorgung* werden syntaktische Mängel wie Formatanpassungen oder Formatinkompatibilitäten beseitigt und die Daten harmonisiert.
- Das *Datenmanagement* mit dem Data Warehouse ist der Kern einer Business Intelligence Anwendung. Ein Data Warehouse beinhaltet logisch zentralisierte

dispositive Datenbestände, die Informationen für Entscheidungen des Managements beinhalten und stehen im Gegensatz zu operativen Daten, die sich beispielsweise mit allgemeinen Daten zu Produktion oder Auftragsabwicklung befassen. Wichtige Kriterien für ein funktionierendes Data Warehouse sind:

- Die Themenorientiertheit, die Daten bestimmten Unternehmens- oder Produktstrukturen zuordnet.
- Die Integration, welche nur entscheidungsrelevante Daten aus operativ gewachsenen Daten extrahiert.
- Der Zeitraumbezug: Da operative Daten oft transaktionsorientiert und zeitpunktsbezogen sind, ist es wichtig, dass Informationen im Data Warehouse zeitraumbezogen sind, beispielsweise über einen Monat, einer Woche oder einen Tag.
- Die Nicht-Volatilität sagt aus, dass Daten dauerhaft historisch abgespeichert werden müssen, um diese auch zu einem späteren Zeitpunkt analysieren zu können. Jedoch sollte aufgrund dieser speicherintensiven Vorgehensmethoden Historisierungskonzepte entwickelt werden, um das Datenwachstum zu begrenzen.

- Das *Metadatenmanagement* verwaltet die im Datenpool gespeicherten Informationen während das *Warehouse Management* die Infrastruktur der BI Software bereitstellt.
- Die oberste Schicht ist die *BI-Anwendung,* welche die Informationen durch gezielte (SQL-)Abfragen an die Datenbank aufbereitet und dem Benutzer die Ergebnisse in Form von Tabellen oder Grafiken darstellt.

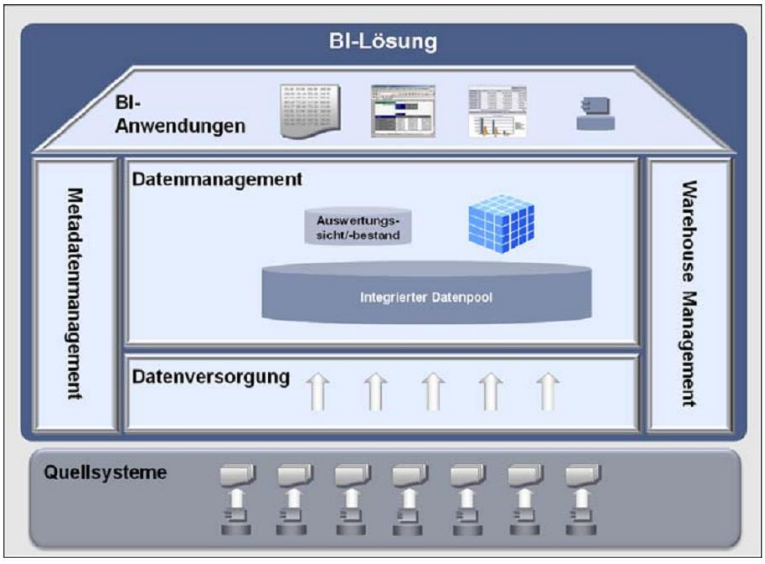


Abbildung 2.8: Business Intelligence Schichtenarchitektur, [18]

Diese vorliegende Masterarbeit befasst sich vorwiegend mit der Erstellung der „BI-Anwendung“ und übernimmt auch Aufgabengebiete des Datenmanagements (*Data Warehouse*) durch gezieltes Trennen von operativen und distributiven Datensätzen in den Abfragen. Die unteren Schichten, wie Datenversorgung oder Quellen, wurden teilweise vorgegeben oder wurden bereits implementiert. Grünwald et al. [18] unterscheidet bei der Erstellung von BI-Anwendungen folgende Kategorien, die aber durchaus inhaltlich miteinander verschmelzen, vergleiche Abbildung 2.9:

Reporting und Analyse

Mit standardisierten Reports in Form von Tabellen oder Diagrammen werden Quartalsberichte erstellt und an die betreffenden Personen automatisiert verteilt.

Performance Management

Die aktuelle Unternehmensleistung wird mit der Unternehmensstrategie in Interaktion versetzt. Durch eine systematische Leistungsmessung werden Kennzahlen (Key Performance Indikatoren, vergleiche Kapitel 2.4) erstellt. Bei Abweichungen vom Ziel werden Korrekturmaßnahmen geplant und ausgeführt.

Analytische Anwendungen

Mit am Markt verfügbaren Standardsoftwarekomponenten wird der interessierende Themenbereich, wie beispielsweise Risikomanagementanwendungen bei Finanzdienstleistern, untersucht.

Planung und Simulation

Es werden Prognosen aufgrund von Istwerten der Vorperiode erstellt. Ein rein lesender Zugriff auf die Daten reicht für eine Simulation zukünftiger Planzahlen nicht aus. Unter Zuhilfenahme von statistischen Methoden können Erwartungswerte prognostiziert und der Planungsprozess unterstützt werden.

Abbildung 2.9: BI-Anwendungen, [18]

Alle oben beschriebenen Kategorien sind miteinander kombinier- und anwendbar. Diese Synergien zwischen den Kategorien sollen bei Analyse der Entwicklungs- und Testdaten genutzt werden und bei der Erstellung des Analysetools miteinfließen.

2.4 Kennzahlen zur Beurteilung des Entwicklungsprozesses

Die Definition und softwaretechnische Implementierung von Kennzahlen zur Bewertung des Entwicklungsprozesses bildet den Kern dieser Arbeit und wird somit in den Grundlagen behandelt. Die Beurteilung des Entwicklungsprozesses erfolgt häufig mit Maßzahlen, sogenannten *Metriken* oder *Key Performance Indikatoren*. Diese sollen während der Projektabwicklung die verantwortlichen Personen, beispielsweise Softwareentwickler oder Projektmanager unterstützen, und mögliche Abweichungen vom Ziel aufzeigen. Um ein bereits entwickeltes Produkt zu verbessern oder es mit anderen zu vergleichen bedarf es einer genauen Beschreibung des Produkts und dessen Herstellungsprozess. Dabei ist es essentiell, dass jede Metrik oder jeder KPI einem konkreten Zweck dient, was einerseits die Kenntnis der konkreten Fragestellung, andererseits die Kenntnis der Zielsetzung voraussetzt. Die Definition einer Metrik lautet laut IEEE wie folgt, [4], [19].

Metrik:

> *"A quantitative measure of the degree to which a system, component, or process possesses a given attribute",* [20].

Ähnlich lautet auch die Definition des Key Performance Indikators:

> *"A measurement of the performance of a particular business system in terms of the aims and goals of an enterprise",* [21].

In der Literatur verschmelzen die beiden Begriffe öfters. Festzuhalten ist, dass KPIs das (Projekt)-Management betreffen und Hilfestellungen bei strategischen Entscheidungen leisten, während Metriken in Softwareprojekten eingesetzt werden und den Softwareentwickler unterstützen.

Wichtige Ziele, die KPIs oder Metriken beantworten sollen, sind, [4], [19]:

- Bewerten der Qualität von Produkten und Prozessen,
- Quantifizieren von Erfahrungen,
- Prognosen,
- Unterstützen von Entscheidungen.

Für die Erstellung von Key Performance Indikatoren sind die Fragestellungen, die sie beantworten sollen, von entscheidender Bedeutung. Diese *Key Performance Questions (KPQ)* sollen nicht nur auf den vergangenen Entwicklungsstatus abzielen, sondern auch auf die gegenwärtige Situation und Zukunft verweisen. So wäre eine beispielhafte KPQ „Erhöhen wir unseren Marktanteil gerade?" eine bessere Fragestellung als „Haben wir unseren Marktanteil erhöht?" um die künftige zu erwartende Situation ebenfalls miteinzubeziehen. Des Weiteren gibt es geschlossene und offene Fragestellungen. Geschlossene Fragestellungen sind kurz und prägnant zu beantworten, während offene Fragestellungen die eigene Meinung zulassen. Deshalb sollen offene Fragestellungen bevorzugt werden, [22].

Eine Vorgehensweise für die Gestaltung von Key Performance Indikatoren liefert Abbildung 2.10. Diese Methode wird als *Goal Question Metric* bezeichnet und zeigt eine Vorgehensweise für eine Erstellung von Metriken oder KPIs auf. Ausgehend von definierten Zielen wird nach dem Top-Down Prinzip mit Fragen bis auf die Ebenen konkreter Metriken verfeinert. Die Auswertung und Interpretation erfolgt danach im Bottom-Up Verfahren zur Beantwortung der Fragen. Die Erreichbarkeit der Ziele lässt sich davon ableiten, [19].

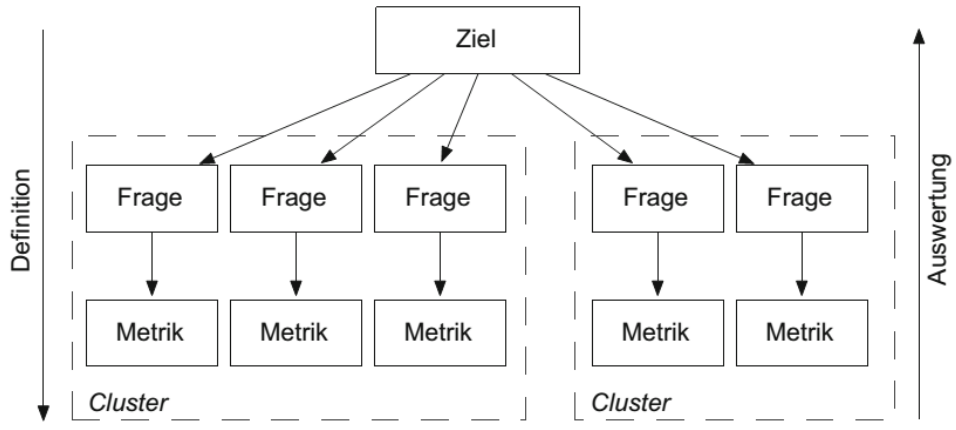


Abbildung 2.10: Struktur Goal Question Metrik, [19]

Um Fragen in Form von Key Performance Indikatoren zu beantworten, sind Messungen erforderlich. Nach Untersuchungen von Messprogrammen fasst Wallmüller [23] seine Erkenntnisse in zwei Punkten zusammen:

- Was nicht vorher gemessen wurde, kann nicht gesteuert oder geregelt werden.
- Nicht so genau wie möglich messen, sondern so genau wie nötig.

Mit aktuellen Application Lifecycle Programmen, vergleiche Kapitel 3, fallen im Laufe des Entwicklungsprozesses umfassende Datenmengen an. Zur Erstellung einer optimalen Metrik sind gezielte Messungen und Datenerhebungen nötig. Sowa [24] definiert folgende Voraussetzungen, um eine optimale Messung zu erstellen:

- Gemessen werden aussagekräftige bzw. für das Unternehmen bedeutende Werte.
- Messungen sind reproduzierbar.
- Messungen sind objektiv und unparteiisch.
- Die Messungen stellen den Fortschritt in der Zielrichtung im Zeitablauf dar.

Um Messungen durchzuführen, bedarf es ausreichenden Datenmaterials, die oft auf großen Datenbanken gespeichert ist. Die vier Phasen der Datenerhebung sind laut Sowa [24]:

- Vorbereitung und Planung,
- Datenerhebung,
- Datenaufarbeitung,
- Analyse des Datenmaterials und Veröffentlichung der Ergebnisse.

Nach der Messung müssen die aufbereiteten Ergebnisse präsentiert werden. Dies geschieht meist in Form von Diagrammen und Tabellen, aber auch Ampelsysteme werden eingesetzt. Dabei verändert ein Symbol (Kreis, Herz, Ampel oder ähnliches) seine Farbe je nach Überschreiten bestimmter Schwellwerte. Beispielsweise kann rot ein hohes Projektrisiko, gelb ein mittleres Projektrisiko und grün wenig Projektrisiko bedeuten. Je nach Einsatzzweck stehen verschiedene Diagrammarten zur Verfügung, wie in Abbildung 2.11 illustriert, [24], [25].

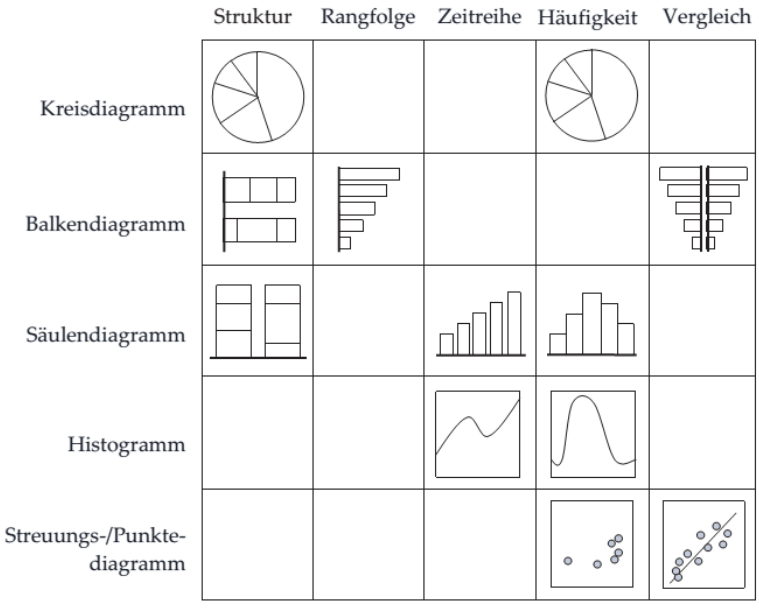


Abbildung 2.11: Mögliche Diagrammarten für KPIs, [24]

Kreisdiagramme werden häufig zur Darstellung relativer Häufigkeiten verwendet, während *Balken- oder Säulendiagramme* sowohl zur absoluten Darstellung bestimmter Messgrößen dienen als auch zur Visualisierung von relativen Häufigkeiten. Der Unterschied der beiden einfachen Diagrammarten ist, dass das Anwachsen der Häufigkeiten horizontal oder vertikal dargestellt wird. Beide Arten von Diagrammen können gestapelt, gruppiert oder nach Häufigkeit ab-/aufsteigend angezeigt werden. Das Säulendiagramm eignet sich für zeitlich fortschreitende Messungen. *Histogramme* teilen bestimmte Bereiche in Klassen ein, die dann gruppiert angezeigt werden. Diese Diagrammart eignet sich für große Datenmengen. *Liniendiagramme* können für einen fortschreitenden relativen oder absoluten Erfüllungsgrad bezogen auf eine Schranke (100% Marke oder einen vordefinierten Wert) verwendet werden. *Punktdiagramme* eignen sich zur zweidimensionalen Darstellung von Werten, im Besonderen wenn es um Korrelationen und Abhängigkeiten zweier Metriken geht, werden diese oft verwendet. Abschließend wird die Darstellung in Form eines *Tachometers* genannt, welche die aktuelle Performance bezogen auf definierte Schwellwerte anzeigt, siehe Abbildung 2.12, [24], [25].

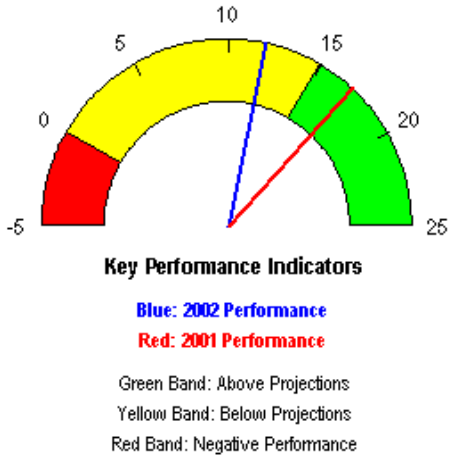


Abbildung 2.12: Visualisierung eines KPIs mittels Tachometer-Anzeige, [26]

2.5 Werkzeuge für die softwaretechnische Umsetzung

Für die softwaretechnische Implementierung des Analysetools bedarf es verschiedener Werkzeuge und Sprachen. Als Werkzeug für die Erstellung eines ausführbaren Programms wird die objektorientierte Programmiersprache VB.NET[1] [27] in der Entwicklungsumgebung Visual Studio 2013[2] [27] herangezogen. Die Abfragen an die Datenbank geschehen mit der Sprache SQL (Structured Query Language) [12]. Die erforderlichen Grundlagen dieser beiden Sprachen werden im Folgenden erläutert, dabei wird zuerst auf das Programm Visual Studio [27] und dessen Handhabung mit Datenbanken eingegangen. Danach werden die Darstellungsmöglichkeiten der Ergebnisse der SQL-Statements in der GUI in Visual Studio näher beschrieben. Abschließend wird der Begriff relationale Datenbank genauer untersucht und einige Abfragen in SQL erklärt.

VB.NET ist eine Weiterentwicklung von Visual Basic [27] und beinhaltet die .NET

[1] Microsoft .NET Framework, Version 4.5.51650, Microsoft Corporation, 2013
[2] Microsoft Visual Studio Ultimate 2013, Version 12.0.30501.00, Microsoft Corporation 2013

Technologie, welche im Jahr 2002 eingeführt worden ist. Diese .NET-Technologie ist eine gemeinsame Plattform vieler Programmiersprachen (C# [27], VB.NET, C++ [12]), die beim Kompilieren den Quelltext in eine MSIL (Microsoft Intermediate Language) [27] übersetzt und ein gemeinsames Laufzeitprogramm, die CLR (Common Language Runtime) [27] benutzt. Der Maschinencode wird dann vom Just-In-Time Compiler generiert, siehe Abbildung 2.13, [28].

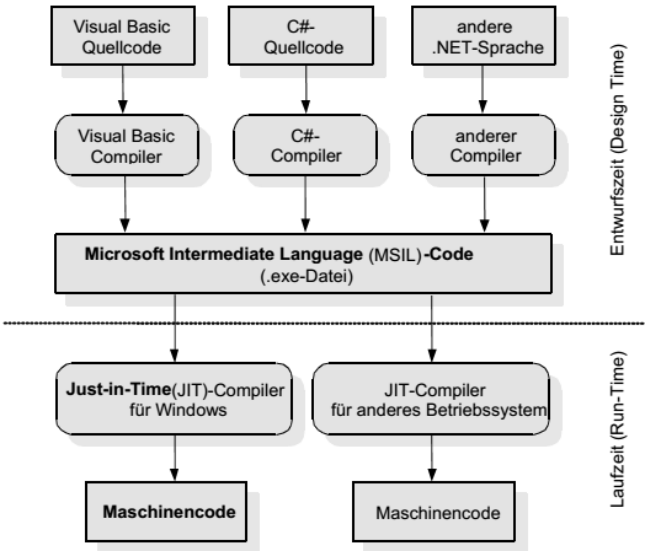


Abbildung 2.13: Codegenerierung VB.NET, [28]

Ein Hauptgrund für die Verwendung der .NET-Technologie ist die Zuhilfenahme der Datenzugriffstechnologie ADO.NET [27], welche vorgefertigte Klassen für die Kommunikation mit der Microsoft Access[3]-Datenbank [27] beinhaltet. Kernelement der ADO.NET [27] Klassenbibliothek bildet das *DataSet*, welches die strikte Trennung von Datenhaltung und Datenbank vorsieht, siehe Abbildung 2.14. Die Datensätze werden in der Datenbank abgeholt und als DataSet in den Arbeitsspeicher gepuffert. Das DataSet kann auch aus mehreren Tabellen (*DataTable*) bestehen und verknüpft deren etwaige Relationen (*Relation*) zueinander. Um mit der Datenbank kommunizieren zu können wird ein .NET-Datenprovider benötigt, in welchem alle vorgefertigten Klassen implementiert sind. Diese beginnen mit *OleDb* und haben in Analogie zu Abbildung 2.14 folgende Aufgaben, [28], [29]:

- *OleDbConnection*: Stellt die Verbindung zur Datenquelle her.
- *OleDbCommand*: Führt die SQL Abfrage aus.
- *OleDbDataAdapter*: Ermöglicht das Füllen des Datasets mit den Ergebnissen der SQL-Abfrage
- *OleDbDataReader*: Ermöglicht einen sequenziellen Nur-Lese-Zugriff auf die Datenquelle.

[3] Microsoft Access 2010, Version 14.0.7166.5000 (32 Bit), Microsoft Corporation, 2010

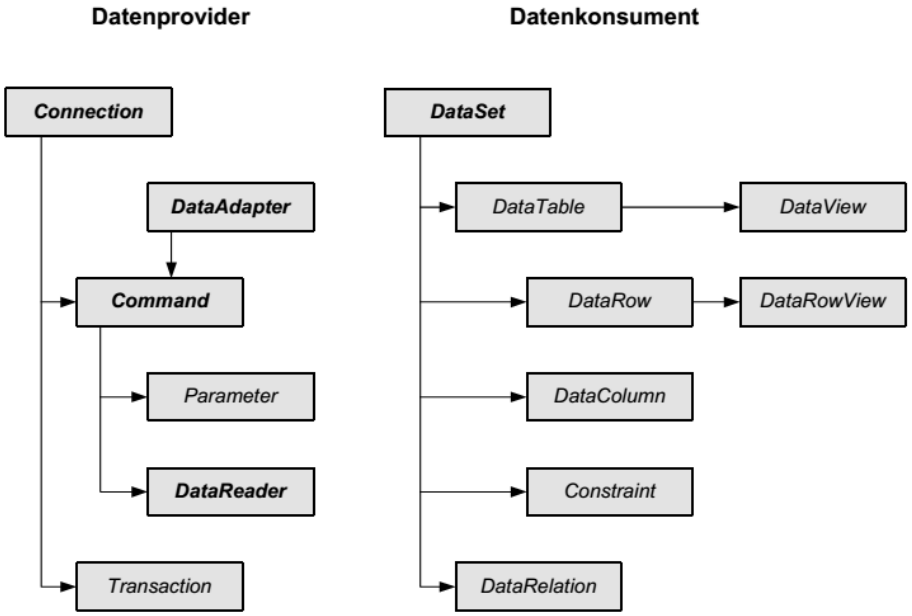


Abbildung 2.14: Datenprovider und Datenkonsument, [28]

Der Anwender erspart sich nun erheblichen Programmieraufwand, welcher sich auf folgende Punkte reduziert, [28]:

- Die Herstellung der Verbindung (*Connection*).
- Die Ausführung SQL-Abfrage.
- Das Befüllen des *DataSets* mittels des *DataAdapters*.
- Die Präsentation der Abfrage mit Diagrammen oder Tabellen.

Die Ergebnisse der Abfragen werden in einem *DataView* gespeichert, siehe Abbildung 2.14, und mit *Microsoft Chart Control* in der Benutzeroberfläche visualisiert. Darin sind über 35 verschiedene Diagrammtypen in 2D/3D implementiert und können ohne großen Programmieraufwand angesprochen werden. Datenpunkte (von der *DataView* kommend) werden über die simple *Points.AddXY*-Methode hinzugefügt, [29].

Eine prinzipielle Vorgehensweise für die manuelle Datenbindung mit Microsoft Access sowie den Chart-Aufbau, wie sie im gesamten Projekt für jeden Key Performance Indikator geschieht, wird im weiteren Verlauf beschrieben. Dazu wird ein Beispiel aus dem Buch „Datenbankprogrammierung in Visual Basic 2012“ übernommen, [29]:

Gewünscht wird ein Diagramm, welches je nach Verkäufer die Abhängigkeit der Verkaufssumme von der Jahreszahl zeigt.

Zunächst lesen wir auf gewohnte Weise die gewünschten Daten in eine DataTable ein:

```
Dim sql As String = "SELECT * FROM PKW_Verkauf"
Dim conn As New OleDbConnection(connStr)
Dim cmd As New OleDbCommand(sql, conn)
Dim da As New OleDbDataAdapter(cmd)
Dim ds As New DataSet()
conn.Open()
da.Fill(ds, "query1")
conn.Close()
Dim dt As DataTable = ds.Tables("query1")
```

Alle Verkäufer durchlaufen:

```
For Each row As DataRow In dt.Rows
```

Pro Verkäufer eine neue Serie hinzufügen:

```
    Dim serName As String = row("Verkäufer").ToString()
    Chart1.Series.Add(serName)
```

Jeder Verkäufer = eine Serie von Punkten = eine Linie:

```
    Chart1.Series(serName).ChartType = SeriesChartType.Line
    Chart1.Series(serName).BorderWidth = 5
```

Alle Jahres-Spalten durchlaufen:

```
    For colNr As Integer = 1 To dt.Columns.Count - 1
```

Pro Jahres-Spalte den Y-Wert als Punkt hinzufügen:

```
        Dim YVal = CDec(row(colNr))
```

Für X-Achse (Beschriftung!):

```
        Dim colName = dt.Columns(colNr).ColumnName
        Chart1.Series(serName).Points.AddXY(colName, YVal)
    Next colNr
Next row
```

Kontrollanzeige der DataTable im Datengitter:

```
DataGridView1.DataSource = ds
DataGridView1.DataMember = "query1"
...
```

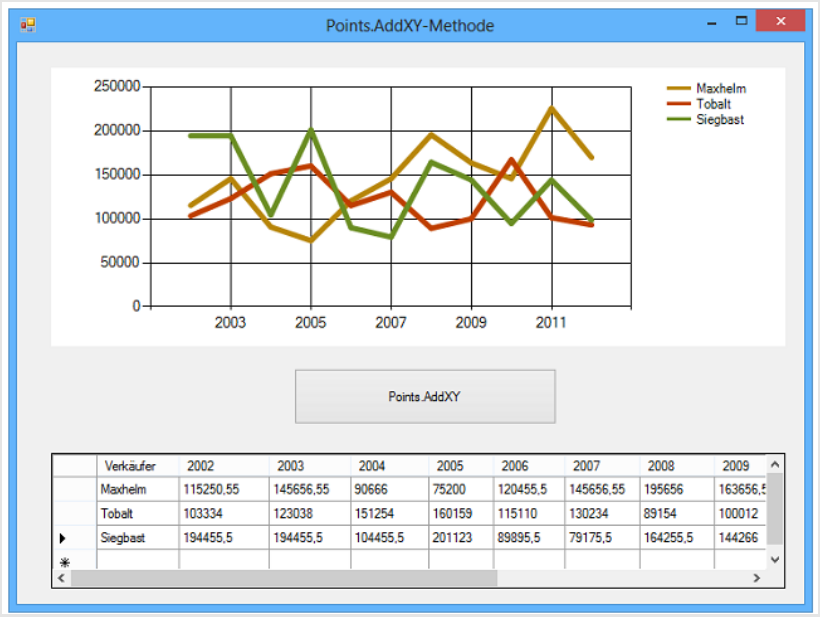


Abbildung 2.15: Points.AddXY-Methode, [29]

Die Datenstruktur von Datenbanken ist meist von relationaler Natur. Eine relationale Datenbank beinhaltet ausschließlich mit Daten gefüllte Tabellen (Relationen). Die Tabellen stehen in Beziehung zueinander, sind also logisch miteinander verknüpft. Dabei ist es wichtig, dass jede Relation einen eindeutigen Bezeichner besitzt, den sogenannten *Primärschlüssel*. Der

grundlegende Aufbau und wichtige Begriffe sind in Abbildung 2.16 ersichtlich. Weitere Begriffe einer relationalen Datenbank sind, [30]:

- Attribute, die Felder oder Spalten einer Tabelle.
- Tupel, eine Zeile einer Tabelle.
- Kardinalität, die Anzahl der Zeilen einer Tabelle.
- Grad, die Anzahl der Spalten einer Tabelle.
- Gebiet, die Menge aller möglichen Werte.

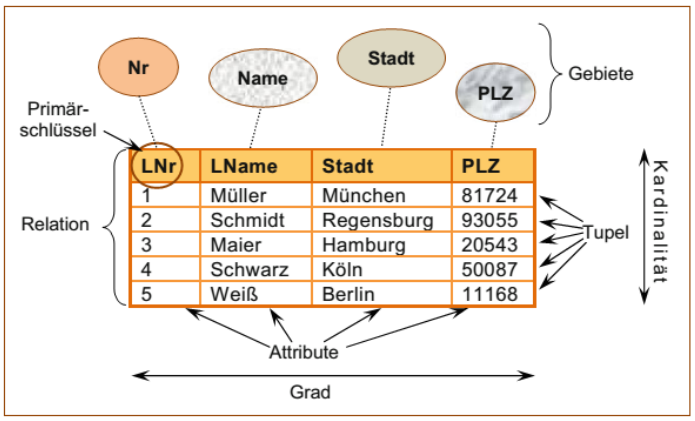


Abbildung 2.16: Wichtige Begriffe einer relationalen Datenbank, [30]

Wenn eine Datenbank optimal aufgeteilt ist, wird diese als *normalisiert* gekennzeichnet. Die Normalisierung einer relationalen Datenbank ist ein schrittweiser Prozess. Um Redundanzen und somit unnötigen Speicherbedarf zu vermeiden, werden drei wichtige Normalformen definiert, die der Spezialliteratur zu entnehmen sind und hier nicht weiter beschrieben werden. Der Zugriff auf die Datenbanken erfolgt lesend oder schreibend. Je nach Zugriffsart spricht man von drei Optionen, den

- Abfragen,
- Mutationen,
- Transaktionen, [30].

Abfragen greifen nur lesend auf die Datenbank zu, während Mutationen die Datenbank verändern. Transaktionen sind schreibende Zugriffe auf die Datenbank, welche die Konsistenz erhalten. Als Beispiel für eine Transaktion sei hier eine Banküberweisung angeführt. Kunde A überweist an Kunden B einen Geldbetrag, was in einer Minderung des Kontostandes von Kunden A resultiert. Dem Kunden B wird genau dieser Geldbetrag auf sein Konto gutgeschrieben. Sind nun beide Kunden in derselben Datenbank registriert so verändert sich der absolute Kontostand beider Konten nicht. Diese beiden Mutationen münden also in einer Transaktion, [30].

Es sei vorweggenommen, dass bei der Erstellung des Analysetools nur lesend auf die Datenbank zugegriffen werden darf, da es sich um heikle Entwicklungs- und Testdaten handelt. Deshalb werden bei der Erläuterung der SQL-Abfragen weiter unten nur lesende Begriffe beschrieben, [29], [30].

Die Zugriffssprache auf Datenbanken ist SQL und hat sich seit seiner ersten Verabschiedung 1986 (SQL1) als Standardzugriffssprache etabliert. Der aktuelle Standard SQL2 wurde 1992

veröffentlicht und wird von aktuellen Datenbanksystemen akzeptiert. SQL befasst sich mit der Bearbeitung und Auswertung von Datenbanken, wobei es je nach Datenbanksystem „SQL-Dialekte“ gibt, die es bei einer Wiederverwendung auf anderen Systemen zu beachten gilt, [31].

In den nachfolgenden Absätzen werden die wichtigsten SQL-Befehle erläutert, die für die Abfragen an die Datenbank benötigt werden, [30].

Der *„SELECT“*-Befehl bildet das Kernelement aller SQL-Abfragen. Mit ihm wird abgefragt WAS ausgegeben werden soll. So steht nach dem Bezeichner *„SELECT“*, welche Gebiete abgefragt werden sollen, oder wie im Beispiel unten das „*“ für alle Attribute, welche sich in einer Relation befinden. Falls nur bestimmte Gebiete abgefragt werden sollen, so werden die Gebiete durch einen Beistrich getrennt. Ein *„SELECT“*-Befehl benötigt eine obligatorische *„FROM“*-Klausel, die angibt, welche Relation ausgewählt wird. Diese beiden Klauseln sind für die Ausführung eines SQL-Befehls zwingend vorgeschrieben.

```
SELECT * FROM Personal;
```

Die *„WHERE“*-Klausel ist jedoch nicht verpflichtend, findet sich aber in Abfragen praktisch immer wieder. Sie beschreibt Restriktionen auf bestimmte Tupel einer Relation. So werden im folgenden Beispiel alle Angestellten angeführt, die ein Gehalt von mehr als 3000 Euro beziehen. In SQL können verschiedene Operatoren verwendet werden, die zusammengefasst in Tabelle 2.1 ersichtlich sind.

```
SELECT MIN Gehalt
FROM Personal
WHERE Gehalt > 3000;
```

Tabelle 2.1: Operatoren für die *„WHERE“*-Klausel, [30]

Boolesche Operatoren	NOT , AND , OR
Vergleichsoperatoren	< , <= , > , >= , = , <>
Intervalloperator	[NOT] BETWEEN ... AND
Enthaltenoperator	[NOT] IN
Ähnlichkeitsoperator	[NOT] LIKE
Auswahloperatoren	ALL , ANY , SOME
Existenzoperator	EXISTS
Nulloperator	IS [NOT] NULL

Die *„GROUP BY“*-Klausel fasst einzelne Tupel nach bestimmten Eigenschaften zusammen. Anhand der unteren SQL-Query soll diese Klausel erläutert werden.

```
SELECT Ort, COUNT (*) AS Anzahl
FROM Personal
GROUP BY Ort ;
```

Die Abfrage gruppiert nach den Wohnorten („Ort“) der Angestellten und zählt diese mittels *„COUNT“*. Als Ausgabe erscheint dabei eine Spalte mit den Wohnorten und die Anzahl der dort wohnenden Angestellten. Ähnlich zum *„COUNT“*-Befehlt agiert die *„DISTINCT“*-Klausel. Bei ihr wird bei mehreren gleichen Ergebniszeilen in einem Gebiet nur eine davon ausgegeben. In relationalen Datenbanken liegen die Daten oft ungeordnet vor. Der Benutzer möchte aber meist, dass die Abfrage auf- oder absteigend sortiert wird. Dies wird durch den *„ORDER BY“*-Befehl abgedeckt. Soll nach Anzahl der Bewohner des gleichen Ortes – wie im obigen Beispiel – absteigend geordnet werden, geschieht dies durch folgendes Statement:

```
ORDER BY Anzahl DESC, Ort;
```

„DESC" (im Gegensatz zu *„ASC"*) kennzeichnet die absteigende Sortierung und bei Gleichheit der Anzahl wird zusätzlich nach dem Ort (alphabetisch) geordnet.

Um ganze Tabellen miteinander verknüpfen zu können wird die *„UNION"*-Klausel verwendet. Dabei sind folgende wichtige Voraussetzungen der Tabellen zu erfüllen:

- Der Grad der Tabelle muss gleich sein.
- Die Attribute der Hauptteile müssen vom gleichen Datentyp sein.

Abschließend sei noch der wohl wichtigste Operator in SQL-Statements zu erwähnen, die *„JOIN"*-Klausel. Im Folgenden wird auf den Operator *„INNER JOIN"* eingegangen, da dieser in der Arbeit von besonderem Interesse ist. Die *„JOIN"*-Klausel benötigt immer einen *„ON"*-Befehl für die erfolgreiche Verbindung zwischen Relationen, wie im untenstehenden Beispiel ersichtlich ist.

```
SELECT *
FROM Auftrag INNER JOIN Personal
ON Auftrag.Persnr = Personal.Persnr ;
```

Hierbei werden die Tabellen „Auftrag" und „Personal" miteinander verknüpft unter der Voraussetzung, dass die Attribute in „Persnr" von der Relation „Auftrag" jener von der Relation „Personal" entsprechen.

Besonders wichtig und in der vorliegenden Arbeit häufig verwendet ist der Operator *„LEFT JOIN"*.

```
SELECT *
FROM Auftrag LEFT JOIN Personal
ON Auftrag.Persnr = Personal.Persnr ;
```

Im Gegensatz zum *„INNER JOIN"* wird die Ausgangstabelle, also die Relation „Auftrag" nicht verändert oder gekürzt, falls Attribute von den Relationen „Auftrag" und „Personal" einander nicht gleichen. Die leeren Attribute der Relation werden mit *„NULL"* befüllt.

2.6 Marktübliche verfügbare Analysetools

Am Markt gibt es unzählige Arten von Analysetools, die je nach Anwendungsfall unterschiedliche Ziele verfolgen. Nachfolgend werden die hilfreichsten Analysetools untersucht und verglichen. Diese sollen auf ihre Anwendbarkeit auf die Anforderungen von automotiver Software geprüft werden.

2.6.1 KPIFire [32]

Dieses Tool ist im Internet ausführbar und eher im Managementbereich angesiedelt. Es ist eine Vielzahl von vorgefertigten Key Performance Indikatoren verfügbar, wie zum Beispiel für die Berechnung von Einnahmen, Kosten, Gewinn oder Kundenzufriedenheit über einen bestimmten Zeitraum. Nach Anlegen eines Benutzer-Accounts lassen sich beliebig viele Personen hinzufügen, die am Projekt oder Ziel mitarbeiten. KPIFire beinhaltet drei Ebenen, die miteinander verknüpft werden können:

- Ziel: Dieses wird nach ihrer Zieldefinition mit vorhandenen oder neu erstellbaren Projekten oder Metriken verknüpft. Der aktuelle prozentuale Erfüllungsgrad des Ziels hängt von den verbundenen Projekten oder Metriken ab und wird in Form von Tortendiagrammen angezeigt.

- Projekt: Nach dem Hinzufügen eines Projekts müssen Projektbeteiligte und ein Workflow in Form von Aufgaben eingegeben werden. Diese Tasks beinhalten die Beschreibung der Aufgabe, betroffene Personen, die Priorität sowie den aktuellen Status (*To Do*, *Doing*, *Done*). Diese Aufgaben wirken sich je nach Erfüllungsgrad auf den Projektstatus aus, der wie bei den Zielen in Form von Tortendiagrammen angezeigt wird. Des Weiteren lassen sich unverbindlich Dateien, verknüpfte Ziele oder Metriken anhängen. Je nach Zustand der Aufgaben ändert sich der prozentuale Erfüllungsgrad des Projekts.

- Metrik: Die Erstellung einer Metrik erfordert einerseits einen Namen sowie Input-Daten. Diese sind in KPIFire nach einem gewissen Schema direkt in die Applikation einzugeben oder können von Microsoft Excel [27] Sheets oder CSV-Dateien importiert werden. Die Input-Daten müssen monatlich einen aktuellen Wert sowie einen Zielwert haben, sei es in Form einer absoluten Zahl, prozentuell oder einem monetären Wert. Abweichungen von diesem Schema oder gar die Möglichkeit einer freien Gestaltung einer Metrik sind nicht erlaubt.

Die erstellten Ziele, Projekte und Metriken sind in eigenen Dashboards übersichtlich visualisiert. Die bestehenden Verknüpfungen, die sich am unteren Rand jedes Projektes befinden, können ebenfalls eingesehen werden, siehe Abbildung 2.17. Wie in Abbildung 2.18 ersichtlich, wirkt sich der Projektstatus der Beispielprojekte 0 bis 2 direkt auf den Erfüllungsgrad des definierten Zieles aus.

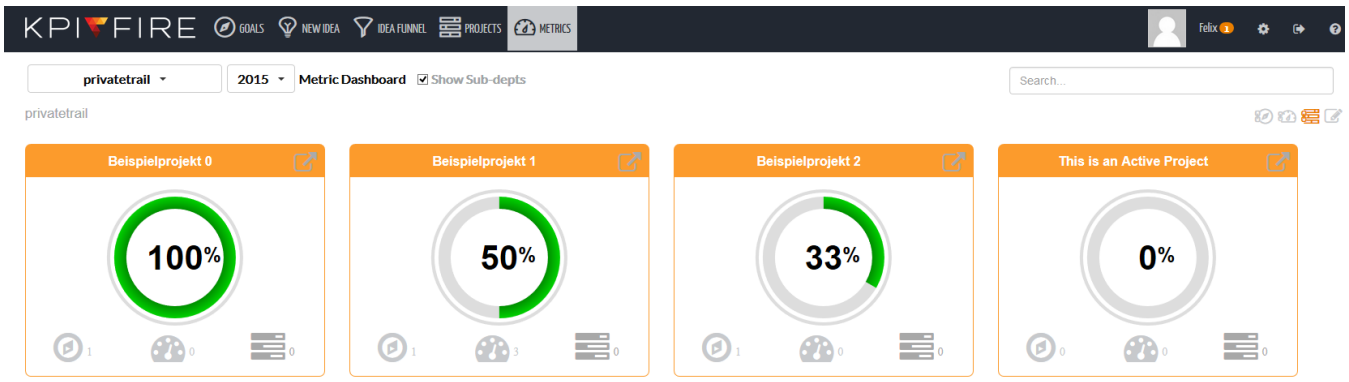


Abbildung 2.17: KPIFire, Dashboard Projekte, [32]

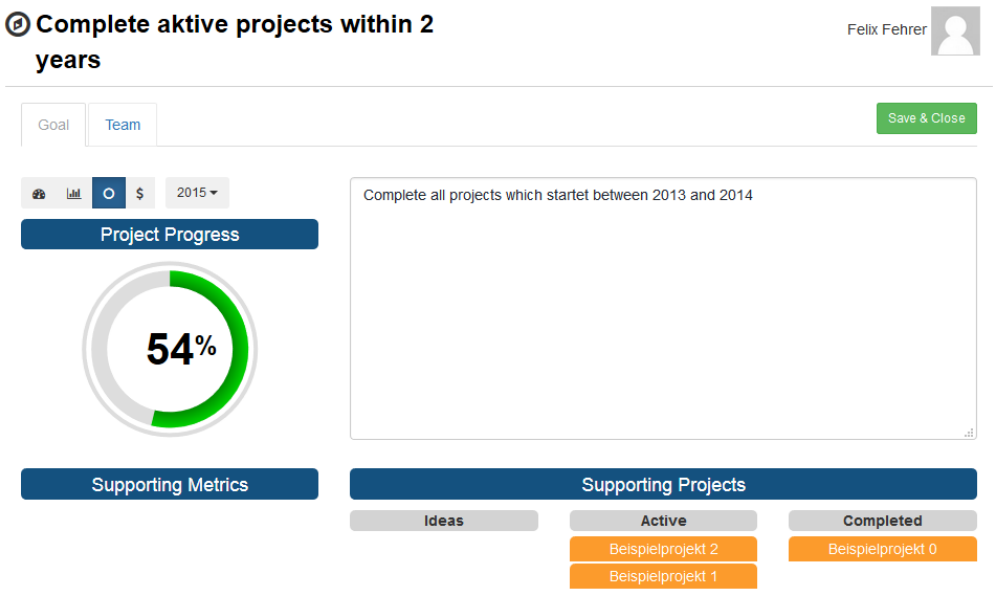


Abbildung 2.18: KPIFire, Dashboard Ziele, [32]

Zusammenfassend sei erwähnt, dass dieses Tool großen Wert auf Übersichtlichkeit, das optische Erscheinungsbild sowie Verständlichkeit legt. Letzteres führt jedoch dazu, dass dieses Tool sehr schwer an eine vorgegebene Projektablaufstruktur anpassbar ist. Es bietet keinerlei Möglichkeit der freien Gestaltung der Metriken. Lediglich eine Art monatlicher „Soll-Ist"-Vergleich -wie oben beschrieben- ist möglich, was der Anforderung der Analyse automotiver Software nicht entsprechen würde. Hervorzuheben ist der visuelle Aufbau der Dashboards, mit denen es möglich ist, sich innerhalb kürzester Zeit einen Überblick über den Projektfortschritt zu schaffen.

2.6.2 SimpleKPI [33]

Dieses, wie der Name schon besagt, einfache Online-Tool SimpleKPI erlaubt es dem Benutzer, recht schnell und unkompliziert Key Performance Indikatoren zu erstellen und zu visualisieren. Ebenen -wie im oben beschriebenen Analysetool- sind nicht implementiert, sehr wohl aber die Visualisierung in Form von Dashboards und die Möglichkeit mehrere Benutzer hinzuzufügen. Die Erstellung eines KPIs erfordert nach Namensgebung, Eingabeintervall (täglich, wöchentlich, monatlich) folgendes Schema der Inputdaten, wie in Tabelle 2.2 ersichtlich.

Tabelle 2.2: SimpleKPI, Inputdaten [33]

Analytics - Default View					
Actual Project Status					
Group 1	**User**	**Entry Date**	**Actual**	**Target**	**Notes**
(Not Set)	Felix Fehrer	16.12.2015	90	100	
(Not Set)	Felix Fehrer	15.12.2015	82	100	
(Not Set)	Felix Fehrer	14.12.2015	80	100	
(Not Set)	Felix Fehrer	13.12.2015	80	100	
(Not Set)	Felix Fehrer	12.12.2015	70	100	

Die Input-Daten können auch von Microsoft Excel importiert werden. Das simple Schema ist ähnlich zu jenem von KPIFire aufgebaut und beinhaltet unter anderem den täglich aktuellen Wert (*Actual*) und den Zielwert (*Target*).

Mit diesen Informationen können die Graphen erstellt werden. SimpleKPI bietet eine Vielzahl von Graphen für einen KPI an, die individuell anpassbar sind und im Dashboard zusammengefasst werden können, siehe Abbildung 2.19.

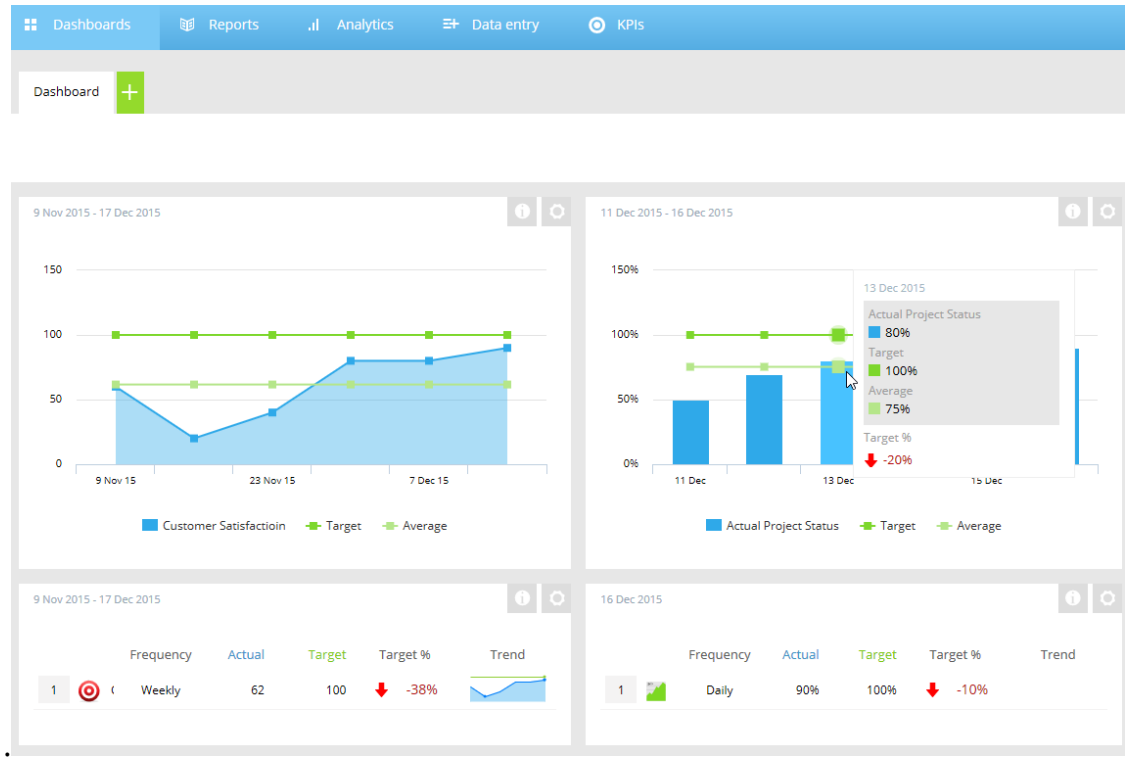


Abbildung 2.19: Simple KPI Dashboard, [33]

Eine individuelle Anpassung der Inputdaten (*Actual*, *Target*) ist nicht möglich, was die freie Gestaltung der KPIs erheblich einschränkt. Für detaillierte Fragestellungen im Projektmanagementbereich ist das Heranziehen dieser Applikation nicht geeignet, da teilweise sehr maßgeschneiderte Key Performance Indikatoren benötigt werden, um den Projektverlauf analysieren zu können. Die vorliegende vielschichtige Datenstruktur mit den Entwicklungs- und Testdaten des Industriepartners können weder in das Analysetool importiert, noch in sinnvoller Form bearbeitet werden. Für komplexe Systeme wie den Entwicklungsprozess des Industriepartners kann diese Software daher nicht angewendet werden.

Betont seien jedoch die leichte Bedienbarkeit und der sehr vielfältig gestaltbare Aufbau des Dashboards. Mit dieser Applikation lassen sich kurz und prägnant aussagekräftige Dashboards erstellen und als .pdf-Datei für etwaige Quartalsberichte an die verantwortlichen Personen exportieren.

2.6.3 JIRA [34]

JIRA ist ein Projektmanagement-Tool, das stark auf die Bedürfnisse eines Projektteams bei der Entwicklung von umfangreichen Softwareprojekten eingeht. Es ist jedoch ebenso möglich neben Softwareprojekten auch Projekte, die das Management betreffen, zu organisieren. Beim Erstellen von Softwareprojekten muss der Benutzer unterschiedliche Vorgehensmodelle

auswählen wie SCRUM, Kanban oder „Basic Software Development“, ein vereinfachtes inkrementelles Modell. Für jedes dieser Modelle gilt es sogenannte *Issues*, also Arbeitspakete zu definieren. Diese müssen nach Type kategorisiert werden, wie zum Beispiel *Bug* oder *Task*. Des Weiteren sind auch die Typen *Epic* und *Story* anwählbar, die einen größeren Arbeitsumfang umfassen und mehrere *Tasks* oder *Bugs* beinhalten können. In einer Datenbank gespeicherte Arbeitspakete können als CSV-Datei importiert werden, jedoch ist es nicht möglich, vom vorgegebenen Schema abzuweichen, was die Anpassung einer bereits vorhandenen Datenstruktur in Projekten erheblich erschwert.

Die *Issues* werden in einem vorgefertigten oder selbst erstellbaren, jedoch nur sehr eingeschränkt bearbeitbaren Workflow abgehandelt. Die Grundbausteine sind *To Do* (blau), *In Progress* (gelb), *In Review* (gelb) und *Done* (grün). Ein modifiziertes Workflow ist in Abbildung 2.20 ersichtlich, wobei auffällt, dass die einzelnen Status obligatorischen Typen zugeordnet werden müssen. Somit kann ein *Task* nie in den Status *Failed* fallen, sondern muss den oben genannten Typen zugeordnet werden.

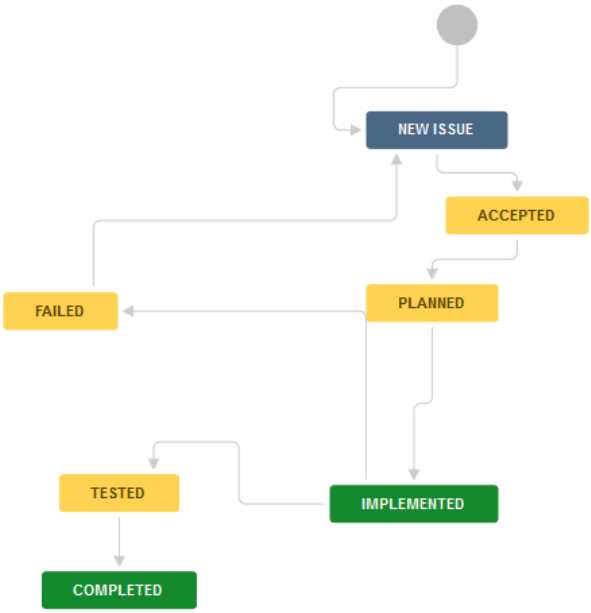


Abbildung 2.20: JIRA benutzerdefinierter Workflow, [34]

Wie in den Analysetools vorher, ist es in JIRA möglich, Dashboards zu erstellen, um den aktuellen Verlauf der Arbeitspakete übersichtlich darzustellen, siehe Abbildung 2.21. Die verschiedenen Torten- oder Liniendiagramme können mittels Add-On Paketen aus dem Internet heruntergeladen werden.

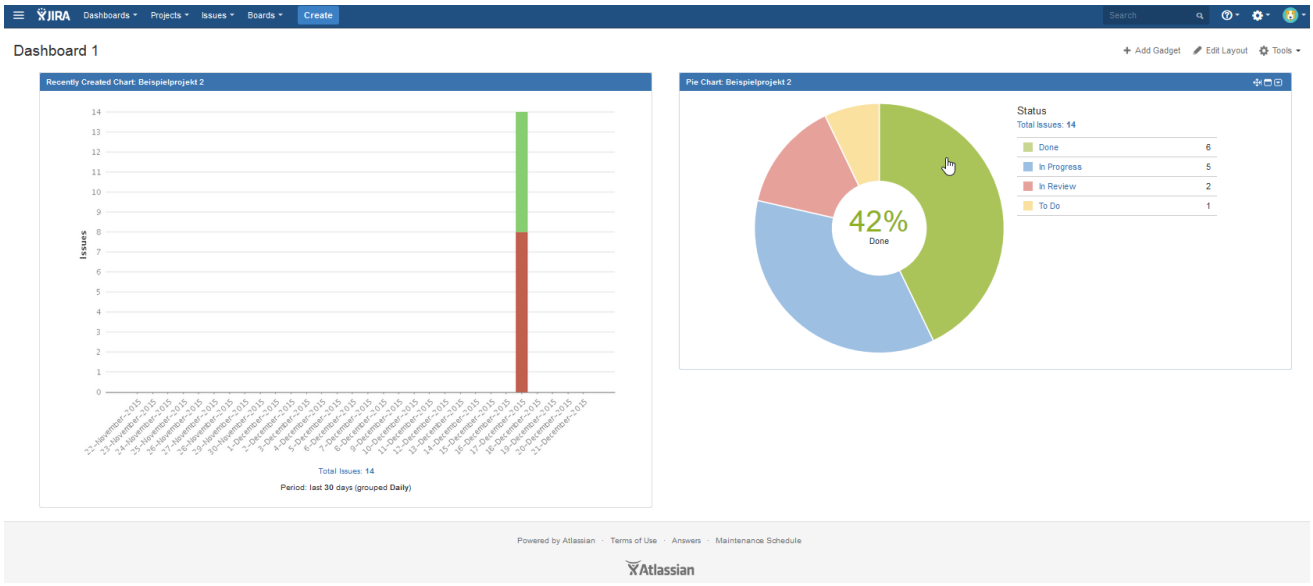


Abbildung 2.21: JIRA Dashboard, [34]

JIRA ist ein Projektmanagement Tool, mit dem es möglich ist, Vorgehensmodelle unterschiedlicher Projektstrukturen übersichtlich darzustellen. Verschiedene Vorgehensmodelle wie SCRUM oder Kanban werden unterstützt, jedoch nicht das in der Automobilindustrie verwendete V-Modell. Testfälle, die bei der Implementierung der Arbeitspakete oder Anforderungen entstehen, werden durch das Zusatzpaket JIRA Capture abgedeckt, bei dem man den *Issues* Testfälle zuordnen kann, [35]. Dieses Analysetool erlaubt es jedoch nicht, den komplexen Entwicklungsprozess des Industriepartners auf der Online-Plattform abzubilden. Der aktuelle Projektfortschritt richtet sich bei JIRA nach den insgesamt abgeschlossenen *Tasks* und Testfällen. In modernen Softwareprojekten gibt es jedoch Zwischenfreigaben im Projekt, die auch vor der endgültigen Systemfreigabe eingehalten werden müssen. Ein fortlaufender Soll-/Ist-Vergleich von Software Releases muss als KPI ebenfalls visualisiert werden, um noch vor Projektabschluss ausreichend Information über die Vitalität des gesamten Entwicklungsprojekts zu erhalten. Ein weiterer Kritikpunkt an diesem Programm ist dessen unzureichende Anpassbarkeit an die Anforderungen unterschiedlicher Abstraktionsebenen gemäß V-Modell, vgl. Kapitel 2.2. Bei einer Analyse des Entwicklungsprozesses muss sehr wohl auf die Heterogenität der Anforderungen bezüglich ihrer Ebenen eingegangen werden. Es wird bei JIRA nicht zwischen Modul- und Systemanforderungen unterschieden, was Schwierigkeiten bei ihrer Klassifikation und letztendlich in der Visualisierung der Projektphasen bereitet.

2.6.4 Datapine [36]

Diese Business Intelligence Software wurde im Rahmen eines Startup-Unternehmens erstellt und ist sehr universell einsetzbar. Für eine Analyse von Test- oder Entwicklungsdaten muss die Datenbank zuerst mit dem Online-Tool verknüpft werden. Dazu ist eine Verbindung zu einer lokalen Datenbank erforderlich, die entweder von einem gesicherten Server des Unternehmens importiert werden kann oder über eine Remoteverbindung mit der Analysesoftware verknüpft wird. Aktuelle Standards wie MySQL [37], Microsoft SQLServer [27] oder Oracle [38] werden akzeptiert und sind untereinander kombinierbar. Falls die Datengröße nicht allzu groß ist, können ebenfalls CSV-Files importiert werden, die im weiteren Verlauf mit anderen importierten Datenbanken verknüpft werden können. Ebenso wie in den oben beschriebenen Analysetools können bei Datapine mehrere Benutzer hinzugefügt werden und die erstellten KPIs oder Metriken in Dashboards visualisiert werden. Erwähnenswert ist die flexible

Datenstruktur, die das Programm erlaubt. Ein vorgegebenes Schema, wie bei KPIFire oder SimpleKPI ist hier nicht erforderlich und kann bei der Verbindung des Online-Tools mit der Datenbank festgelegt werden. Ein besonderes Merkmal dieses Tools ist die Möglichkeit der Eingabe von SQL-Abfragen an die verknüpften Datenbanken für die Erstellung der KPIs. Mit dem Ergebnis der Query können dann die Charts in Form von Balken-, Torten- oder Liniendiagrammen angezeigt werden. In Dashboards können schließlich die erstellten Diagramme angezeigt und als PDF-Datei exportiert werden, wie in Abbildung 2.22 ersichtlich.

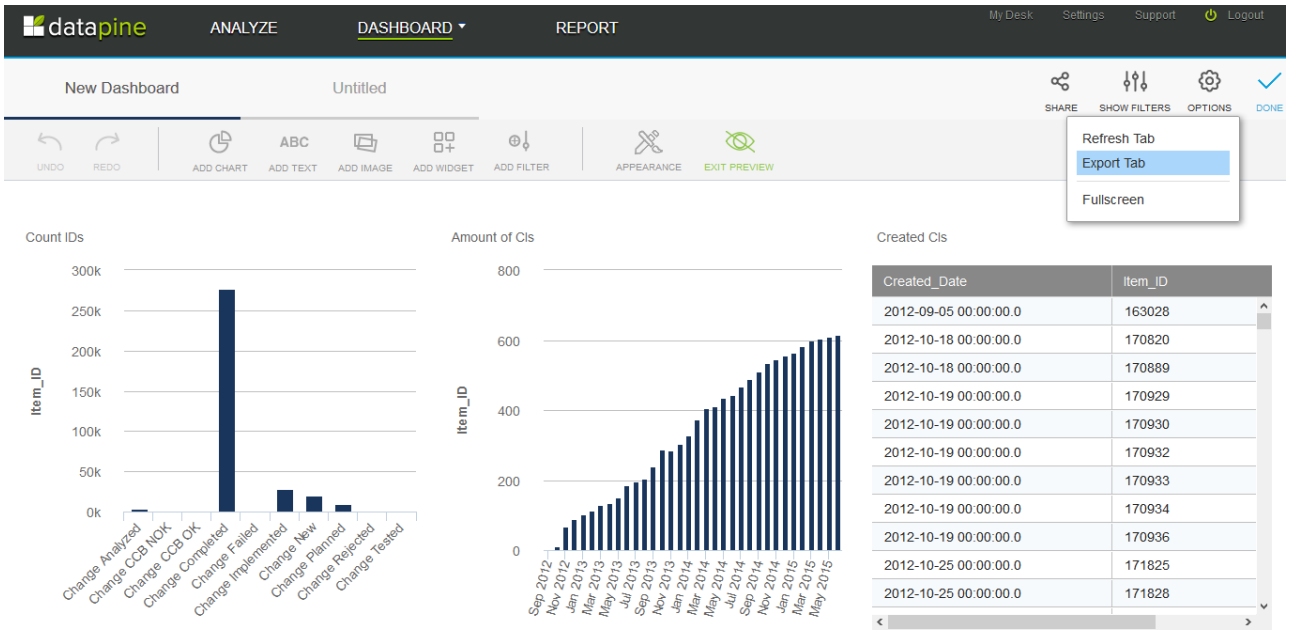


Abbildung 2.22: Datapine, Dashboard, [36]

In Abbildung 2.23 ist die Generierung einer Tabelle mit SQL Abfragen unter Verwendung einer Testdatenbank visualisiert.

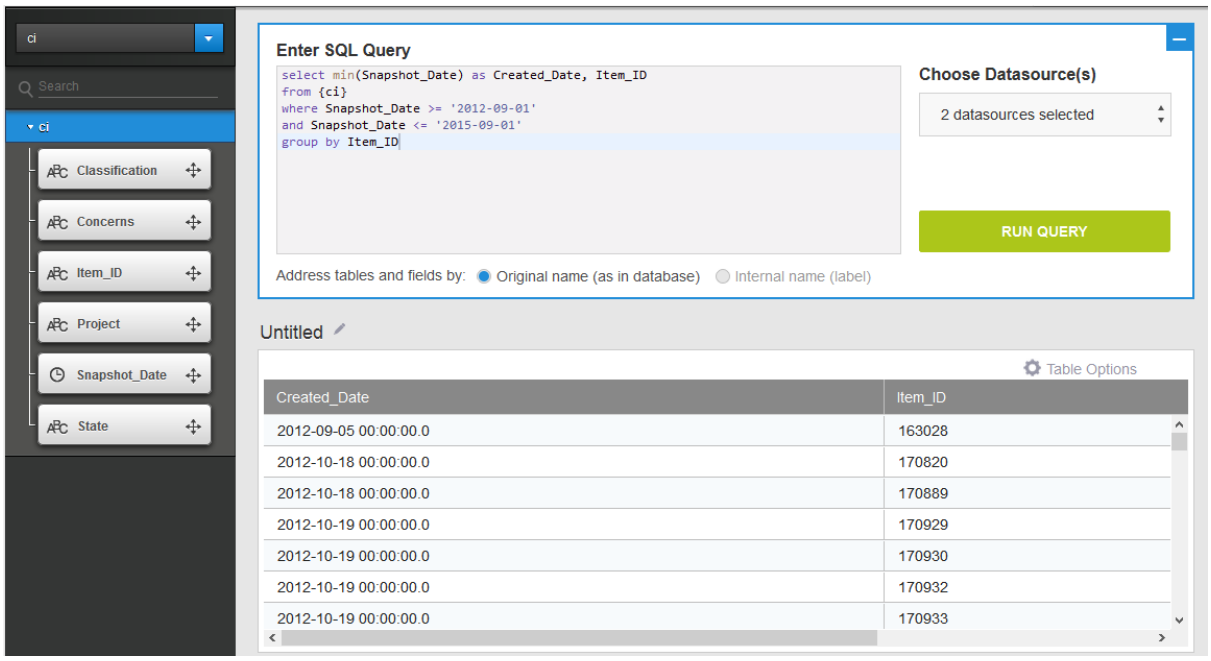


Abbildung 2.23: Datapine, SQL Abfrage und Ergebnis, [36]

Trotz der Möglichkeit, mehrere Datenbanken in eine SQL-Abfrage mit einzubinden, sind diese Abfragen in ihrer Komplexität limitiert. Bei der Untersuchung von Entwicklungs- und Testdaten mithilfe des Analysetools hat sich ergeben, dass beispielsweise Unterabfragen nach einer *FROM*-Klausel nicht zulässig sind. Auch *LEFT JOIN*-Befehle zur Verknüpfung mit anderen Tabellen konnten nicht in vollem Umfang ausgeführt werden. Diese Restriktionen sind aber für die Erstellung der KPIs mit SQL-Abfragen von erheblicher Bedeutung. Eine andere Möglichkeit der Tabellen- oder Diagrammerstellung ist mit einer „Drag and Drop"-Funktion gegeben. Hierbei werden die zu interessierenden Spalten der Datenbank der X- bzw. Y-Achse zugeordnet, wie in Abbildung 2.24 ersichtlich. In diesem Beispiel wird eine historische Datenbank in Form einer CSV-Datei herangezogen, um die Anzahl der Arbeitspakete über die Zeitachse zu visualisieren. Es fällt auf, dass diese Form des Diagramm-Aufbaus zwar einfach ist, aber erhebliche Einschränkungen bezüglich der Gestaltungsmöglichkeit mit sich bringt. Lediglich Zähl-(*COUNT*), Summen-(*SUM*) und Mittelwertfunktionen (*AVERAGE*) sind für die Darstellung auf der Ordinate möglich. Die Darstellung von relativen Häufigkeiten über die Zeit oder Softwarefreigaben sowie die Visualisierung von Workflows von Planungselementen, vgl. Abbildung 2.20, sind mit Datapine nicht möglich.

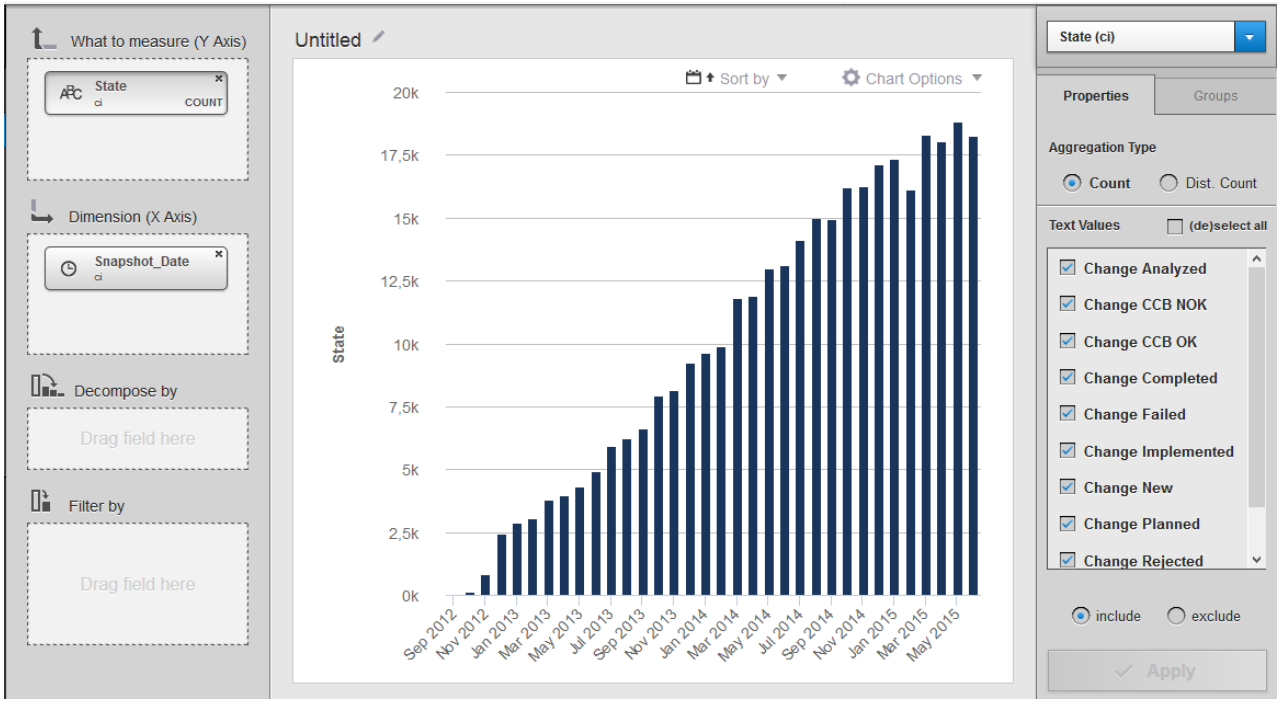


Abbildung 2.24: Datapine, „Drag and Drop"-Funktion [36]

Da sich dieses Analysetool stark an wirtschaftlichen Bedürfnissen ausrichtet, müsste die historisch aufgebaute Datenbank schon vor der Anbindung an das Online-Tool mit SQL-Queries bearbeitet werden, um eine ausreichende Entwicklungs- und Testdatenanalyse durchführen zu können.

2.6.5 Sisense [39]

Die BI-Software Sisense wurde 2010, nach einer langen Entwicklungszeit von zehn Jahren, für die Öffentlichkeit zugänglich gemacht und zu deren Kundenstamm zählen seither viele namhafte. Dieses Tool muss im Gegensatz zu den anderen Analyseprogrammen heruntergeladen werden und ist somit auch Offline verfügbar. Die zu analysierenden Daten können auf vielfältigster Weise importiert werden. Neben dem Datenimport über MySQL, Oracle oder Microsoft SQLServer ist auch eine lokale Datenanbindung über Microsoft Access-

Datenbanken, Microsoft Excel-Dateien oder CSV-Dateien möglich. Dadurch, dass das Tool offline verfügbar ist, müssen die lokalen Datenbanken nicht auf Online-Servern hochgeladen werden, sondern lediglich verknüpft werden. Sisense bietet dazu ein eigenes Interface für die Anbindung an und nennt sich Sisense ElastiCube. Das Tool erlaubt dadurch den Import mehrdimensionaler Datenbanken (Würfelform). Mit diesem Interface können, nach erfolgreicher Datenbank-Verknüpfung, die Tabellen miteinander verbunden werden, siehe Abbildung 2.25.

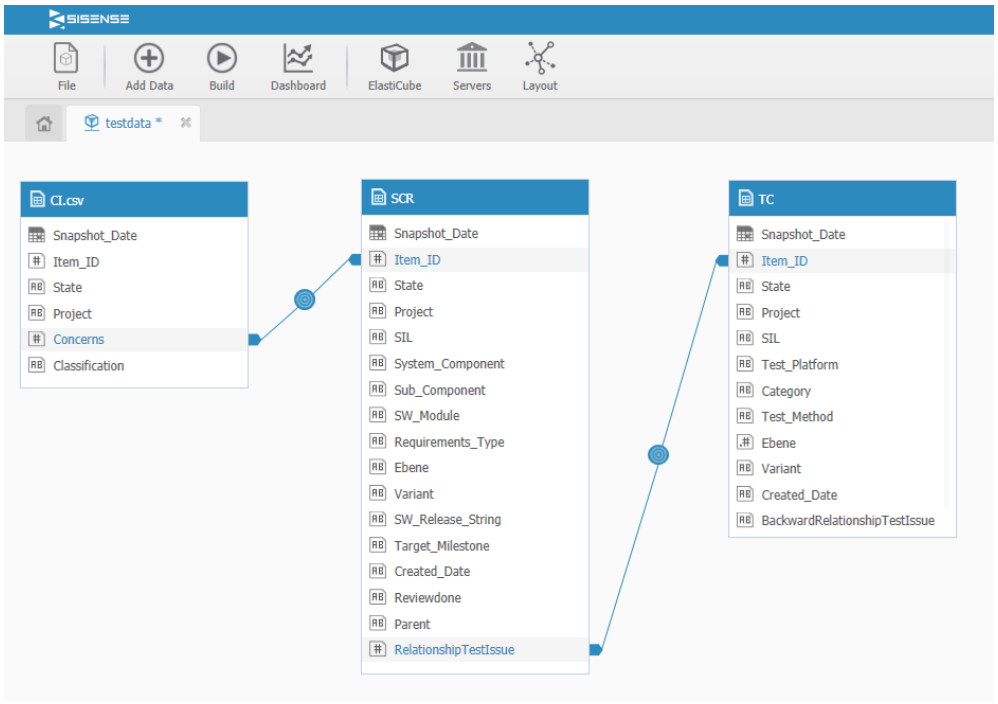


Abbildung 2.25: Sisense, ElastiCube, [39]

Nach dem Erfolgreichen Aufbau der Datenbank können nun in einem Dashboard Diagramme unterschiedlichster Art erstellt werden. Dazu wird der erstellte „ElastiCube“ ausgewählt und der Diagrammaufbau kann erfolgen. Beispiele erstellter Grafiken sind in Abbildung 2.26 ersichtlich. Zur Analyse wurden Beispielprojekte mit deren Anforderungen, Testfällen und Arbeitspaketen mit dem Programm verknüpft und Diagramme im Dashboard erstellt. Das erste Diagramm illustriert die Anzahl täglich vorkommender Arbeitspakete über einen bestimmten Betrachtungszeitraum. Chart Nummer zwei (orange) und drei (hellblau) veranschaulichen die erstellten Anforderungen bzw. Testfälle. Diese werden zur besseren Übersicht monatlich gruppiert.

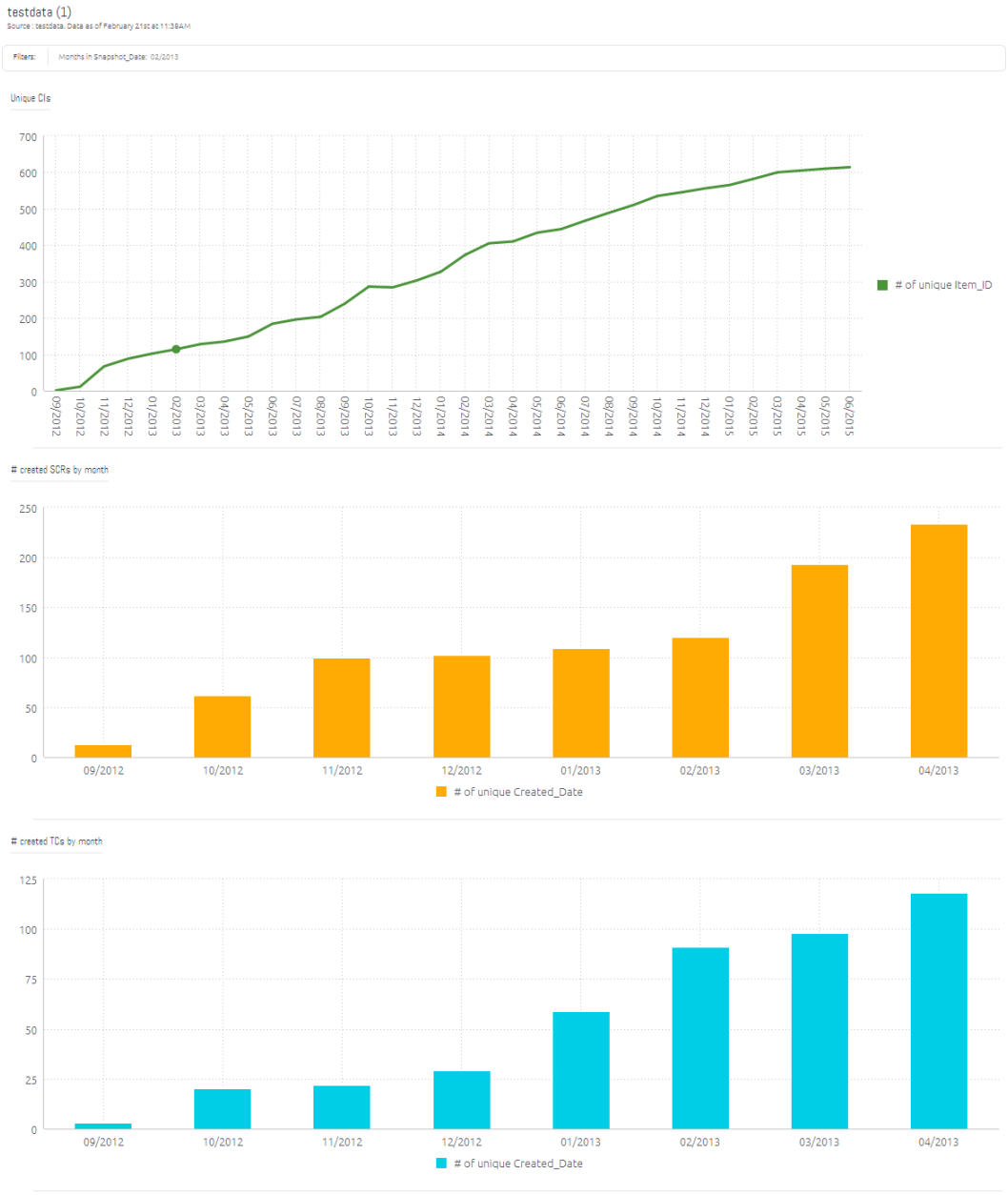


Abbildung 2.26: Mit Sisense erstellte Grafiken, [39]

Sisense bietet sehr viele Möglichkeiten des Datenimports und legt Wert auf eine übersichtliche Darstellung der Grafiken in Dashboard, wie in den bisher untersuchen Tools. Jedoch ergaben sich im Laufe der Untersuchung folgende Einschränkungen bei der Erstellung des Reports:

- Eine Verknüpfung der *CI*-Datenbank mit der *SCR*-Datenbank über *Concerns* ist nicht möglich, da die Spalte *Concerns* nicht der ersten Normalform entspricht. Es müssten temporäre Tabellen erstellt werden, dass das Programm allerdings nicht erlaubt.

- Die Anzeige des Workflows der Arbeitspakete über ein Zeitintervall war nicht umsetzbar, weil auf der Ordinate nur Werte des Datentyps Integer aufgetragen werden dürfen. Ähnlich zum Analysetool Datapine können mit den Befehlen *SUM*, Item-IDs aufsummiert werden, mit *AVERAGE* der Mittelwert berechnet werden bzw. mit *COUNT unique* unterschiedliche Item-IDs gezählt werden.

- Die Darstellung relativer Häufigkeiten ist wegen der im oberen Punkt genannten Restriktionen ebenfalls nicht erlaubt. Es müssten temporäre Tabellen erstellt werden, welche die relativen Häufigkeiten vorab berechnen. Ein SOLL-/IST-Vergleich von abgeschlossenen zu nicht abgeschlossenen Planungspaketen ist demnach nicht umsetzbar.

- Aufgrund der Tatsache, dass die vorliegenden Datenbanken über eine Verknüpfungslogik miteinander verbunden sind, müssen Planungselemente in Diagrammen vereint werden, um diese systematisch vergleichen zu können. Bei den letzten beiden Charts in Abbildung 2.26 bietet es sich an, die Balken von Anforderungen und Testfällen zu gruppierten Balken in ein Diagramm zu vereinen. Da die Tabellen in ElastiCube nicht über die vorgegebene Verknüpfungslogik verbunden werden konnten, ließen sich die Anforderungen und Testfälle auch nicht zu einem Diagramm zusammenfassen.

Für sehr einfache Analysen des Entwicklungsprozesses ist dieses Tool durchaus brauchbar, aber die mit dem Industriepartner vereinbarten Fragestellungen konnten durch Sisense nicht beantwortet werden. Die umzusetzenden KPIs konnten nicht erstellt werden, da der Diagrammaufbau zu unflexibel ist und nicht mit SQL-Abfragen auf die Datenbank zugegriffen werden kann.

2.6.6 CA Technologies (Rally) [40]

Das Projektmanagement-Tool Rally wurde erst kürzlich von CA Technologies übernommen und bietet eine Vielzahl an Möglichkeiten, den Ablauf von Projekten zu dokumentieren bzw. zu beobachten. Jedoch konnte die Applikation mit den Entwicklungs- und Testdaten nicht verbunden werden. Es steht allerdings ein Beispielprojekt zur Verfügung, mit dem das Tool untersucht werden kann. Dabei wird ein Projekt auf unterschiedliche Interessensgruppen aufgeteilt. So befasst sich beispielweise ein Team mit der Softwareentwicklung, während sich ein anderes Team auf die Hardwareentwicklung fokussiert. Den Teams stehen verschiedene Vorgehensbausteine zur Projektabwicklung zur Verfügung, ähnlich dem V-Modell. Die Bausteine teilen sich dabei wie folgt auf:

- Die *User-Stories* beinhalten Anforderungen an das Produkt und werden von einem Projektmitarbeiter definiert. Um eine *User-Story* abzuschließen muss diese in einem vorgegebenen Workflow bearbeitet werden. Der Workflow teilt sich in fünf Status auf (*Idea → Defined → In-Progress → Completed → Accepted*).

- An den *User-Stories* können *Tasks, Deflects* und *Test Cases* angehängt werden. Ein Task ist ein Arbeitspaket, das abgearbeitet werden muss, um die *User-Story* erfolgreich abzuschließen. *Deflects* sind Fehler, die im Laufe der Produktentwicklung entstehen und behoben werden müssen. Abschließend wird die Anforderung mit *Test Cases* getestet. Die drei Planungselemente folgen dabei einem ähnlichen Workflow, wie jenem der *User-Story*.

- Alle Planungselemente werden einem *Release* zugeordnet, der eine bestimmte Deadline besitzt. Bis zu diesem Datum sollen alle *User-Stories* mit deren *Tasks, Deflects* und *Test Cases* abgeschlossen sein. Dabei wird ein Release in eine bestimmte Anzahl von Iterationen unterteilt. Eine Iteration ist ein vordefinierter Zeitraum und umfasst meist ein bis zwei Wochen. Ein Beispiel dieser Vorgehensweise ist Abbildung 2.27 ersichtlich. Hierfür ist für die Iteration 6 zum *Release 2 (R2)* eine Vielzahl von *User-Stories (US), Tasks (TA)* und *Test Cases (TC)* abzuarbeiten.

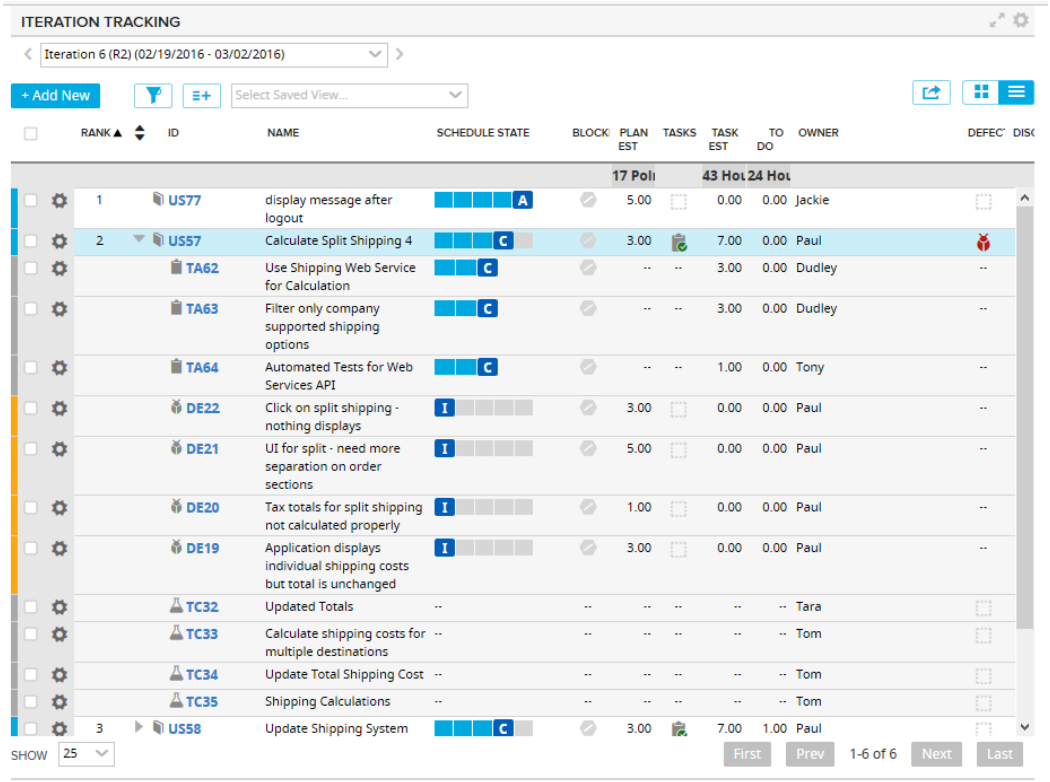


Abbildung 2.27: CA Technologies [40], Iteration-Tracking

Schlussendlich können die Planungselemente für einen bestimmten *Release* in Grafiken visualisiert werden, vergleiche Abbildung 2.28. Es werden die Planungselemente nach deren Status beurteilt und die Vitalität des *Releases* beurteilt. In diesem Beispiel ist die Vitalität des *Release 2* der Iterationen 5, 6 und 7 kritisch, da viele Planungselemente noch nicht abgeschlossen wurden.

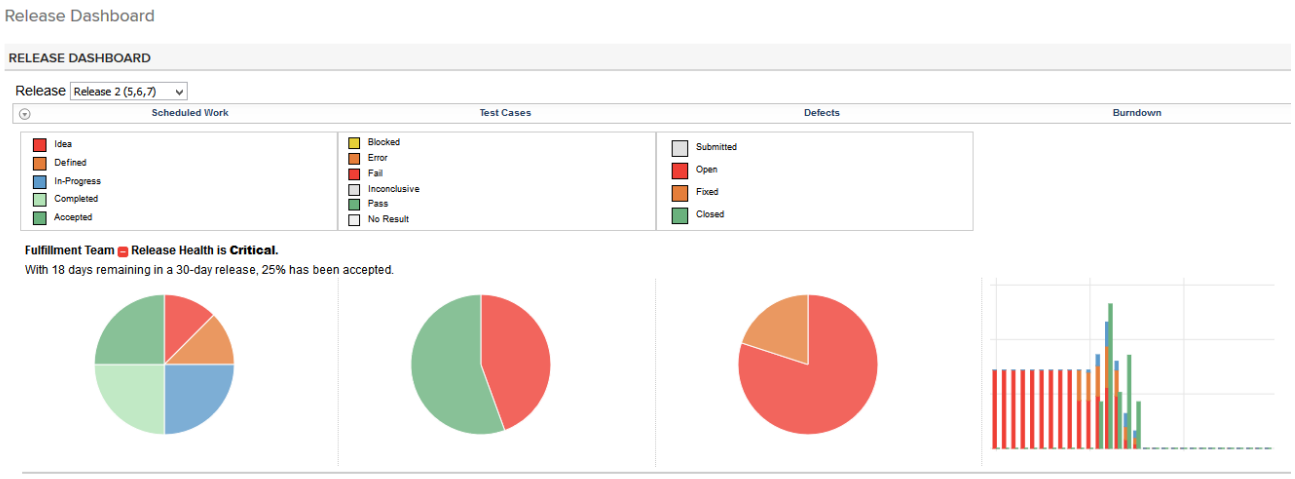


Abbildung 2.28: CA Technologies [40], Release Dashboard

Die Vorgehensweisen, die dieses Analysetool bietet, ähneln jenem des Industriepartners. Jedoch wurde keine Möglichkeit gefunden, die Entwicklungs- und Testdaten zu importieren bzw. mit SQL-Befehlen abzufragen. Die Datenstruktur mit den Workflows und Planungselementen ist vorgegeben und kann nicht angepasst werden. Auch der Diagrammaufbau ist sehr unflexibel, da viele Parameter bereits vorgegeben sind und nicht mehr

abänderbar sind. Des Weiteren konnte kein KPI annähernd nachgebaut werden und somit ist dieses Projektmanagement-Tool ungeeignet für eine Analyse mit Entwicklungs- und Testdaten.

2.6.7 InetSoft [41]

Inetsoft wurde im Jahr 1996 gegründet und ist ein Business Intelligence-Tool. Nach der Installation am Rechner können Datenbank vieler Hersteller mit der Applikation verbunden werden oder kleinere Tabellen im Microsoft Excel-Format oder CSV-Format importiert werden. Die Benutzeroberfläche ist sehr vielseitig gestaltbar. Neben Diagrammen und Tabellen können auch Kontrollkästchen, wie eine *Checkbox* oder *Radiobuttons*, eingefügt werden, die dann mit dem Diagramm interagieren. Wie schon bei den oben beschriebenen Tools erfolgt auch hier die Präsentation der Daten in Dashboards. Durch die „Drag & Drop"-Funktion werden die Achsen des Diagramms mit den importierten Daten gefüllt und bei Bedarf mit den Kontrollkästchen verlinkt. Das Ergebnis kann danach als Microsoft Excel-, Microsoft Powerpoint [27]- oder PDF-Datei exportiert werden. Ein Beispiel für ein – mit Inetsoft erstelltes – Dashboard ist in Abbildung 2.29 ersichtlich. Erwähnenswert ist die dynamische Veränderung der drei Diagramme, wenn im Kalender das Datum verändert wird. Ein Schiebebalken am unteren Rand filtert zusätzlich nach *Product Price* und interagiert ebenfalls mit den Diagrammen.

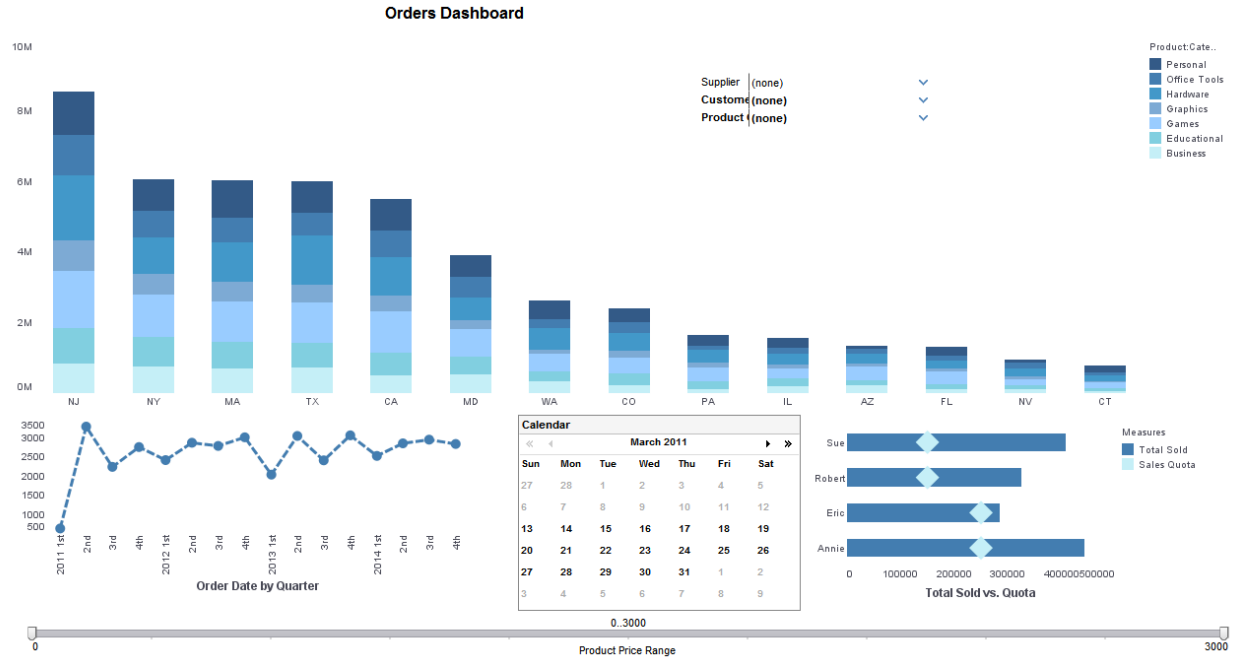


Abbildung 2.29: Inetsoft, Dashboard-Beispiel [41]

Für die Anwendbarkeit des BI-Tools auf den Entwicklungsprozess des Industriepartners wurden modifizierte Entwicklungs- und Testdaten eines Beispielprojekts im CSV-Format importiert und Dashboards erstellt. Aussagekräftige Diagramme können schnell und einfach erstellt werden und ein maßgeschneiderter Report generiert werden. Dabei wurde versucht, einzelne KPIs aufzubauen, die mit dem Industriepartner vereinbart wurden. Die KPIs konnten jedoch im Zuge der Testphase nur sehr eingeschränkt im Tool implementiert werden, da es keine Möglichkeit gibt die Daten mit SQL-Befehlen zu bearbeiten. Ein Beispiel eines modifizierten KPI liefert Abbildung 2.30. Es wurde versucht, Arbeitspakete mit deren Workflow über einen bestimmten Zeitraum zu veranschaulichen. Der Versuch konnte nur eingeschränkt umgesetzt werden, da Datentypänderungen nicht zulässig sind und die

Verknüpfung mehrerer Tabellen (zum Beispiel die Anbindung von Meilensteintabellen) ebenfalls nicht möglich war. Weitere Einschränkungen ergaben sich hinsichtlich Rechenzeit beim Chartaufbau und die fehlende Zoomfunktion, die für diese Tabelle unerlässlich ist.

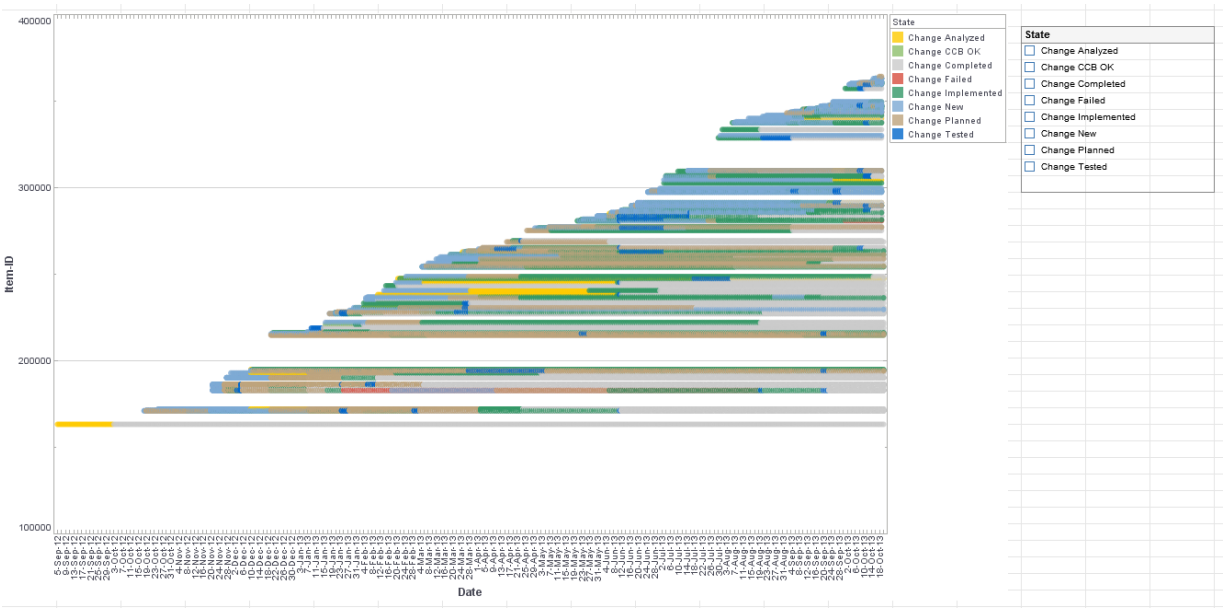


Abbildung 2.30: Inetsoft, Beispiel eines KPIs, [41]

Inetsoft zeichnet sich durch den flexiblen Aufbau von Charts und durch die vielfältigen Möglichkeiten der Datenbankanbindung aus. Bei keinem anderen Analysetool konnten einzelne KPIs so detailgetreu nachgebildet werden. Es war nicht möglich Berechnungen wie Prognosen oder Zeitverzüge durchzuführen, welche für die Beurteilung des Entwicklungsprozesses von erheblicher Bedeutung sind. Die Entwicklungs- und Testdaten müssten vor einem Datenimport mit SQL-Abfragen bearbeitet werden, um eine Analyse unter Zuhilfenahme des Tools durchführen zu können. Daher kann Inetsoft für eine Entwicklungs- und Testdatenanalyse nicht herangezogen werden.

2.6.8 Resümee der Marktanalyse

Alle vorgestellten Analysetools legen großen Wert auf die Benutzerfreundlichkeit und Darstellungsmöglichkeiten der Kennzahlen. Jedes der sieben untersuchten Applikationen hat anpassbare Dashboards implementiert, die als PDF oder Bild exportierbar sind. Unterschiede ergeben sich in deren Grundidee, welche Zielgruppen angesprochen werden sollen. So decken JIRA und CA Technologies mit deren integrierten Vorgehensmodellen vorrangig Bedürfnisse seitens des Projektmanagements ab, während KPIFire, Sisense oder Datapine Fragen seitens des operativen Managements beantworten sollen. Ein „ideales“ Analysetool konnte im Zuge der Marktanalyse nicht gefunden werden, da es sehr detailliert ausformulierte Fragestellungen an den Entwicklungsprozess zu beantworten gilt. Dennoch konnte durch systematisches Testen der (Online)-Tools ein guter Überblick über die aktuelle Marktsituation geschaffen werden und wertvolle Erkenntnisse für die Implementierung des eigenen Analysetools gewonnen werden.

3 Ausgangssituation und Grundlagen des Entwicklungsprozesses beim Industriepartner

Der Entwicklungsprozess des Industriepartners richtet sich nach dem in der Automobilbranche üblichen AUTOSAR-Standard, siehe Abbildung 2.3. Die Analyse des Entwicklungsprozesses bedarf der Kenntnis der Vorgehensweise und Datenstruktur von verschiedenen Projekten. Hierfür wird ein modifiziertes V-Modell herangezogen, welche die Anforderungen bzw. die zugehörigen Testfälle in Ebenen einteilt. Die oberste Ebene definiert die *System Requirements*, bei denen die Anforderungen auf Systemebene spezifiziert werden. Die Gesamtsoftwareebene umfasst den höchsten softwarebezogenen Abstraktionsgrad und wird mit *System Component Requirements (SCRs)* gekennzeichnet. Die Integration und Kommunikation der untersten und somit detailreichsten Schicht – den *Module Requirements (MRs)* – erfolgt auf der Integrationsebene. Die einzelnen Ebenen werden – wie beim V-Modell üblich – mit Tests validiert bzw. verifiziert, den sogenannten *Test Cases (TCs)*. Für die Analyse der Entwicklungs- und Testdaten erfolgt eine Untersuchung der

- Anforderungen (*System Component Requirements SCRs* & *Module Requirements MRs)*,
- Testfälle (*Test Cases TCs*),
- Arbeitspakete, Fehler und Änderungen (*Change Issues CIs*)

und eine weitere Betrachtung der *System Requirements* ist nicht erforderlich. Der Vollständigkeit halber werden sie in Abbildung 3.1 angeführt. Anforderungen und Testfälle enthalten Arbeitspakete, die infolge von Änderungen oder Fehlern auftreten. Die Zusammenhänge und Verknüpfungen der Planungselemente sind in Abbildung 3.3 beschrieben, [25].

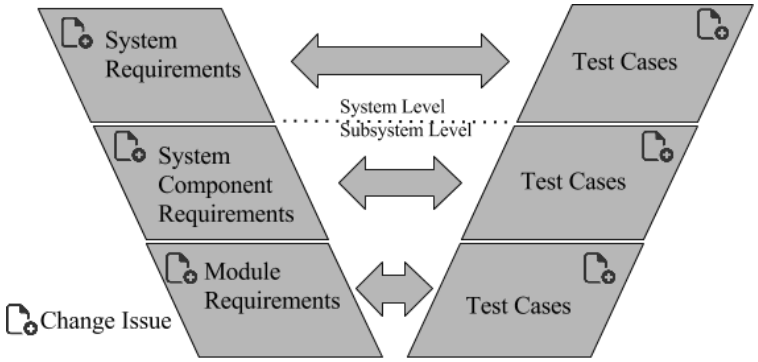


Abbildung 3.1: Modifiziertes V-Modell, in Anlehnung an, [42]

Die Entwicklungs- und Testdaten werden in dem Application Lifecycle Management (ALM) und Systems-Engineering Programm PTC MKS eingetragen und verwaltet. Dieses beinhaltet den Lebenszyklus eines Projektes und gliedert sich in die Phasen

- Anforderungsspezifikation,
- Entwicklung,
- Testen,
- Inbetriebnahme,
- Instandhaltung, [43].

Die Anwender können somit alle Planungselemente in einem Programm verwenden. Jedes Planungselement (*CI, SCR, MR* oder *TC*) lässt sich eindeutig einem Projekt zuordnen und wird mit einem Status versehen. Im Projektverlauf können die Planungselemente ihren Status nach einem vorgegebenen Workflow verändern. Dieser Workflow wird in den Kapiteln 3.1 bis 3.3 beschrieben.

Abbildung 3.2: Application Lifecycle Management-Diagramm, [43]

Wie in Abbildung 3.3 ersichtlich, sind die Planungselemente miteinander verknüpft, um deren Zugehörigkeit untereinander eindeutig zu definieren. Die Verknüpfungslogik mit den Planungselementen und Transitionen *Concerns*, *Child* und *Relationship Test Issue* ist in Abbildung 3.3 ersichtlich. Hierfür müssen im MKS-System die Zugehörigkeiten folgender Maßen definiert werden:

- *Change Issues* sind über *Concerns* mit den *System Component Requirements* verbunden.
- *System Component Requirements* sind über *Child* mit den *Module Requirements* verbunden.
- *Change Issues, System Component Requirements* und *Module Requirements* sind über *RelationshipTestIssue* mit den *Test Cases* verbunden.

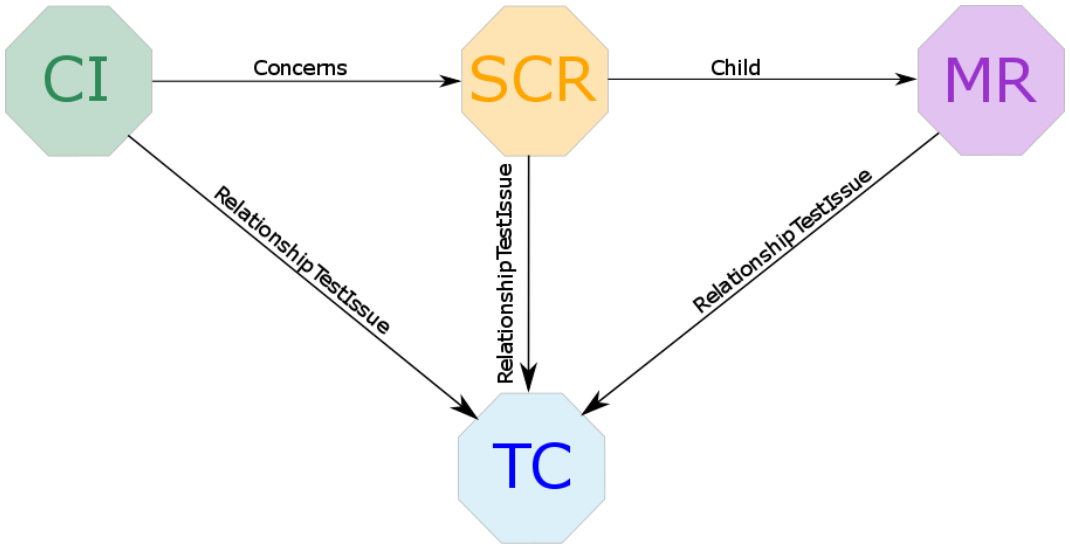


Abbildung 3.3: Verlinkung der Planungselemente, [6]

Nachdem sich im Entwicklungsprozess einige Spezifikationen für unterschiedliche Projekte nicht ändern, wurde eine sogenannte */ProductPlatform* erstellt. Diese Plattform verhält sich hierbei wie ein eigenständiges Projekt und enthält Anforderungen, Testfälle und Arbeitspakete mit derselben Datenstruktur wie vergleichbare Projekte. Der Unterschied liegt jedoch darin, dass diese Plattform für mehrere Projekte anwendbar ist. Um ein Projekt richtig zuordnen zu können, werden die Item-ID der Planungselemente mit einer Variante (*variant*) gekennzeichnet. Diese Variante verweist auf die ursprüngliche Projektzugehörigkeit, siehe Abbildung 3.4

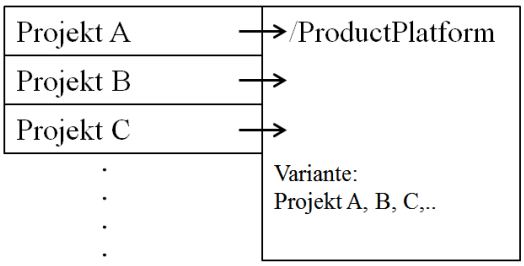


Abbildung 3.4: Aufbau der *Product Platform*

3.1 Anforderungen (*SCRs*/MRs)

Die Anforderungen gliedern sich in zwei Ebenen, zum einen auf die Systemebene (*SCRs*) und zum anderen auf die detailliertere Modulebene (*MRs*). Dabei lässt sich jedes *System Component Requirement* in feinere *Module Requirements* abstufen. Den eindeutigen Zusammenhang der Anforderungen bilden die *Child*-Einträge bei dem die betreffende *SCR*-Item-ID auf deren *MR* Item-ID verweist. Beide Anforderungen durchlaufen je nach Bearbeitungszustand bestimmte Status, die wie folgt definiert sind, [44]:

- *Requirement New,*
- *Requirement Specified,*
- *Requirement Implemented,*
- *Requirement Closed.*

Nach Erstellen einer Anforderung wird diese in den Status *Requirement New* gesetzt. Diese ist dadurch identifiziert, muss jedoch noch genauer beschrieben und analysiert werden. Ist dies geschehen, kann die verantwortliche Person die Anforderung in den Status *Requirement Specified* versetzen. Hierbei ist die Anforderung spezifiziert und der ASIL-Level definiert. Der Automotive Safety Integrity Level kennzeichnet das Risiko eines Ausfalls der zugeordneten Anforderung und ist bei der Entwicklung automotiver Software von relevanter Bedeutung. Demnach muss einer Anforderung des ASIL-Levels D mehr Beachtung beigemessen werden als jener in ASIL-A, vergleiche Kapitel 2.1.5. Nach Spezifikation und Definition des ASIL-Levels kann mit der Implementation und Verifikation begonnen werden. Dieser Status löst einen neuen Testfall (*TC*) und gegebenenfalls eine Modul-Anforderung aus. Ist die Anforderung verfügbar und im System implementiert worden, kann diese auf *Requirement Implemented* gesetzt werden. Wird die Anforderung nicht mehr benötigt, wird sie auf *Requirement Closed* gesetzt. Die möglichen Transitionen sind in Abbildung 3.5 ersichtlich, wobei eine Anforderung mehrmals denselben Status durchlaufen kann, also zirkuliert, [44], [45].

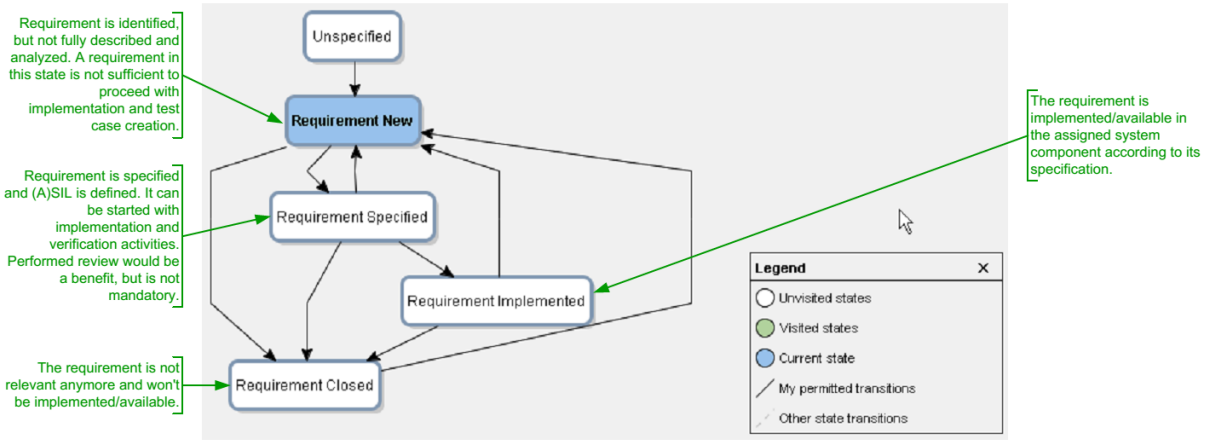


Abbildung 3.5: Workflow der *SCRs*, [44]

3.2 Testfälle (*TCs*)

Der rechte Ast des oben beschriebenen V-Modells erfordert eine Verifikation der Anforderungen. Diese werden mit Testfällen, den *Test Cases* (*TC*), verifiziert. Jede Anforderung auf System- oder Modulebene benötigt einen Testfall, der über *RelationshipTestissue* mit der Anforderung verbunden ist. Auch Testfälle durchlaufen – je nach Bearbeitungszustand – verschiedene Stadien, [44]:

- *TC New,*
- *TC Specified,*
- *TC In Work,*
- *TC Completed,*
- *TC Retest,*
- *TC Failed,*
- *TC Completed with Restriction,*
- *TC Closed.*

Die Verknüpfungen der Zustände sind in Abbildung 3.6 ersichtlich. Eine Spezifikation einer

Anforderung löst einen neuen Testfall aus, welcher mit *TC New* gestartet wird. Sind alle nötigen Informationen für den *Test Case* spezifiziert, wird dieser in *TC Specified* gesetzt. Im Falle eines Modultests wird erst mit der Bearbeitung (*TC In Work*) begonnen, wenn die Modulanforderung implementiert wurde, also den Status *Requirement Implemented* besitzt. Ein Systemkomponententest kann nach vorheriger Spezifikation sofort gestartet werden. Je nach Ergebnis der Testsession kann der Testfall in verschiedene Zustände (rück-)gesetzt werden. Ein erfolgreicher Test wird mit *TC Completed* gekennzeichnet. Der Test kann nach Bedarf noch einmal wiederholt werden und wird mit *TC Retest* klassifiziert. Ein fehlerhafter Test setzt ein Arbeitspaket (*Change Issue*) in Gang, welches die Klassifikation *Finding* trägt und den Fehler untersucht. Steht der Fehler beim Testen nicht direkt in Zusammenhang mit dem Testfall, so wird er in den Status *TC Completed With Restriction* versetzt und es ist möglich, den Testfall noch einmal zu wiederholen (*TC Retest*). Dabei gibt es die folgenden Möglichkeiten, diesen ohne Einschränkungen abzuschließen (*TC Completed*), zurück auf *TC Failed* oder auf *TC Closed* zu setzen. Wurde der Testfall auf *TC Closed* gesetzt, darf nicht mehr auf andere Zustände gewechselt werden. Des Weiteren darf jeder Status in *TC Closed* versetzt werden, falls der Testfall nicht mehr relevant ist, [25], [44], [45].

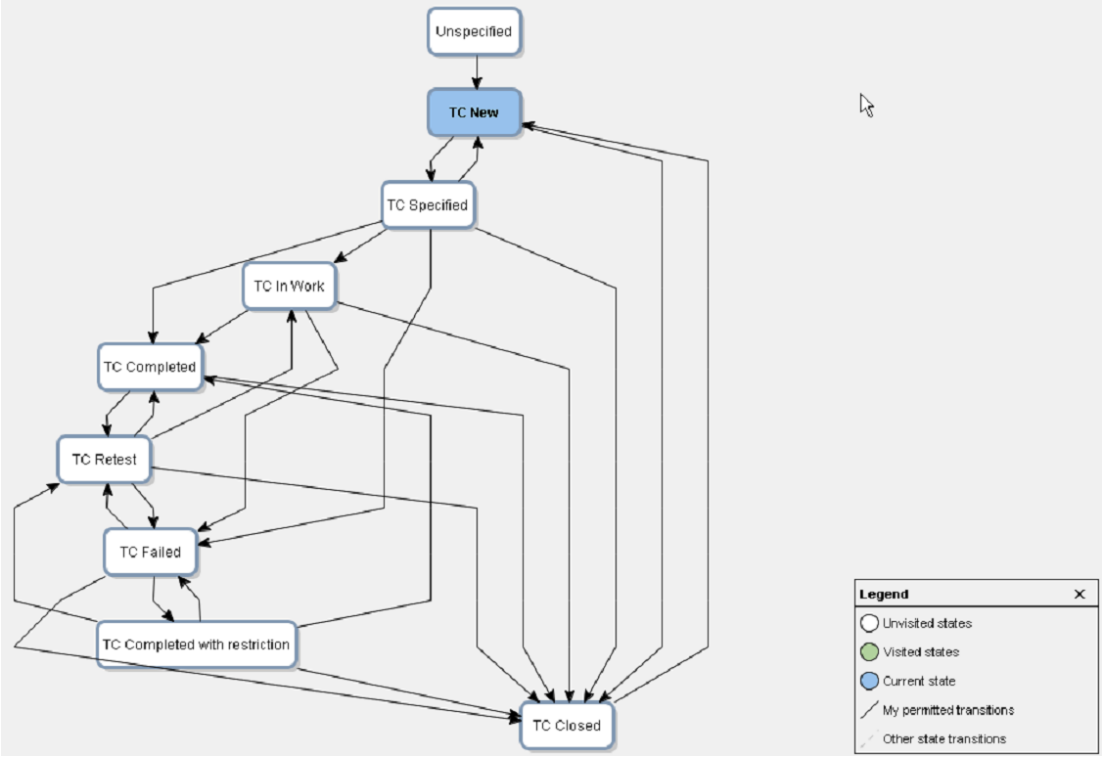


Abbildung 3.6: Workflow der *TCs*, [44]

3.3 Arbeitspakete (CIs)

Arbeitspakete entstehen infolge Änderungen, hervorgerufen durch Anforderungen (*SCRs* oder *MRs*) oder durch Fehler, die im Verlauf der Testphase auftreten. Abbildung 3.7 illustriert die Möglichkeiten Status der Arbeitspakete. Auch *Change Issues* durchlaufen verschiedene Zustände, [44]:

- *Change New*,
- *Change Analysed*,
- *Change CCB OK*,

- *Change CCB NOK,*
- *Change Rejected,*
- *Change Accepted,*
- *Change Planned,*
- *Change Implemented,*
- *Change Tested,*
- *Change Completed,*
- *Change Failed,*
- *Change Closed.*

Der Workflow der Arbeitspakete oder Fehler ist in Abbildung 3.7 ersichtlich. Eine geplante Änderung oder ein auftretender Fehler initiiert ein Arbeitspaket mit dem Status *Change New*. Nach der obligatorischen Analyse des Pakets wechselt es in den Status *Change Analysed*. Nun entscheidet ein *Change Control Board (CCB)* – je nach Komplexität des Arbeitspaketes – über dessen weitere Vorgehensweise. Wird der Fehler oder die Änderung nicht genehmigt, so wechselt das *CI* in den Status *Change CCB NOK* und muss noch einmal neu erstellt werden (*Change New*) oder der Fehler wird geschlossen (*Change Closed*). Eine Genehmigung des *CIs* vom *Control Board* befördert das Arbeitspaket in den Zustand *Change CCB OK* und das Paket kann weiterbearbeitet werden. Die Entscheidung des *Control Boards* beeinflusst die weitere Vorgehensweise zur Bearbeitung des Arbeitspaketes. Zum einen kann das *CI* in den Zustand *Change Rejected* gesetzt werden, falls die Änderung zurückgewiesen wurde, andererseits in den Zustand *Change Accepted* versetzt werden, falls das *CI* vom *Control Board* akzeptiert wurde. Des Weiteren ist noch der Zustand *Change Planned* möglich, falls die Änderung geplant und somit implementiert werden kann. Nach erfolgreicher Implementierung (*Change Implemented*) muss das Arbeitspaket noch getestet (*Change Tested*) werden, um letztendlich in den Zustand *Change Completed* zu gelangen. Tritt ein Fehler infolge der geplanten Änderung oder nach der Implementation auf, wird das Arbeitspaket in den Status *Change Failed* gesetzt. Nun kann entschieden werden, ob die nicht erfolgreiche Änderung noch einmal neu erstellt wird (*Change New*) oder geschlossen wird (*Change Closed*). Die meisten Zustände können in *Change New* zurückversetzt werden, falls während der Bearbeitung des Arbeitspaketes Änderungen auftreten, [25], [44], [45].

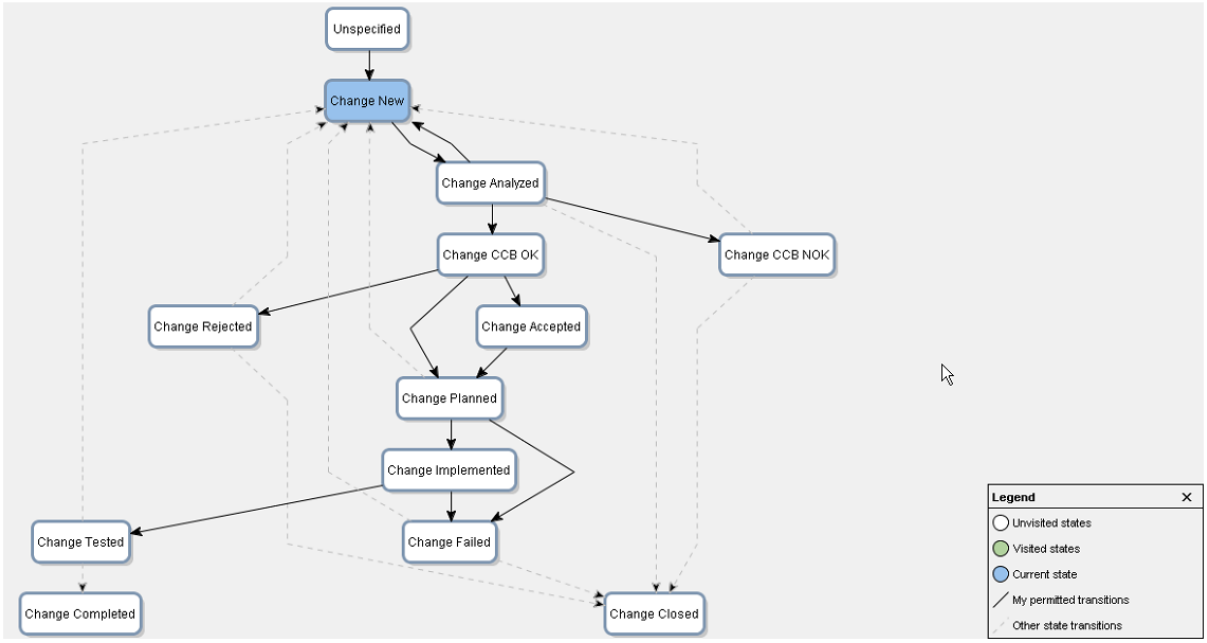


Abbildung 3.7: Workflow der *CIs*, [44]

4 Erstellung des Analysetools

Die vorliegende Masterarbeit stellt eine Mitarbeit bei der Entwicklung eines Analysetools mit Einsatz von KPIs dar, [46]. Die in den nächsten Kapiteln folgenden Inhalte sind an die Quellen [6], [46], [47], [48] und [49] angelehnt.

Die in der Masterarbeit verwendeten Beispielprojekte enthalten Entwicklungs- und Testdatensätze, um die Funktionalität des KPI Tools vollständig zu testen.

Die softwaretechnische Umsetzung der Key Performance Indikatoren geschieht – wie in den Grundlagen beschrieben – in VB.NET unter Zuhilfenahme eines Exportprogrammes, das alle relevanten Entwicklungs- und Testdaten in eine lokale Microsoft Access Datenbank exportiert. Diese Datenbank ist historisch aufgebaut. Dabei werden die Zustände der Anforderungen, Arbeitspakete und Testfälle täglich exportiert, um deren zeitlichen Verlauf analysieren zu können. Eine Front-End Access-Datei verwaltet alle vom MKS-System exportierten Dateien und fügt diese gruppiert nach Arbeitspaketen (*CI*), Anforderungen (*SCR/MR*) und *Test Cases* (*TC*) von den einzelnen Back-End-Dateien zusammen. Der Umstand, dass Microsoft Access eine Datengröße von maximal 2 GB erlaubt, erfordert eine weitere projektspezifische Aufteilung der Planungselemente, [50]. So werden mehrere Back-End Dateien erzeugt und diese im weiteren Verlauf in der Front-End Datei zu großen *CI*-, *SCR*-, *MR*- und *TC*-Tabellen zusammengefügt, um damit die Restriktion betreffend die Maximalgröße der Datenbank zu umgehen. Dies bietet den Vorteil, dass die Abfragen mit SQL an die Datenbank vollständig über das VB.NET-Programm erfolgen können. In der Front-End Datei sind bis auf die Zusammenführung der Tabellen keine SQL-Abfragen enthalten. Die Meilensteinliste wird ebenfalls der Front-End Datei angehängt, befindet sich aber extern im Projektordner, da diese Liste nicht vom MKS-System mitexportiert wird. Die prinzipielle Datenstruktur ist in Abbildung 2.3 illustriert.

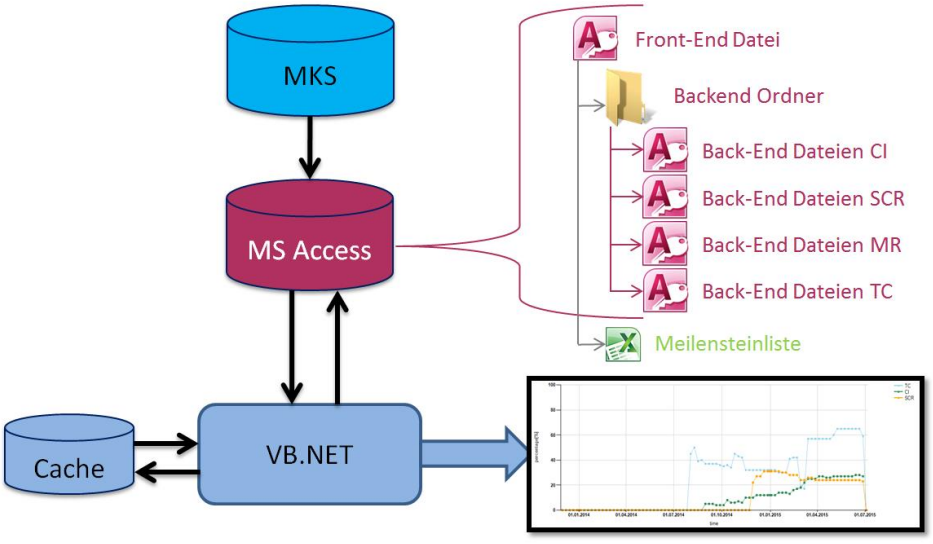


Abbildung 4.1: Datenbankverknüpfung, [25]

Das VB.NET-Programm kommuniziert mit dem Front-End Ordner der Datenbank und greift mit SQL-Befehlen lesend auf diese zu. Das Ergebnis des SQL-Statements in Form eines *DataSets* wird mit Microsoft Chart Control [27] in Diagrammen und Tabellen umgewandelt und auf der Benutzerfläche präsentiert. Aufgrund der rechenintensiven Abfragen wird jedes Ergebnis im Cache in Form einer XML-Datei gespeichert. Dies hat den Vorteil, dass bei

gleicher SQL-Abfrage nicht wiederholt auf die Microsoft Access-Datenbank zugegriffen werden muss, sondern das Ergebnis sofort in der Benutzeroberfläche angezeigt werden kann. Das Tool erkennt durch gleichen SQL-Text idente Abfragen im Cache und greift auf das gespeicherte Ergebnis direkt zu.

4.1 Struktur der Software

Das Analysetool basiert auf dem Model-View-Controller Prinzip. Das *Model* befasst sich mit der Handhabung der Daten und umfasst die Datenbank mitsamt dem Cache. Die *View* beinhaltet alle graphischen Information für die Anzeige, während der *Controller* die Eingaben seitens des Benutzers interpretiert und die Berechnungen steuert, [51]. In Abbildung 4.2 ist ersichtlich, dass die einzelnen KPIs unabhängig voneinander in Klassen getrennt sind. Jeder KPI besitzt eigenständige SQL-Statements, die über den sogenannten Datenbankhandler (*dbHandler*) Abfragen an die Datenbank schicken und mit dem Ergebnis die Diagramme und Tabellen aufbauen.

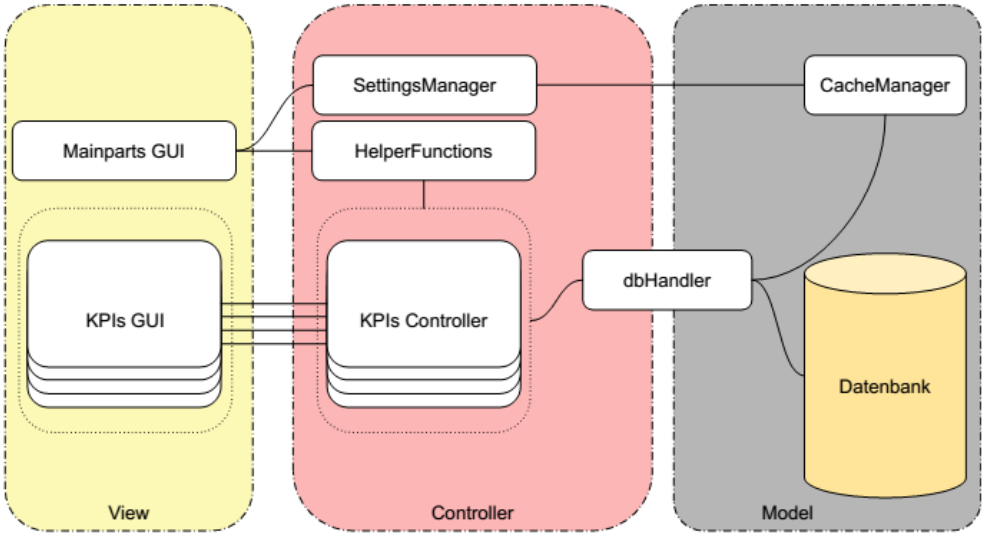


Abbildung 4.2: MVC-Aufbau des Analysetools, [52]

Wie in Abbildung 4.3 ersichtlich, wird jeder KPI für sich in einem Reiter in der Benutzeroberfläche angezeigt und durch Anklicken des gewünschten Reiters geöffnet. Dem Anwender steht nun für jeden Reiter im linken Bereich ein Eingabefenster, im rechten Bereich ein Ausgabenfenster zur Verfügung. Im linken Bereich werden die gewünschten Restriktionen eingegeben, die für den betreffenden KPI erforderlich sind. Der rechte Bereich ist für Diagramme und Tabellen reserviert, die nach Berechnungsende im Feld *Output* angezeigt werden.

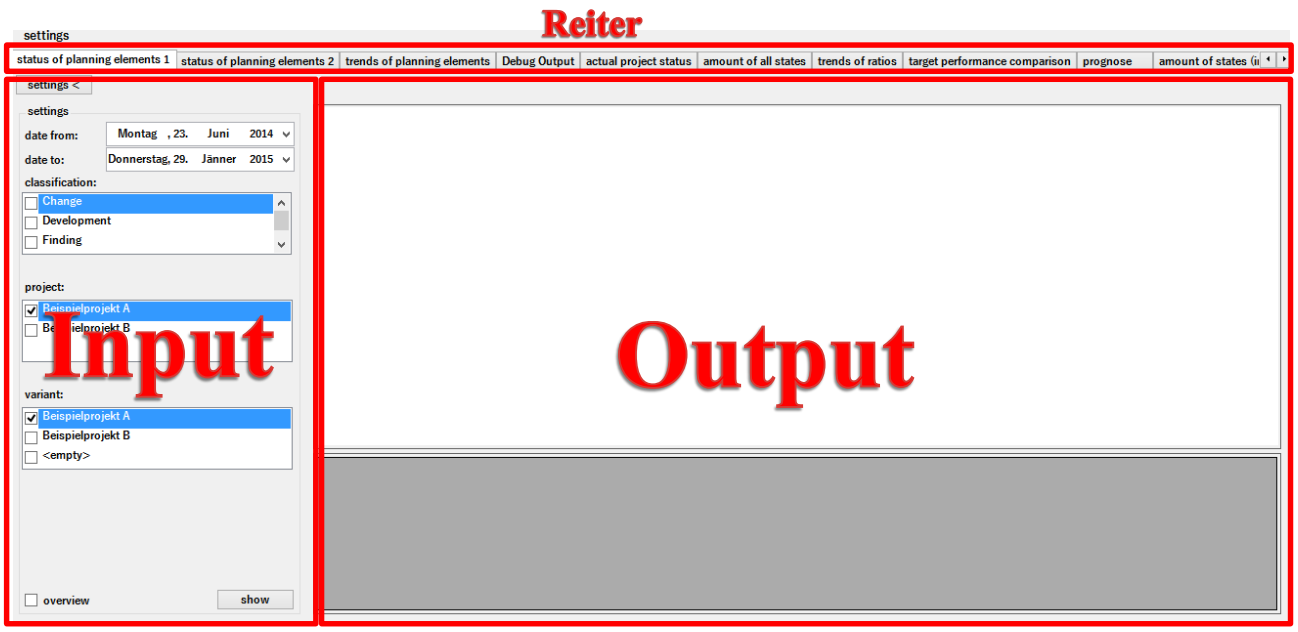


Abbildung 4.3: GUI des Analysetools

Alle Einstellungen, die für die Berechnung der KPIs benötigt werden, sind im Menü-Button *settings* anwählbar und in Abbildung 4.4 illustriert. Das Programm benötigt einen Verweis auf die zu untersuchende Datenbank, welche in der Zeile *Database* angeführt ist und die Front-End Datei beinhaltet. Die Meilensteinliste ist wie oben beschrieben nicht Teil der Datenbank und muss zusätzlich in der Zeile „Milestonelist“ festgehalten werden. Die beiden Zeilen *RScript.exe* und *R-File* sind Teil eines statistischen Projekts und werden in dieser Arbeit nicht weiterverfolgt.

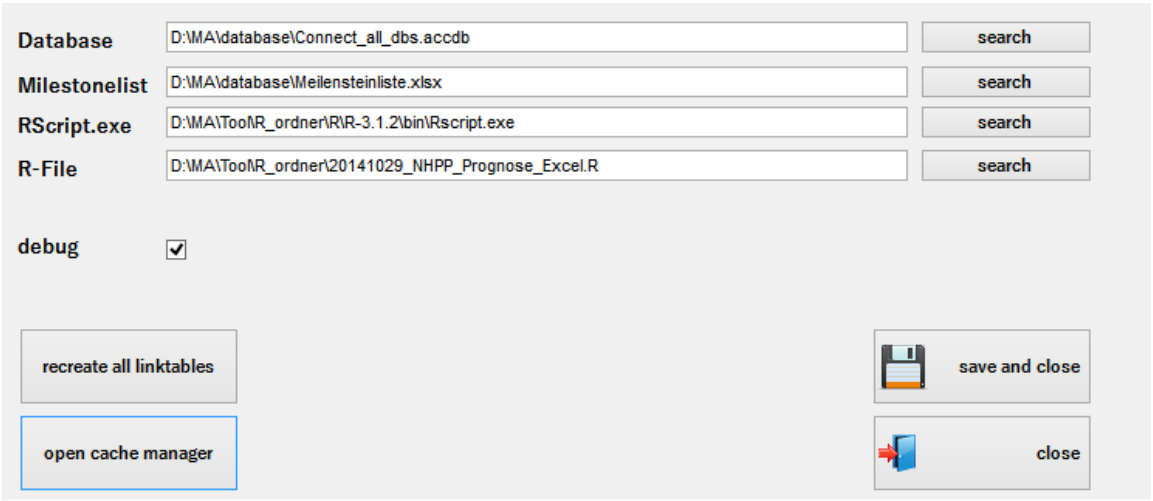


Abbildung 4.4: Menüpunkt Einstellungen

Da die Verlinkungen zwischen den Planungselementen, wie in Kapitel 3 beschrieben, in den Back-End Dateien nicht normalisiert vorliegen, müssen diese separat erstellt werden. Dazu wird der Menü-Punkt *recreate all linktables* aufgerufen und die Berechnung gestartet. Die Applikation erstellt eine *linktable* Microsoft Access-Datenbank mit Tabellen, die aus eindeutigen Verweisen der Planungselemente untereinander bestehen. Folgende Tabellen werden dabei erstellt:

- *CI_Concerns* zur Verknüpfung der *CIs* auf die *SCRs*,
- *CI_Relationshi*pTC zur Verknüpfung der *CIs* auf die *TCs*,
- *MR_Relationshi*pTC zur Verknüpfung der *MRs* auf die *TCs*,
- *SCR_Relationshi*pTC zur Verknüpfung der *SCRs* auf die *TCs*,
- *SCR_Child* zur Verknüpfung der *SCRs* auf die *MRs*.

Es wird an dieser Stelle auf die Abbildung 3.3 verwiesen, welche die Verknüpfungslogik der Elemente illustriert.

Abschließend ist noch der Menü-Punkt *open cache manager* anzuführen. Da die SQL-Abfragen der KPIs sehr rechenintensiv sind, wird jedes Ergebnis anhand dessen SQL-String Identifier abgespeichert, um es für einen späteren Abruf ohne Berechnung bzw. Datenbankzugriff verfügbar zu machen. Mit dem *Cache Manager* kann der Speicherautomatismus aktiviert oder deaktiviert werden. Man kann den Speicherort der Cache-Dateien bestimmen oder Cache-Dateien importieren.

Ein Teil der im Analysetool implementierten KPIs wird in den folgenden Kapiteln beschrieben. Dabei gliedert sich deren Beschreibung in folgende Abschnitte:

1. Im ersten Abschnitt werden die Fragestellungen (Key Performance Questions) an den Indikator beantwortet.
2. Der zweite Abschnitt befasst sich mit der graphischen Ausgabe des Ergebnisses.
3. Im dritten Abschnitt wird der Algorithmus bzw. die Berechnungsmethodik erklärt.
4. Der letzte Abschnitt erläutert Anhand eines Beispiels die Anwendung des KPIs, falls es benötigt wird.

4.2 KPI: Status of Planning Elements 1&2, [6], [47], vgl. [48], [49]

Beantwortete Fragestellungen:

Fragestellungen zum Projekt Status (Management)

- Welche Abweichungen im Projekt / in der Entwicklungsschleife gibt es?
  - Welche *Finding*s sind aufgetreten?
  - Wie lange sind diese in Bearbeitung oder nicht bearbeitet worden?
  - Wie ist das Risiko betreffend dieser *Finding*s?

Fragestellung zum Projekt Status (Projektleiter)

- Wurden Milestones des Entwicklungszyklus zeitlich und inhaltlich eingehalten?
  - Hatten die Items und Felder zu den Zeiten einen adäquaten Zustand?
- Welche Themen stellen sich als schwierig heraus?
  - Wie erkennt man kritische Themen?
  - Wie stellt man diese Themen am besten dar?
- Welche Auswirkungen können offene Punkte auf zukünftige Releases haben?
  - Wie sieht der zeitliche Verlauf von kritischen Themen aus?
- Werden Arbeitspakete termingerecht abgearbeitet?

- Wie oft werden Pakete umgeplant?
- Wie oft werden Termine eingehalten?

Fragestellungen zur Qualität (Qualitätssicherung)

- Wurden die Freezepoints erfüllt? Wenn nein, warum nicht?

Weitere Fragestellungen

- Wie ist der Status von Planungselementen bzw. *Working Packages*?
- Wie lange wurden Planungselemente bearbeitet oder nicht bearbeitet?
- Gibt es Zeitverzögerungen bzw. Umplanungen?
- Welche Planungselemente sind als kritisch einzustufen?
- Welche Abweichungen entstanden bei einem spezifischen Projekt?
- Welches Risiko beinhalten nicht abgeschlossene Planungselemente?

Dieser KPI befasst sich mit dem historischen Verlauf von *Change Issues* über einen definierten Zeitraum. Der Workflow eines Arbeitspaketes erfolgt, wie in Kapitel 3.3 beschrieben, vorgegeben Richtlinien. Um die Korrektheit des Workflows oder mögliche Verzögerungen bei der Bearbeitung eines Arbeitspaketes zu ermitteln, wird der tägliche Bearbeitungsstatus über einen gewünschten Betrachtungszeitraum farblich visualisiert. Der Benutzer muss den gewünschten Betrachtungszeitraum, die Klassifikation des Arbeitspakets, sowie die miteinbezogenen Projekten auswählen, siehe Abbildung 4.5. Das Ergebnis wird danach in einem Diagramm sowie als Tabelle ausgegeben.

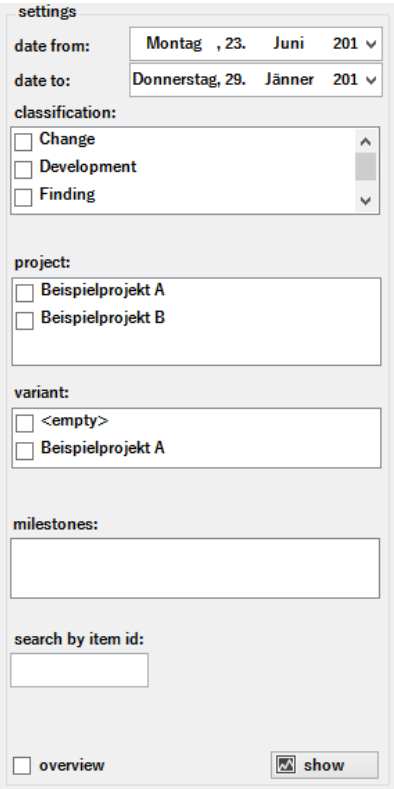


Abbildung 4.5: Status of Planning Elements 1&2, Benutzereingabe, [6], [47], vgl. [48], [49]

4.2.1 Diagramm in Status of Planning Elements 1

Die Datenpunkte werden pro Item-ID kumuliert, um die absolute Anzahl an Tagen für jedes Arbeitspaket aufzuzeigen. Jeder Datenpunkt enthält für die Ordinate die Item-ID Nummer und für die Abszisse die kumulierte Anzahl der Tage. Der größere Datenpunkt visualisiert das Zieldatum (*Target Date*) an dem das *CI* abgeschlossen sein soll (mit Status → *Change Implemented*), [6], [47], vgl. [48], [49].

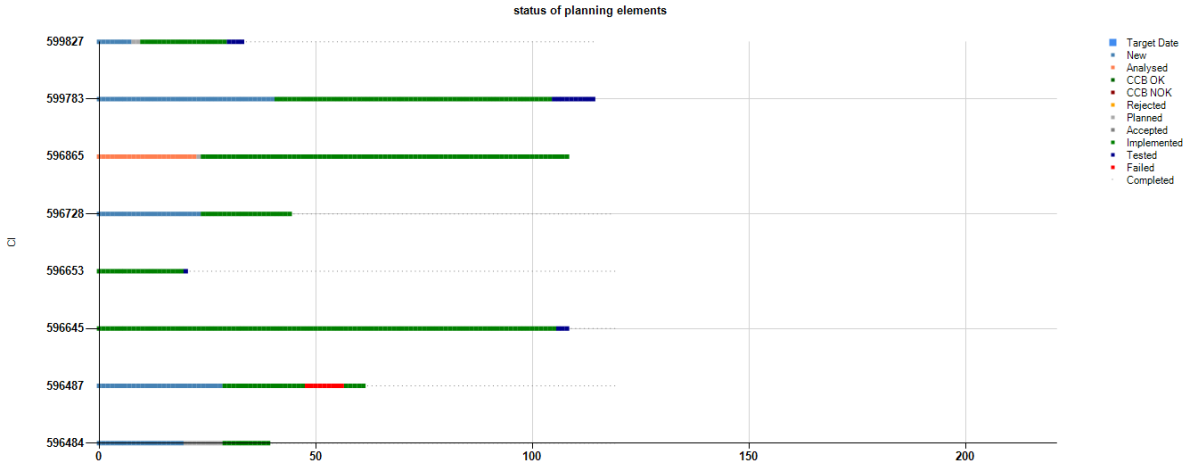


Abbildung 4.6: Status of Planning Elements 1, Diagramm, [6], [47], vgl. [48], [49]

4.2.2 Diagramm in Status of Planning Elements 2

Bei diesem Diagramm erfolgt die Anzeige datumsbezogen, d.h. jeder Datenpunkt enthält für die Ordinate die Item-ID Nummer und für die Abszisse das zugehörige Datum (wie in der *CI*-Datenbank aufgelistet). Der größere Datenpunkt visualisiert das Zieldatum (*Target Date*) an dem das *CI* abgeschlossen sein soll (mit Status → *Change Implemented*), siehe Abbildung 4.7, [6], [47], vgl. [48], [49].

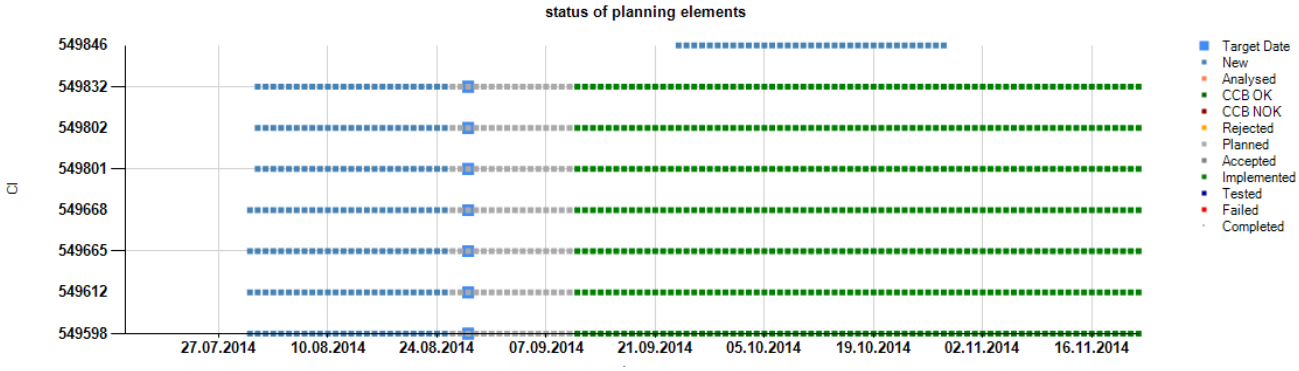


Abbildung 4.7: Status of Planning Elements 2, Diagramm, [6], [47], vgl. [48], [49]

4.2.3 Tabelle zu Status of Planning Elements 1 und 2

In der Tabelle sind folgende Spalten ersichtlich, [6], [47], vgl. [48], [49]:

- *item_id*: Die eindeutig zuordenbare Item-ID des *Change Issues*
- *project*: Projektzuordnung des *CIs*
- *variant*: Variante des IDs (in der */ProductPlatform* verwendet)
- *created date*: Erstelldatum des CIs

- *classification*: Klassifikation des CIs
- *planned SW release*: Software release, bei dem das ID geplant abgeschlossen sein sollte
- *completion SW release*: Software release, bei dem das *CI* abgeschlossen wurde
- *assigned user*: zugeordneter Benutzer
- *defined priority*: Priorität des *Change Issues*
- *Target Date*: Zieldatum, an dem das *Change Issue* implementiert sein soll
- *SIL*: zugeordnete *SIL*-Level
- *item looped*: wie oft wurde das Item-ID gelooped
- *time lag*: Verzugsberechnung, siehe Abschnitt „Berechnungsmethode/Algorithmus“
- *last record date*: Letzter möglicher Eintrag in der *CI* Liste im Betrachtungszeitraum (Es kann vorkommen, dass Item-IDs zu späteren Zeitpunkten in der *CI*-Datenbank nicht mehr vorkommen, obwohl die Datenbank kumulierend aufgebaut ist → diese Spalte dient als Indiz dafür, falls *last record date* nicht *date to* entspricht und beinhaltet den Stand der letzten Eintragung der *CI*)

item id	project	variant	created date	classification	planned SW	completion S	assigned user	defined prio
384080	Beispielprojekt A		26.11.2013 09:16:00	Change	1.005.000	SW	Max Mustermann	Med
389525	Beispielprojekt A		05.12.2013 14:42:00	Change	3.000.000	SW	Max Mustermann	Med
389526	Beispielprojekt A		05.12.2013 14:51:00	Change	2.000.000	2.000.000	Max Mustermann	Med
389541	Beispielprojekt A		05.12.2013 15:13:00	Change	2.000.000	2.000.000	Max Mustermann	Med
389542	Beispielprojekt A		05.12.2013 15:17:00	Change	2.001.000	2.001.000	Max Mustermann	Med
389545	Beispielprojekt A		05.12.2013 15:27:00	Change	2.000.000	2.000.000	Max Mustermann	Med
389546	Beispielprojekt A		05.12.2013 15:33:00	Change	2.000.000	2.000.000	Max Mustermann	Med
389547	Beispielprojekt A		05.12.2013 15:36:00	Change	3.000.000	SW	Max Mustermann	Med
389548	Beispielprojekt A		05.12.2013 15:39:00	Change	3.000.000	SW	Max Mustermann	Med
389550	Beispielprojekt A		05.12.2013 15:47:00	Change	3.000.000	SW	Max Mustermann	Med
389584	Beispielprojekt A		06.12.2013 09:17:00	Change	2.001.000	2.001.000	Max Mustermann	Med
389602	Beispielprojekt A		06.12.2013 09:24:00	Change	2.000.000	2.000.000	Max Mustermann	Med

target date	SIL	State	item looped	time lag	last record date
	A	Change Planned	1		29.01.2015 00:00:00
20.02.2015		Change New	1	-22	29.01.2015 00:00:00
31.03.2014	No	Change Implemented	1	99	29.01.2015 00:00:00
31.12.2013	No	Change Implemented	1	211	29.01.2015 00:00:00
15.10.2014	No	Change Implemented	2	9	29.01.2015 00:00:00
31.12.2013	No	Change Implemented	1	189	29.01.2015 00:00:00
31.12.2013	No	Change Implemented	1	211	29.01.2015 00:00:00
20.02.2015		Change New	1	244	29.01.2015 00:00:00
20.02.2015		Change New	1	-22	29.01.2015 00:00:00
20.02.2015		Change New	1	-22	29.01.2015 00:00:00
15.10.2014	No	Change Implemented	1	15	29.01.2015 00:00:00
31.12.2013	No	Change Implemented	1	169	29.01.2015 00:00:00

Abbildung 4.8: Status of Planning Elements 1 & 2, Tabelle, [6], [47], vgl. [48], [49]

4.2.4 Berechnungsmethodik/Algorithmus

Die Berechnungsmethode ist für Status of Planning Elements 1 und 2 gleich und unterscheidet sich lediglich in der Diagrammanzeige. Die Berechnung wird mit dem *show*-Button gestartet. Hierfür wird mit SQL-Befehlen auf die Datenbank bzw. die Meilensteinliste zugegriffen. Eine Auswahl des Benutzers, wie Start- und Zieldatum, Klassifikation, Projekt, Variante schränken die Datenbank je nach Anforderung ein. Jedes *CI* hat für einen Tag einen bestimmten Status wie z.B. *Change New* oder *Change Implemented* und erhält für jeden Tag einen Tabelleneintrag. Zusätzlich wird – falls vorhanden – aus der Meilensteinliste der Meilenstein bzw. die Systemfreigabe und das Fälligkeitsdatum, an dem das *CI* abschlossen sein sollte (*Target Date*), angehängt. Die erzeugte Tabelle aus SQL-Abfragen wird in einem Diagramm visualisiert. Falls

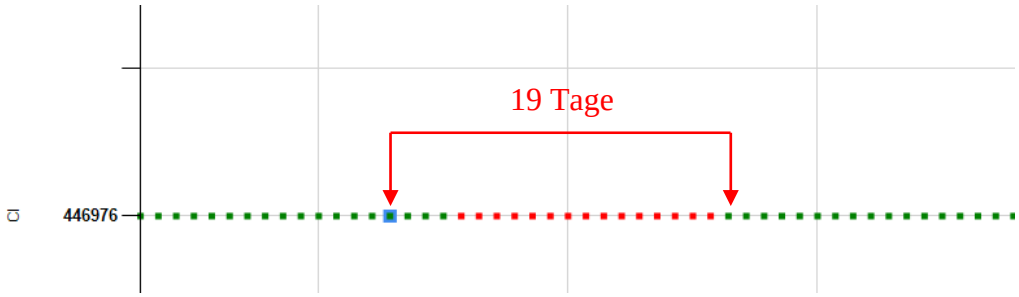
sich nun an einem Tag der Status eines *CI* ändert, wird die Farbe entsprechend dem Zustand angepasst.

In der Tabelle wird aus Übersichtsgründen nur der letzte mögliche Tabelleneintrag für die Anzeige herangezogen. Es kann vorkommen, dass *CIs* aus einem Projekt in die */ProductPlatform* aufgenommen werden bzw. auf *Change Closed* gesetzt werden. Die Anzeige des *CIs* zum Zieldatum (*date-to*) ist in diesem Falle daher nicht mehr möglich. Ersichtlich wird dieser Umstand im Tabelleneintrag *last record date*. Falls dieser nun nicht gleich dem Zieldatum ist, kann davon ausgegangen werden, dass sich das *CI* in der */ProductPlatform* befindet oder auf *Change Closed* gesetzt wurde.

Der Zeitverzug (*time lag*) eines *CIs* berechnet sich folgenderweise: Voraussetzung zur Berechnung ist ein vorhandener Stichtag (*Target Date*) zu dem das *CI* abgeschlossen, also im Zustand *Change Implemented*, sein soll. Befindet sich nun das *CI* zum Stichtag nicht im Zustand *Change Implemented*, wird fortlaufend ein Verzug berechnet bis zu jenem Datum an dem das *CI* abgeschlossen ist. Wird im betrachteten Zeitraum das *CI* nicht abgeschlossen, ergibt sich der Verzug aus der Differenz zwischen Stichtag (*Target Date*) und Zieldatum (*last record date*). Es kann durchaus vorkommen, dass das *CI* nach dem Status *Change Implemented* noch einmal zu einem vorherigen Status zurückversetzt, also „geloopt“ wird und somit zu einem späteren Zeitpunkt noch einmal den Status *Change Implemented* erreicht. In diesem Fall wird die letzte Zeitspanne in dem sich das *CI* im Status *Change Implemented* befindet, zur Verzugsberechnung herangezogen, [6], [47], vgl. [48], [49].

4.2.5 Beispiel zur Verzugsberechnung

Das Diagramm mit der zugehörigen Tabelle ist in Abbildung 4.9 ersichtlich: Das Item ID 446976 springt zweimal in den Status *Change Implemented* (grüne Punkte im Diagramm). Der Verzug in Tagen ergibt sich nun aus der Differenz von *Target Date* (15.05.2014) und dem ersten Tag im Status *Change Implemented* der letzten möglichen Iterationsschleife.



item id	project	time lag
446976	*Beispielprojekt A*	19

Abbildung 4.9: Status of Planning Elements 2, Beispiel Diagramm und Ausschnitt aus der Tabelle, [6], [47], vgl. [48], [49]

Status of Planning Elements verschafft dem Anwender einen Überblick über die Dauer und Güte des Workflows der Arbeitspakete. Zeitverzüge einzelner *Change Issues* können schnell erfasst werden und animieren die Projektgruppe, diese möglichst schnell abzuschließen, um weitere Verzögerungen im Projektverlauf zu vermeiden. Durch die graphische Darstellung kann die Bearbeitungsdauer insgesamt bzw. die Zeitintervalle der einzelnen Bearbeitungsstatus jedes Arbeitspakets dargestellt werden. Zusammen mit der Tabelle können dann Entscheidungen getroffen werden, ob sich dieses Paket hinsichtlich Priorität, geplantem

Software Release oder *Target Date* in einem adäquaten Bearbeitungsstatus befindet.

4.3 KPI: Amount of All States, [6], [47], vgl. [48], [49]

Beantwortete Fragestellungen:

- Wie ist der Entwicklungsstatus aller Projekte bzw. eines spezifischen Projektes?
- Wie viele Arbeitsschritte sind im Gange und wie ist deren Status?

Der KPI „Amount of All States“ gibt Auskunft über die wöchentliche Anzahl von Planungselementen. Die Anzahl der im Projektverlauf erstellten Arbeitspakete, Anforderungen oder Testfälle wird wöchentlich als Balken dargestellt, siehe Abbildung 4.10. Die einzelnen Status werden farblich unterscheidbar dargestellt und zum jeweiligen Planungselement gruppiert. Die Anzahl der Einträge spiegelt sich in der Länge der Balken wieder und kann bei Überfahren mit dem Mauszeiger zusätzlich als Kurzinfo angezeigt werden. Darin sind Informationen zum Status des betreffenden Elements, Anzahl der Elemente sowie das Datum enthalten. Der Entwicklungsprozess nach dem V-Modell erfordert ein strukturiertes Vorgehen bei der Planung von Arbeitspaketen, Anforderungen oder Testfällen. Falls nun zu viele Elemente einer Ebene im V-Modell geplant wurden, ist dies im KPI „Amount of all States“ in Form erhöhter Ungleichverteilung von Planungselementen erkenntlich.

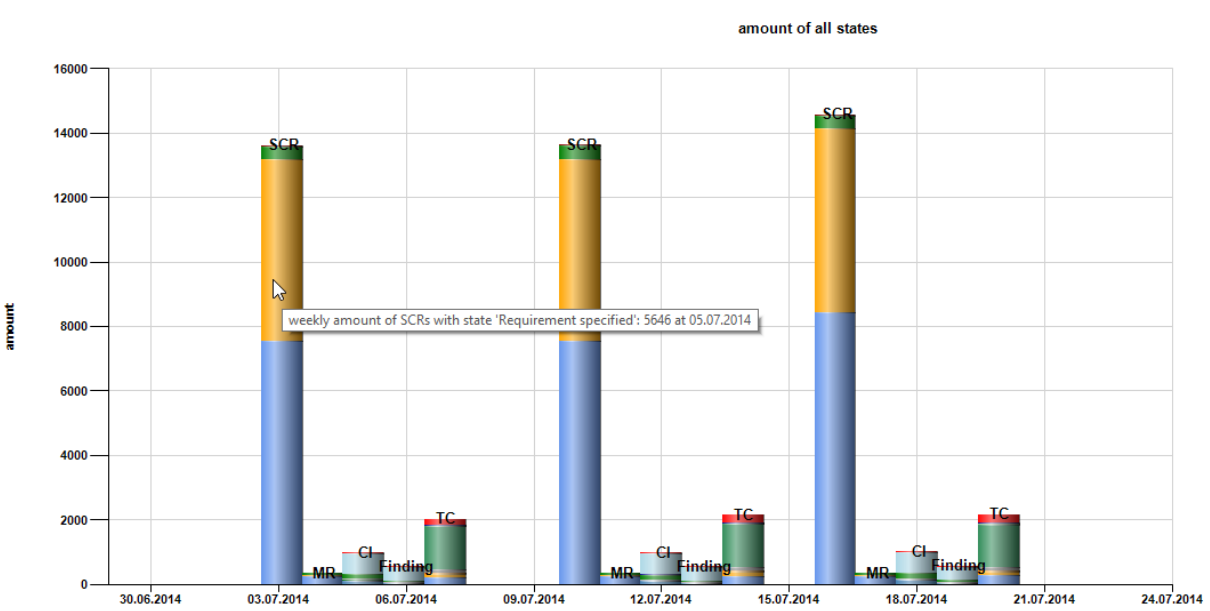


Abbildung 4.10: Amount of All States, Diagramm, [6], [47], vgl. [48], [49]

4.3.1 Berechnungsmethodik/Algorithmus

Nach Anwählen des gewünschten Betrachtungszeitraums (*date from* und *date to*), der Planungselemente und anschließendem Starten der Berechnung mit dem *show*-Button werden die betreffenden Datenbanken (*CI, SCR, MR* und *TC*) wöchentlich abgerufen. Die einzelnen Item-IDs werden nach deren Status gruppiert und dem Diagramm bzw. der Tabelle zugewiesen, [6], [47], vgl. [48], [49].

4.3.2 Beispiel zur Berechnung des letzten Wochentags

Da die historischen Datenbanken der Arbeitspakete, Anforderungen und Testfälle täglich

aktualisiert werden, aber für diesen KPI ein wöchentlicher Report ausreicht, werden die Planungselemente zu einem Wochentag zusammenfasst. SQL stellt dazu die *DATEPART*-Funktion zur Verfügung. Die eindeutige Kennzeichnung einer Woche wird durch die Bezeichnung der Jahreszahl und der Kalenderwoche festgelegt und kann in weiterer Folge durch die *MAX*-Klausel zu einem eindeutigen Datum ausgegeben werden, siehe SQL-Beispiel unten, [6], [47], vgl. [48], [49].

```
SELECT MAX(snapshot_date) as weekly_snap

FROM ci

WHERE snapshot_date >= #2014-07-01# and snapshot_date <= #2014-07-18#

GROUP BY ((DATEPART('yyyy', snapshot_date) & DATEPART('ww',
snapshot_date)))
```

Zum Abgleich dieser SQL-Abfrage werden im Analysetool folgende Einstellungen getroffen, vergleiche Abbildung 4.11, [6], [47], vgl. [48], [49]:

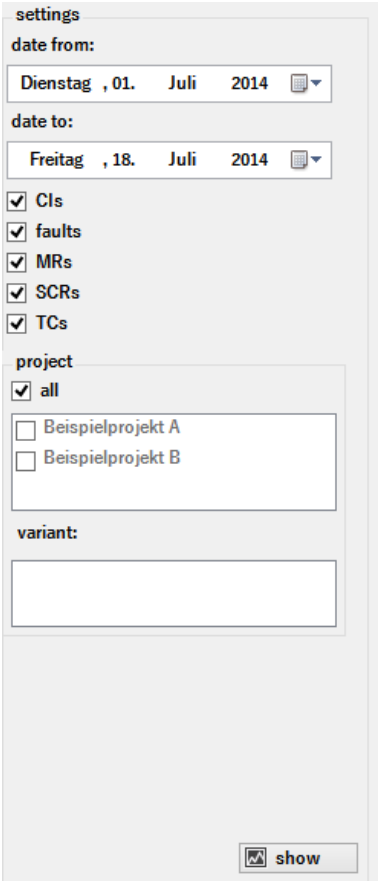


Abbildung 4.11: Amount of All States, Benutzereingabe, [6], [47], vgl. [48], [49]

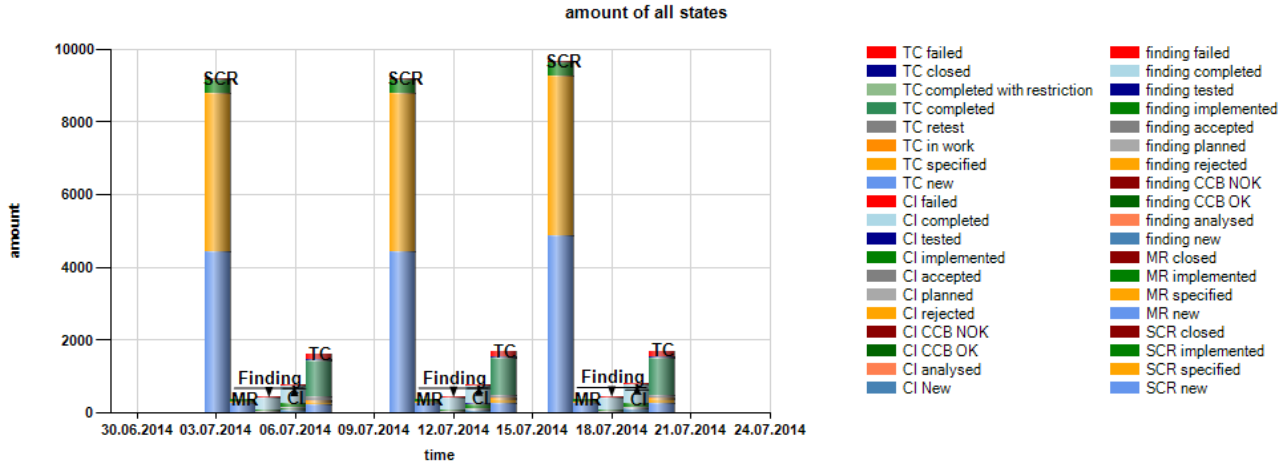
Nach Abschluss der Berechnung sind auf der Benutzeroberfläche pro Woche vier gruppierte Balken ersichtlich, die die Anzahl der Planungselemente quantifizieren. Es wird im Diagramm immer der letzte Arbeitstag einer Arbeitswoche angezeigt, siehe Abbildung 4.12.

Im konkreten Beispiel, [6], [47], vgl. [48], [49]:

05.07.2014 → letztes Datum in der Arbeitswoche 30.06.2014-05.07.2014

12.07.2014 → letztes Datum in der Arbeitswoche 07.07.2014-12.07.2014

18.07.2014 → in diesem Fall nicht der 19.07.2014, weil das Enddatum vor dem letzten Wochentag liegt.



Snapshot Date	CI New	CI Analysed	CI Planned	CI CCB OK	CI CCB NOK	CI Rejected	CI Accepted	CI Implemented	CI Complete
05.07.2014 00:0...	05	11	18	8	0	0	0	113	500
12.07.2014 00:0...	06	10	6	8	0	0	0	117	515
18.07.2014 00:0...	121	10	8	10	0	0	0	122	517

Abbildung 4.12: Amount of All States, Beispiel, [6], [47], vgl. [48], [49]

4.4 KPI: Amount of All States in work/done, [6], [47], vgl. [48], [49]

Beantwortete Fragestellungen:

- Wie ist der Entwicklungsstatus aller Projekte bzw. eines spezifischen Projektes?
- Wie viele Arbeitsschritte sind im Gange und wie ist deren Status?

Der KPI Amount of All States in work/done verhält sich ähnlich dem KPI Amount of All States mit dem Unterschied, dass bestimmte Status des Workflows zu *in work* und *done* zusammengefasst werden.

Die wöchentliche Anzeige der gewählten Planungselemente erfolgt mittels Balken. Die einzelnen Status werden farblich unterscheidbar gemacht. Jedes Planungselement besteht aus zwei gruppierten Balken, die den Vergleich von abgeschlossenen zu den nicht abgeschlossenen Arbeitspaketen liefern. In Tabelle 4.1 sind die Zuordnungen von abgeschlossenen und nicht abgeschlossenen Planungselementen ersichtlich.

Die Anzahl der Planungselemente spiegelt sich in der Länge der Balken wieder und kann bei Überfahren mit dem Mauszeiger zusätzlich als Kurzinfo angezeigt werden. Darin sind Informationen zum Status des betreffenden Elements, zur Anzahl der Elemente, sowie das Datum enthalten, siehe Abbildung 4.13, [6], [47], vgl. [48], [49].

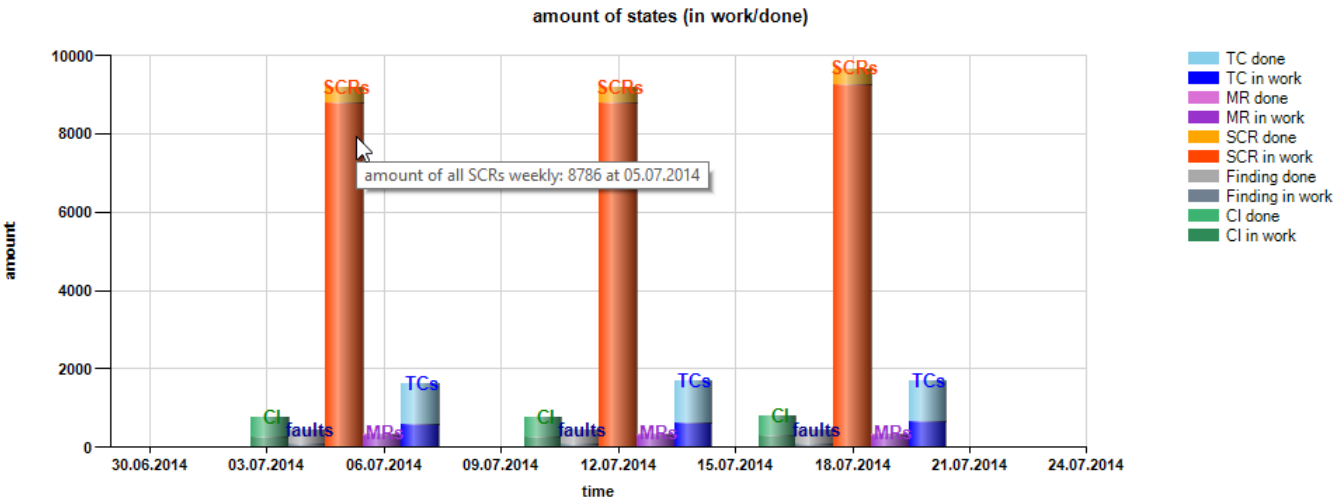


Abbildung 4.13: Amount of States in work/done, Diagramm, [6], [47], vgl. [48], [49]

4.4.1 Berechnungsmethodik/Algorithmus

Nach Anwählen des gewünschten Betrachtungszeitraums der Planungselemente und anschließendem Starten der Berechnung werden die betreffenden Datenbanken (*CI, SCR, MR* und *TC*) wöchentlich abgerufen. Die Item-IDs werden je nach deren Status zu *in work* oder *done* zugeordnet, siehe Tabelle 4.1:

Tabelle 4.1: Amount of All States, Zuordnungen, [6], [47], vgl. [48], [49]

Planungselement	**Status für Eintrag: *in work***	**Status für Eintrag: *done***
CIs	*Change New, Change Analysed, Change CCB OK, Change CCB NOK, Change Rejected, Change Accepted, Change Planned, Change Implemented, Change Failed, Change Tested*	*Change Completed*
faults (*CIs* mit *classification Finding*)	*Change New, Change Analysed, Change CCB OK, Change CCB NOK, Change Rejected, Change Accepted, Change Planned, Change Implemented, Change Failed, Change Tested*	*Change Completed*
SCRs	*Requirement New, Requirement Specified*	*Requirement Implemented*
MRs	*Requirement New, Requirement Specified*	*Requirement Implemented*
TCs	*TC New, TC Specified, TC in Work, TC Retest, TC failed*	*TC Completed, TC Completed with restriction*

4.4.2 Beispiel zur Berechnung des letzten Wochentags nach einem Jahreswechsel

Die Ermittlung des letzten Wochentags erfolgt analog zum vorigen KPI „Amount of All States“ und kann in Kapitel 4.4 nachgelesen werden, [6], [47], vgl. [48], [49].

In der *settings*-Bar werden folgende Einstellungen getroffen:

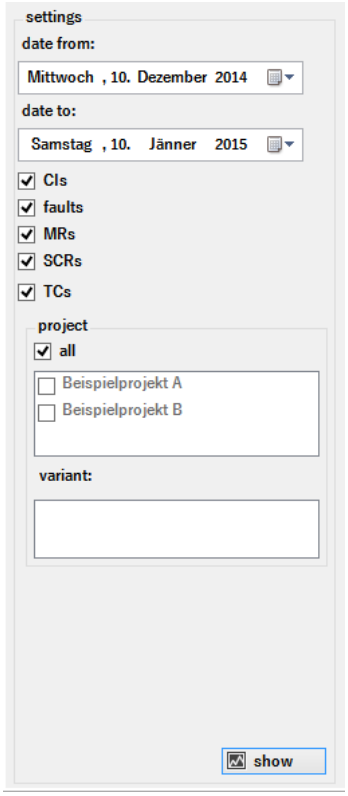


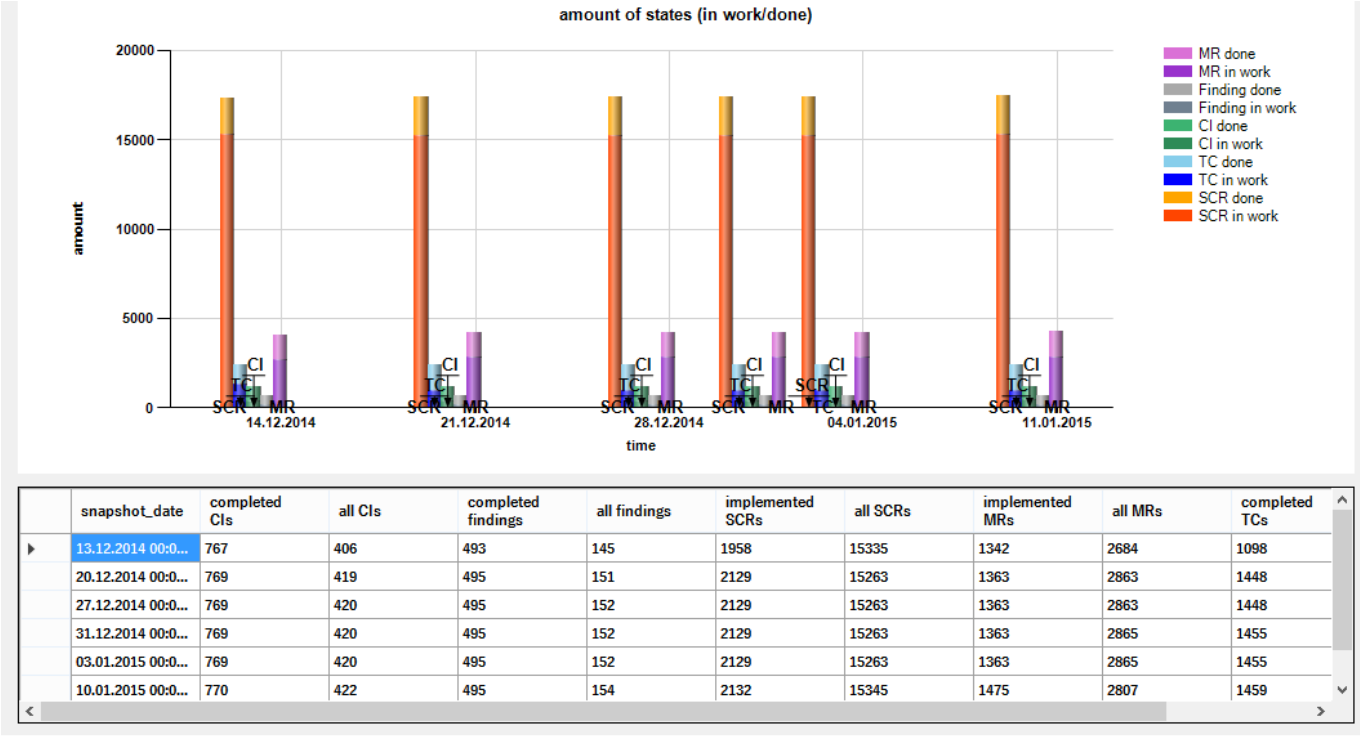
Abbildung 4.14: Amount of All States in work/done, Benutzereingabe, [6], [47], vgl. [48], [49]

Nach Abschluss der Berechnung erscheinen pro Woche vier gruppierte Balken, welche die Anzahl der Planungselemente quantifizieren. Es wird im Diagramm immer der letzte Arbeitstag einer Arbeitswoche angezeigt.

Im konkreten Beispiel, [6], [47], vgl. [48], [49]:

13.12.2014 → letztes Datum in der Arbeitswoche 08.12.2014-13.12.2014

31.12.2015 und 03.01.2015 → letztes Datum im Jahr 2014 und erste Woche im Jahr 2015 werden angezeigt, darum befinden sich die Balken knapp nebeneinander.



snapshot_date	completed CIs	all CIs	completed findings	all findings	implemented SCRs	all SCRs	implemented MRs	all MRs	completed TCs
13.12.2014 00:0...	767	406	493	145	1958	15335	1342	2684	1098
20.12.2014 00:0...	769	419	495	151	2129	15263	1363	2863	1448
27.12.2014 00:0...	769	420	495	152	2129	15263	1363	2863	1448
31.12.2014 00:0...	769	420	495	152	2129	15263	1363	2865	1455
03.01.2015 00:0...	769	420	495	152	2129	15263	1363	2865	1455
10.01.2015 00:0...	770	422	495	154	2132	15345	1475	2807	1459

Abbildung 4.15: Amount of States in work/done, Beispiel, [6], [47], vgl. [48], [49]

4.5 KPI: Trend of Ratios, [6], [47], vgl. [48], [49]

Beantwortete Fragestellungen:

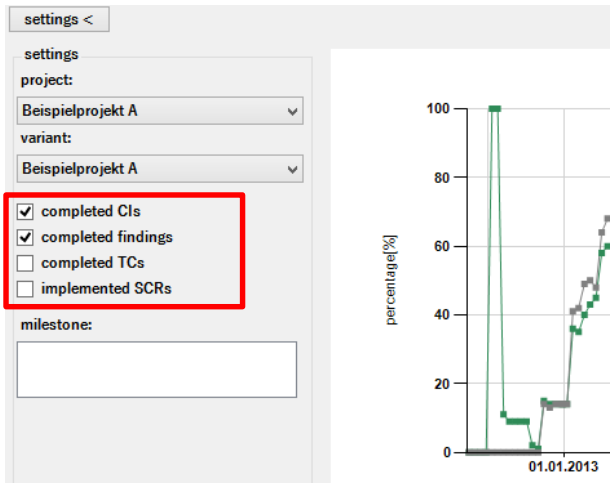
Fragestellungen zum Projekt Status (Management)

- Welche Abweichung im Projekt / in der Entwicklungsschleife gibt es?
  - Welche *Findings* sind aufgetreten?
  - Wie lange sind diese in Bearbeitung oder nicht bearbeitet worden?
  - Wie ist das Risiko betreffend dieser *Findings*?
- Welches Risiko beinhaltet das aktuelle Projekt?
  - Welche Auswirkungen haben die bekannten Fehler?
  - Wie groß ist das Risiko von versteckten Fehlern (z.B. Testabdeckung, oder Extrapolation basierend auf bekannten Informationen)?
- Können wir die Systemfreigabe erteilen?
- Welche Ableitungen von Zielwerten können aus den Auswertungen mehrerer Projekte gezogen werden?

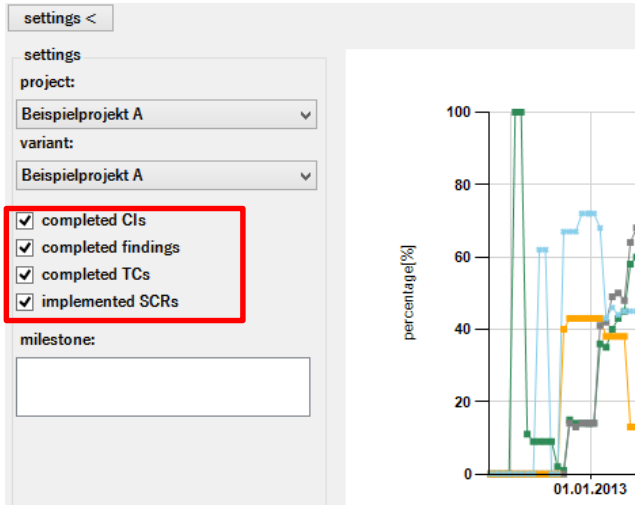
Weitere Fragestellungen

- Wie ist der Verlauf des Implementierungsstatus?
- Wie viele Planungselemente wurden im Vergleich zu den gesamten Planungselementen umgesetzt?
- Wie viele Fehler wurden aufgedeckt im Vergleich zu den implementierten Funktionen?
- Wie hoch ist die Testabdeckung?
- Wie ist der Verlauf der absolvierten Tests über die Produktreife?

Ziel dieses KPIs ist es, Trends von projektspezifischen Planungselementen erkennen und beurteilen zu können. Wöchentlich wird ein Vergleich von abgeschlossenen zu nicht abgeschlossenen Arbeitspaketen, Anforderungen oder Testfällen durchgeführt. Die Zuordnung von der abgeschlossenen und nicht abgeschlossenen KPIs ist in Kapitel 4.4, Tabelle 4.1 illustriert. Es ergibt sich aufgrund der relativen Häufigkeiten eine Trendlinie, mit deren es möglich ist, den Projektverlauf objektiv beurteilen zu können. Durch diese Häufigkeitsverteilung kann ebenso ein Vergleich der Planungselemente untereinander durchgeführt werden. Je nach ausgewählten Elementen (*CIs*, *Findings*, *TCs*, *SCRs*) erfolgt die Anzeige als fortlaufende Linie unterschiedlicher Farbe mit dem Prozentsatz des Erfüllungsgrads. Die Anzeige erfolgt wöchentlich und wird jeweils zum Ende einer Arbeitswoche als Punkt gekennzeichnet. Die Linien können einzeln durch Ab- und Anwählen der Kontrollkästchen ein- oder ausgeblendet werden, siehe Abbildung 4.16, [6], [47], vgl. [48], [49].



Auswahl *completed CIs* und *completed Findings:* Im Diagramm werden für detaillierte Untersuchungen einzelner Ebenen im V-Modell nur die ausgewählten Planungselemente angezeigt.



Alle Planungselemente sind ausgewählt.

Abbildung 4.16: Trend of Ratios, Auswahlmöglichkeit der Planungselemente, [6], [47], vgl. [48], [49]

4.5.1 Tabelle zu Trend of Ratios

Die Tabelle wird je nach ausgewähltem Planungselement erweitert. Darin befinden sich folgende Informationen, [6], [47], vgl. [48], [49]:

- *snapshot_date*: Wochentag, an dem die Planungselemente abgefragt werden
- *completed CIs, completed Findings, implemented SCRs, completed TCs*: Anzahl der Elemente mit Status *Completed* (*CIs, Findings, TCs*) bzw. *Implemented* (*SCRs*)
- *all CIs, all Findings, all SCRs, all TCs*: Anzahl aller Planungselemente zum betreffenden Datum

percentage CIs, percentage Findings, percentage SCRs, percentage TCs: Prozentuale Anzeige des Erfüllungsgrades der Anzahl der Planungselemente *completed* oder *implemented* im Verhältnis zur Gesamtanzahl der Planungselemente. Diese Prozentanzeige findet sich auch im Diagramm wieder.

snapshot_date	completed CIs	all CIs	percentage CIs	completed findings	all findings	percentage findings	implemented SCRs	all SCRs	percentage SCRs	completed TCs	all TCs	percentage TCs
08.09.2012 00:0...	0	1	0	0	0	0	0	0	0	0	0	0
15.09.2012 00:0...	0	1	0	0	0	0	0	0	0	0	0	0
22.09.2012 00:0...	0	1	0	0	0	0	0	0	0	0	89	0
29.09.2012 00:0...	0	1	0	0	0	0	0	0	0	0	89	0
06.10.2012 00:0...	1	1	100	0	0	0	0	3	0	0	89	0
13.10.2012 00:0...	1	1	100	0	0	0	0	6	0	0	89	0
20.10.2012 00:0...	1	9	11	0	6	0	0	6	0	0	89	0
27.10.2012 00:0...	1	11	9	0	8	0	0	8	0	0	89	0
03.11.2012 00:0...	1	11	9	0	8	0	0	8	0	104	169	62

Abbildung 4.17: Trend of Ratios, Ausschnitt der Tabelle, [6], [47], vgl. [48], [49]

4.5.2 Berechnungsmethodik/Algorithmus

Die Berechnung wird nach dem Auswählen der gewünschten Einstellung wie Projekt, Variante und Planungselement (im *settings*-Panel) mit dem *show*-Button gestartet. Es folgt ein Zugriff auf die *CI*-Datenbanken der Projekte bzw. auf die Meilensteinliste, welche die Wochentage (*snapshot_date*) für die weiteren Berechnungen vorgeben. Der Betrachtungszeitraum richtet sich nach den *CI*-Einträgen. Dies bedeutet, dass das erste und letzte projektbezogene *Change Issue* die Zeitspanne bestimmt. Das gilt auch für *Findings*, *SCRs* und *TCs*. Je nach angewählten Kontrollkästchen werden folgende Berechnungsmethoden gestartet, [6], [47], vgl. [48], [49]:

4.5.2.1 Berechnung *completed CIs* und *completed Findings*

Die *CI*-Datenbank wird wöchentlich nach abgeschlossenen *Change Issues* im Status *completed* und nach allen *Change Issues* durchsucht. Dasselbe gilt für *completed Findings*, diese werden jedoch zusätzlich nach der Klassifikation (Eintrag in der *CI*-Datenbank: *classification*) *Finding* gefiltert. Die Werte werden dann den Spalten *completed CIs* bzw. *all CIs* zugewiesen. Der Prozentsatz ergibt sich aus dem Quotienten von *completed CIs* und *all CIs*. Analog dazu ist die Vorgehensweise bei den *Findings*, [6], [47], vgl. [48], [49].

4.5.2.2 Berechnung *completed TCs*

Entsprechend zur Berechnungsmethode der *CIs* im vorigen Absatz werden die Erfüllungsgrade der *Test Cases* abgefragt. Der dafür benötigte Status für *completed TCs* ist *TC completed* oder *TC completed with restriction*, [6], [47], vgl. [48], [49].

4.5.2.3 Berechnung *implemented SCRs*

Analog zur Berechnungsmethode der *CIs* im vorigen Absatz werden die Erfüllungsgrade der *System Component Requirements* abgefragt. Der dafür benötigte Status für *implemented SCRs* ist *SCR implemented*, [6], [47], vgl. [48], [49].

4.5.3 Beispiel zur Berechnung

Anhand eines Beispiels soll die Vorgehensweise für die Handhabung dieses KPIs verdeutlicht werden. In der *settings*-Bar werden folgende Einstellungen getroffen:

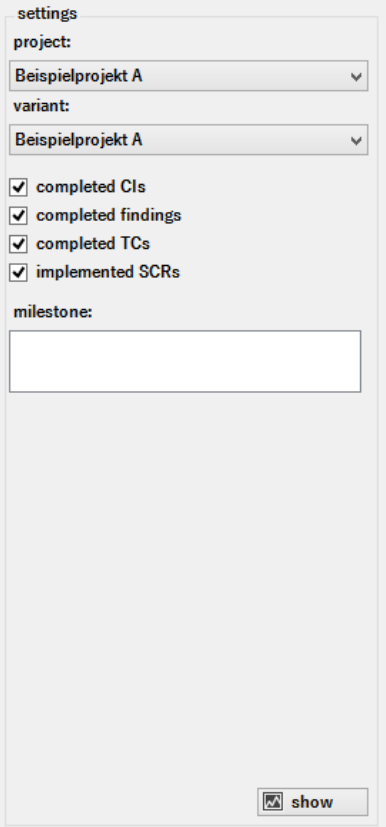


Abbildung 4.18: Trend of Ratios, Benutzereingabe, [6], [47], vgl. [48], [49]

Die Planungselemente werden überlappend angezeigt und farblich unterschieden. Die farbliche Unterscheidung ist in der Tabelle ebenfalls ersichtlich, siehe Abbildung 4.19. Für genauere Untersuchungen der Verläufe einzelner Planungselemente, wie sie in Kapitel 5 erfolgt, sind Ausschnitte des Diagrammbereichs nötig. Auffälligkeiten und Abweichungen des Entwicklungsprozesses werden erst durch Zoomen des Bereichs sichtbar, wie in Abbildung 4.20 erkenntlich, [6], [47], vgl. [48], [49].

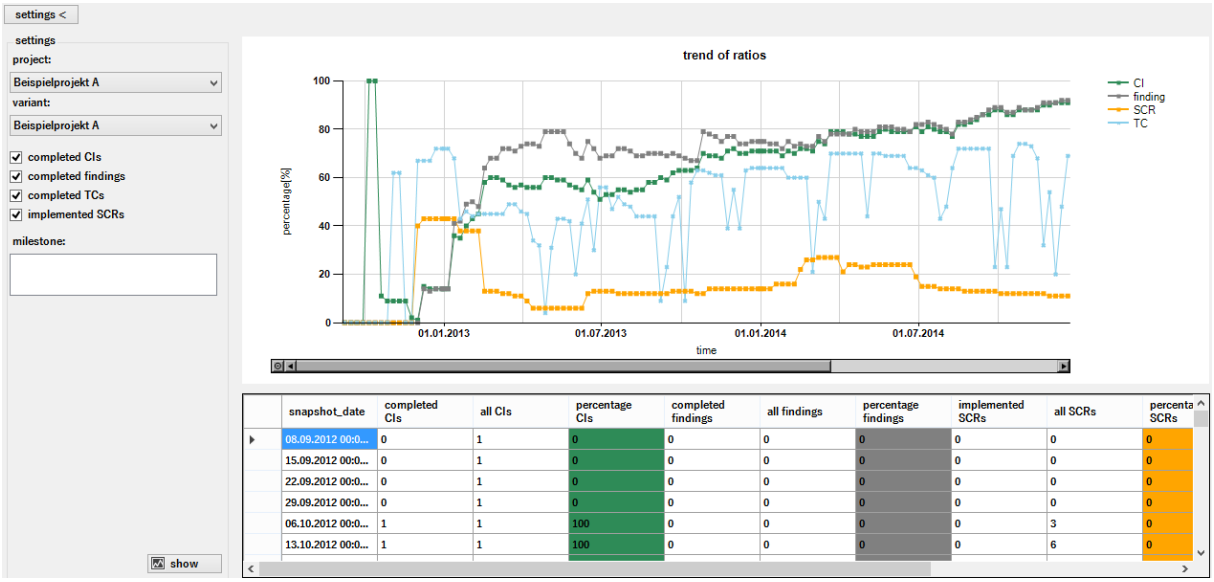


Abbildung 4.19: Trend of Ratios, Beispiel, [6], [47], vgl. [48], [49]

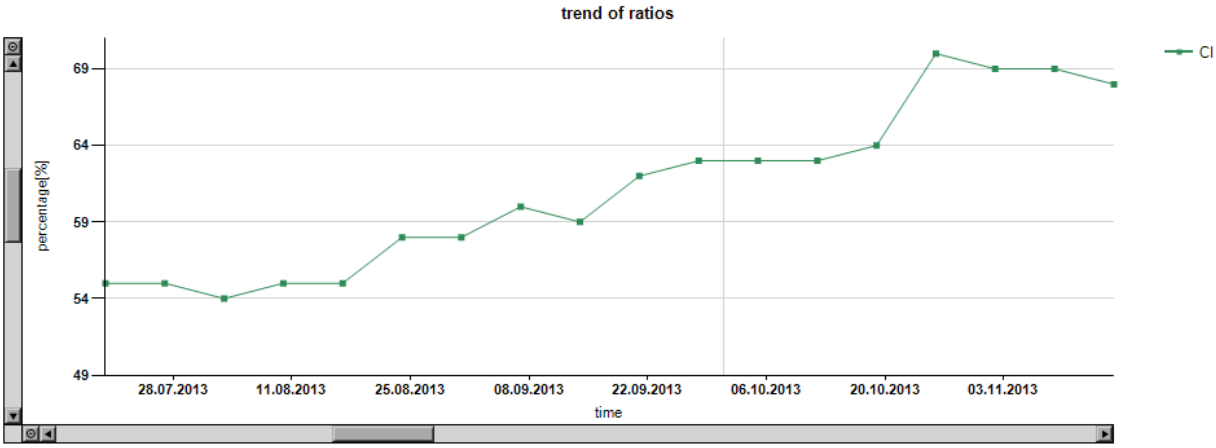


Abbildung 4.20: Trend of Ratios, gezoomtes Beispiel, [6], [47], vgl. [48], [49]

4.6 KPI: Target Performance Comparison, [6], [47], vgl. [48], [49]

Beantwortete Fragestellungen:

Fragestellungen zum Projekt Status (Management)

- Wo stehen wir im Projekt und in der aktuellen Entwicklungsschleife?
  - Wie ist der Milestone-Status hinsichtlich Inhalt, Zeit und Qualität zu beurteilen?
- Welche Abweichungen im Projekt/Entwicklungsschleife gibt es?
  - Welche *Findings* sind aufgetreten?
  - Wie lange sind diese in Bearbeitung oder nicht bearbeitet worden?
  - Wie ist das Risiko betreffend dieser *Findings*?
- Werden die Prozesse und Entwicklungsschleifen in der Praxis umgesetzt?
  - Wie gestaltet sich die Trend-Analyse betreffend der Freeze-Points (Inhalt, Zeit, Qualität)?
- Welches Risiko beinhaltet das aktuelle Projekt?
  - Welche Auswirkungen haben die bekannten Fehler?

    - Wie groß ist das Risiko von versteckten Fehlern (z.B. Testabdeckung, oder Extrapolation basierend auf bekannten Informationen)?
- Können wir die Systemfreigabe erteilen?

Fragestellung zum Projekt Status (Projektleiter)

- Wurden Milestones des Entwicklungszyklus zeitlich und inhaltlich eingehalten?
    - Hatten die Items und Felder zu den Zeiten einen adäquaten Zustand?
- Welche Themen stellen sich als schwierig heraus?
    - Wie erkennt man kritische Themen?
    - Wie stellt man diese Themen am besten dar?
- Welche Auswirkung können offene Fragestellungen auf zukünftige Releases haben?
    - Wie sieht der zeitliche Verlauf bei der Behandlung kritischer Themen aus?
- Werden Arbeitspakete termingerecht abgearbeitet?
    - Wie oft werden Pakete umgeplant?
    - Wie oft werden Termine eingehalten?

Fragestellungen zur Qualität (Qualitätssicherung)

- Wie kann durch kontinuierliche Beobachtung der SOLL-/IST-Werte die Qualitätssicherung gewährleistet werden?
  SOLL → was soll erreicht werden (Umfang anhand z.B. Planungselemente),
  IST → welcher Umfang (bzw. Features) ist ungesetzt und einsatzbereit pro Entwicklungszyklus
- Wurden die Freezepoints erfüllt? Wenn nein, warum nicht?
- Ist die Qualität der Software ausreichend?

Charakterisierend für diesen KPI ist die starke Anlehnung an den Entwicklungsprozess nach dem V-Modell des Industriepartners, da nur jene Planungselemente zur Berechnung herangezogen werden, die nach dem Schema in Abbildung 3.3 aufgebaut sind. Ähnlich dem KPI „Trend of Ratios“ visualisiert „Target Performance Comparison“ relative Häufigkeiten von abgeschlossenen zu nicht abgeschlossenen Planungselementen. Das Diagramm beinhaltet – je nach angewählten Planungselementen – Balken, die für jede Systemfreigabe – aus der Meilensteinliste vorgegeben – einen Soll-/Ist-Vergleich in Prozent anzeigen, siehe Abbildung 4.21. Dabei erfolgt die fortlaufende Anzeige der Systemfreigaben NICHT nach aufsteigender Systemfreigabenummer, sondern nach deren Zieldatum in der Meilensteinliste. Aufgrund der Tatsache, dass es im Entwicklungsprozess des Industriepartners zu Umplanungen der Systemfreigabe kommt, ergibt sich diese Prämisse der fortlaufenden Sortierung nach Datum, vergleiche Abbildung 4.21, [6], [47], vgl. [48], [49].

Die Planungselemente lassen sich auch einzeln durch An- und Abwählen der Kontrollkästchen in der *settings*-Bar anzeigen. Mit der implementierten Zoomfunktion lassen sich einzelne Systemfreigaben vergrößern. Die Kurzinfo veranschaulicht bei Überfahren der Balken mit dem Mauszeiger die Berechnungsmethode des Planungselements, [6], [47], vgl. [48], [49].

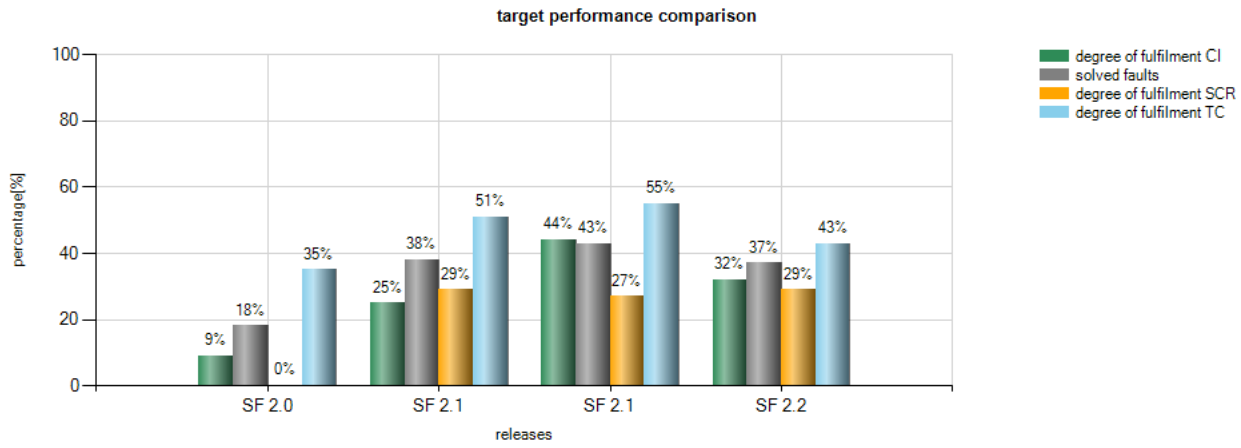


Abbildung 4.21: Target Performance Comparison, Diagramm, [6], [47], vgl. [48], [49]

4.6.1 Tabelle zu Target Performance Comparison

Zusätzlich zum Diagramm sind die prozentualen Werte zu jeder Systemfreigabe bzw. zu jedem KPI in der Tabelle ersichtlich und farblich gekennzeichnet. Sie ergeben sich aus dem Quotienten aus *performance* und *target*. Die anderen Einträge werden in Kapitel 4.6.2 erklärt, [6], [47], vgl. [48], [49].

	release	target CI	performance CI	target findings	performance findings	degree of fulfilment	solved faults
▶	SF 2.0	94	8	17	3	9	18
	SF 2.1	211	171	126	47	81	37
	SF 2.1	262	171	126	47	65	37
	SF 2.2	432	138	125	44	32	35

Abbildung 4.22: Target Performance Comparison, Ausschnitt aus der Tabelle, [6], [47], vgl. [48], [49]

4.6.2 Berechnungsmethodik/Algorithmus

4.6.2.1 Berechnung der *duration* aus der Tabelle

Das *Target Date* entspricht dem zu einem Software Release bzw. einer Freigabe zugeordneten Datum aus der Meilensteinliste. Zu diesem Datum sollten die *CIs* mit gleichem geplantem Software Release (aus der *CI*-Datenbank mit Spaltenbezeichnung *Planned_SW_Release_String*) abgeschlossen sein. Die *duration* berechnet sich aus der Differenz zwischen *Target Date* und erstmals auftretenden projektbezogenem Software Release aus der Meilensteinliste, [6], [47], vgl. [48], [49].

4.6.2.2 Algorithmus zu den *CIs* und *Findings*

target CI & „target *Findings*"

Für die Berechnung von *target CI* werden alle projektbezogenen *Change Issues* gezählt, deren geplante Software Release (aus der *CI*-Datenbank mit Spaltenbezeichnung *Planned_SW_Release_String*) kleiner oder gleich dem *SW Release* aus der Meilensteinliste ist. Diese werden nach Systemfreigaben gruppiert. Der Systemfreigabe zugehörige Software-Release, sowie das Datum werden aus der Meilensteinliste abgerufen und mit der *CI*-Datenbank verlinkt, siehe Tabelle 4.2. Die *target Findings* werden analog zu *target CI* berechnet mit der

Einschränkung auf die Klassifikation *Finding* aus der Datenbank, [6], [47], vgl. [48], [49].

Tabelle 4.3: Target Performance Comparison, Meilensteinliste, [6], [47], vgl. [48], [49]

3.000.000	23.06.2015	System Release	SF 2.2
3.000.000	16.06.2015	Feature Freeze	
3.000.000	30.06.2015	Implementation Freeze	

performance CI & performance Findings

Für die Berechnung von *performance CI* werden alle abgeschlossenen (im Status *Change Completed*) projektbezogenen *Change Issues* gezählt, deren geplante Software Release (aus der *CI*-Datenbank mit Spaltenbezeichnung *Planned_SW_Release_String*) kleiner oder gleich dem *SW Release* aus der Meilensteinliste ist. Diese werden nach Systemfreigaben gruppiert. Der Systemfreigabe zugehörige Software Releases, sowie das Datum werden aus der Meilensteinliste abgerufen und mit der *CI*-Datenbank verlinkt, siehe Tabelle 4.4. Die *performance Findings* werden analog zu *performance CI* berechnet mit der Einschränkung auf die Klassifikation *Finding* aus der Datenbank, [6], [47], vgl. [48], [49].

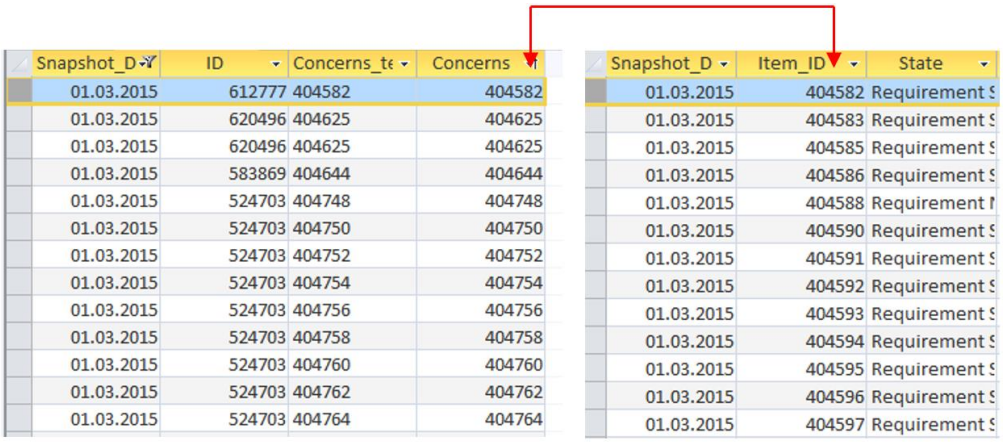
degree of fulfilment CI & solved faults

Der Quotient aus *performance CI* und *target CI* ergibt den Erfüllungsgrad (*degree of fulfilment*) in Prozent für die *Change Issues* bzw. *solved faults* und berechnet sich aus dem Quotienten aus *target Findings* und *performance Findings*. Die Quotienten sind farblich als Balken im Diagramm bzw. in der Tabelle ersichtlich, [6], [47], vgl. [48], [49].

4.6.2.3 Algorithmus zu den *SCRs*

target SCR

Hierbei werden alle projektbezogenen *SCRs*, die über *Concerns* mit der *CI*-Datenbank verbunden sind, zu jeder Systemfreigabe gezählt. Dabei werden nur jene *SCR* Item-IDs herangezogen, die den IDs aus der *CI*-Datenbank *CI_concerns* entsprechen, [6], [47], vgl. [48], [49].



Snapshot_D	ID	Concerns_te	Concerns
01.03.2015	612777	404582	404582
01.03.2015	620496	404625	404625
01.03.2015	620496	404625	404625
01.03.2015	583869	404644	404644
01.03.2015	524703	404748	404748
01.03.2015	524703	404750	404750
01.03.2015	524703	404752	404752
01.03.2015	524703	404754	404754
01.03.2015	524703	404756	404756
01.03.2015	524703	404758	404758
01.03.2015	524703	404760	404760
01.03.2015	524703	404762	404762
01.03.2015	524703	404764	404764

Snapshot_D	Item_ID	State
01.03.2015	404582	Requirement S
01.03.2015	404583	Requirement S
01.03.2015	404585	Requirement S
01.03.2015	404586	Requirement S
01.03.2015	404588	Requirement I
01.03.2015	404590	Requirement S
01.03.2015	404591	Requirement S
01.03.2015	404592	Requirement S
01.03.2015	404593	Requirement S
01.03.2015	404594	Requirement S
01.03.2015	404595	Requirement S
01.03.2015	404596	Requirement S
01.03.2015	404597	Requirement S

Abbildung 4.23: Target Performance Comparison, Concerns, [6], [47], vgl. [48], [49]

performance SCR

Hierbei werden alle implementierten (Status *Requirement Implemented*) projektbezogenen *SCRs*, die über *Concerns* mit der *CI*-Datenbank verbunden sind, zu jeder Systemfreigabe gezählt. Dabei werden nur jene *SCR* IDs zur Berechnung herangezogen, die den Item-IDs aus der *CI*-Datenbank *CI_concerns* entsprechen, [6], [47], vgl. [48], [49].

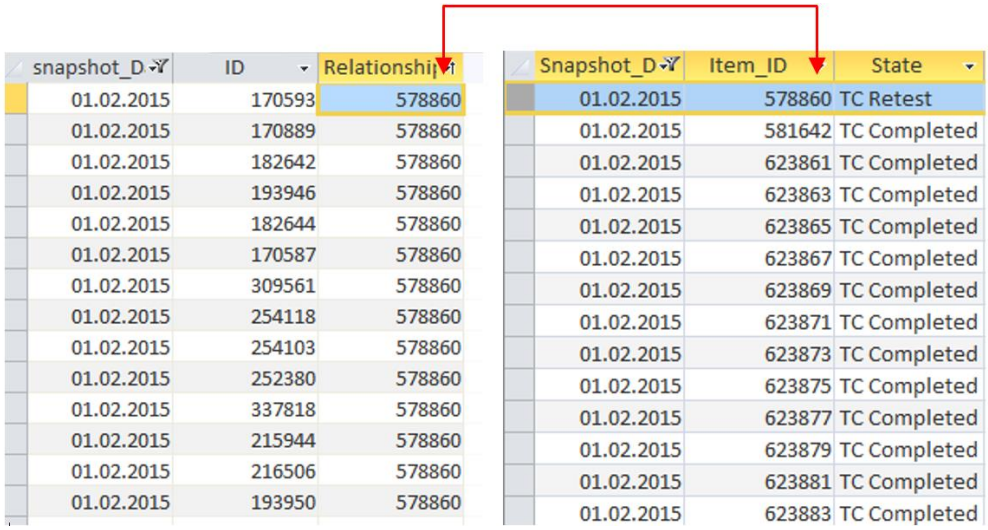
degree of fulfilment SCR

Der Quotient aus *performance SCR* und *target SCR* ergibt den Erfüllungsgrad (*degree of fulfilment SCR*) in Prozent für die *System Component Requirements*, [6], [47], vgl. [48], [49].

4.6.2.4 Algorithmus zu den *TCs*

target TC

Hierbei werden alle projektbezogenen *TCs*, welche über *RelationshipTestIssue* mit der *CI*-Datenbank, *SCR*-Datenbank und *MR*-Datenbank verbunden sind, zu jeder Systemfreigabe gezählt. Dabei werden nur jene *TC* IDs herangezogen, die den *RelationshipTestIssue*-IDs aus den *CI*, *SCR* und *MR* Datenbanken entsprechen. Die betreffenden IDs für dieses Planungselement werden vorher in den *Linktables* berechnet, [6], [47], vgl. [48], [49].



snapshot_D	ID	Relationship
01.02.2015	170593	578860
01.02.2015	170889	578860
01.02.2015	182642	578860
01.02.2015	193946	578860
01.02.2015	182644	578860
01.02.2015	170587	578860
01.02.2015	309561	578860
01.02.2015	254118	578860
01.02.2015	254103	578860
01.02.2015	252380	578860
01.02.2015	337818	578860
01.02.2015	215944	578860
01.02.2015	216506	578860
01.02.2015	193950	578860

Snapshot_D	Item_ID	State
01.02.2015	578860	TC Retest
01.02.2015	581642	TC Completed
01.02.2015	623861	TC Completed
01.02.2015	623863	TC Completed
01.02.2015	623865	TC Completed
01.02.2015	623867	TC Completed
01.02.2015	623869	TC Completed
01.02.2015	623871	TC Completed
01.02.2015	623873	TC Completed
01.02.2015	623875	TC Completed
01.02.2015	623877	TC Completed
01.02.2015	623879	TC Completed
01.02.2015	623881	TC Completed
01.02.2015	623883	TC Completed

Abbildung 4.24: Target Performance Comparison, *RelationshipTestIssue*, [6], [47], vgl. [48], [49]

performance TC

Hierbei werden alle abgeschlossenen (Status *TC Completed* oder *TC Completed with Restriction*) projektbezogenen *TCs*, welche über *RelationshipTestIssue* mit der *CI*-Datenbank, *SCR*-Datenbank und *MR*-Datenbank verbunden sind, zu jeder Systemfreigabe gezählt. Dabei werden nur jene *TC* IDs herangezogen, die den *RelationshipTestIssue*-IDs aus den *CI*, *SCR* und *MR* Datenbanken entsprechen. Die betreffenden IDs für dieses Planungselement werden vorher in den *Linktables* berechnet, [6], [47], vgl. [48], [49].

degree of fulfilment *TC*

Der Quotient aus performance *TC* und target *TC* ergibt den Erfüllungsgrad („degree of fulfilment TC“) in Prozent für die *Test Cases*, [6], [47], vgl. [48], [49].

4.6.3 Beispiel zur Berechnung

Anhand eines Beispiels soll die Vorgehensweise für die Handhabung dieses KPIs verdeutlicht werden. In der *settings*-Bar werden folgende Einstellungen getroffen, [6], [47], vgl. [48], [49]:

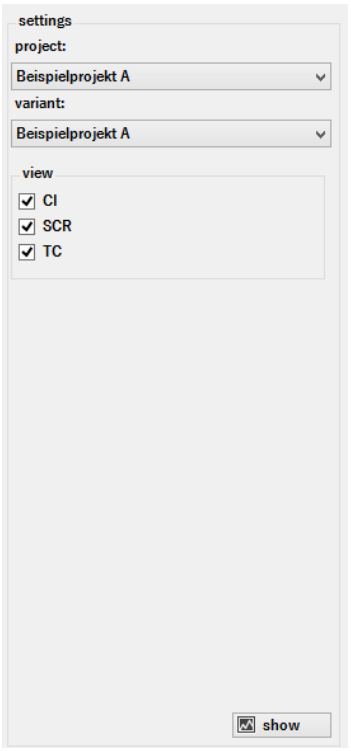
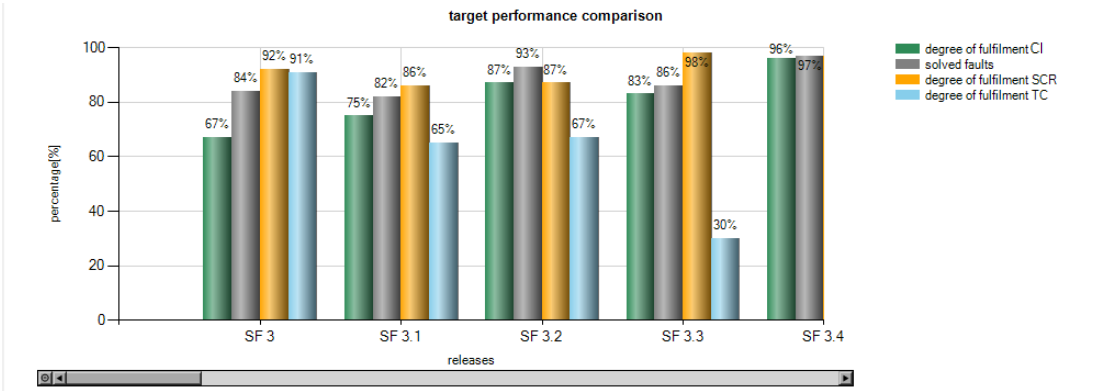


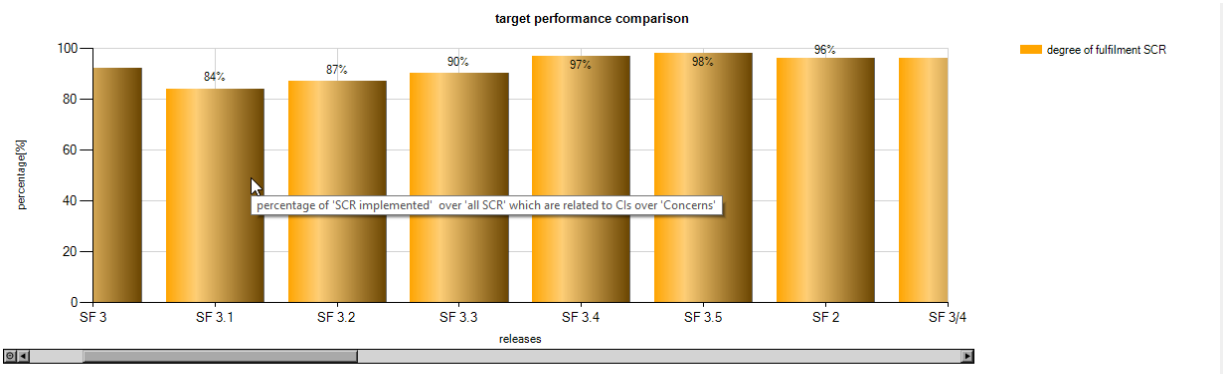
Abbildung 4.25: Target Performance Comparison, Benutzereingabe, [6], [47], vgl. [48], [49]

Nach Abschluss des Berechnungsvorgangs sind die prozentualen Anteile der Zielerfüllung der angewählten Planungselemente in Form von Balken ersichtlich, siehe Abbildung 4.26. Diese sind für jede Systemfreigabe gruppiert und sind einzeln in den Kontrollkästchen in der *settings*-Bar an- und abwählbar. Bei mehreren Systemfreigaben empfiehlt es sich für Detailuntersuchungen, die Planungselemente einzeln anzuwählen und ggf. zu zoomen, siehe Abbildung 4.27. Die Tabelle enthält zusätzliche Informationen wie Anzahl der Planungselemente, der Systemfreigabe zugeordnetes Zieldatum (*Target Date*) und Laufzeit (*duration*) nach erstmaligem Auftreten eines projektbezogenen Software Releases, [6], [47], vgl. [48], [49].



release	sw release	target CI	performance CI	target findings	performance findings	degree of fulfilment	solved faults	target date	duration
SF 3	3.000.080	117	78	88	74	67	84	27.06.2013 00:0...	269
SF 3.1	3.010.073	174	131	127	104	75	82	10.10.2013 00:0...	374
SF 3.2	3.020.071	214	187	150	140	87	93	09.12.2013 00:0...	434
SF 3.3	3.030.072	333	278	225	193	83	86	11.03.2014 00:0...	526
SF 3.4	3.030.074	311	300	211	205	96	97	28.04.2014 00:0...	574

Abbildung 4.26: Target Performance Comparison, Beispiel, [6], [47], vgl. [48], [49]



release	sw release	target CI	performance CI	target findings	performance findings	degree of fulfilment	solved faults	target date	duration	target SCR	performance SCR
SF 3	3.000.080	117	78	88	74	67	84	27.06.2013 00:0...	269	83	76
SF 3.1	3.010.073	174	131	127	104	75	82	10.10.2013 00:0...	374	90	76
SF 3.2	3.020.071	214	187	150	140	87	93	09.12.2013 00:0...	434	89	77
SF 3.3	3.030.072	333	278	225	193	83	86	11.03.2014 00:0...	526	189	171
SF 3.4	3.030.074	311	300	211	205	96	97	28.04.2014 00:0...	574	184	179
SF 3.5	3.030.074	305	300	207	205	98	99	15.05.2014 00:0...	591	184	180

Abbildung 4.27: Target Performance Comparison, Detailansicht, [6], [47], vgl. [48], [49]

4.7 KPI: Iteration Durations, [6], [47]

Beantwortete Fragestellungen:

Fragestellungen zu den Durchlaufzeiten (Prozessmanagement)

- Durchlaufzeiten:
  - Wie groß sind die aktuellen Durchlaufzeiten?
  - Welche Durchlaufzeiten sind von Relevanz?
  - Wo liegen die Ober- und Untergrenzen?
  - Wie verhalten sich die Laufzeiten unter Berücksichtigung
    - des Entwicklungszyklus, und
    - der Projekt-/Produktreife?

  - Welche Ober- und Untergrenzen sind aussagekräftig?
  - Welche realistischen Ober- und Untergrenzen können definiert werden?
  - Womit muss ein Projekt in den unterschiedlichen Projektphasen bzgl. der Prozessanwendung rechnen?
  - Können Trends bei der Prozessabarbeitung festgestellt werden?
- Prozessfluss:
  - Wie sieht der aktuelle Prozessfluss aus?
  - Wo und wie kann der Prozessfluss vereinfacht werden?
  - Wo und wie müssen Ergänzungen eingeführt werden?

Fragestellungen zum Projekt Status (Management)

- Wo stehen wir im Projekt und in der aktuellen Entwicklungsschleife?
  - Wie ist der Milestone-Status hinsichtlich Inhalt, Zeit und Qualität zu beurteilen?
- Welche Abweichung im Projekt / in der Entwicklungsschleife gibt es?
- Werden die Prozesse und Entwicklungsschleifen in der Praxis umgesetzt?
  - Wie gestaltet sich die Trend-Analyse betreffend der Freeze-Points (Inhalt, Zeit, Qualität)?
- Welches Risiko beinhaltet das aktuelle Projekt?
- Können wir die Systemfreigabe erteilen?
- Welche Zielwerte lassen sich aus Auswertungen über mehrere Projekte ableiten?

Fragestellung zum Projekt Status (Projektleiter)

- Wurden Milestones des Entwicklungszyklus zeitlich und inhaltlich eingehalten?
  - Hatten die Items und Felder zu den Zeiten einen adäquaten Zustand?
- Welche Themen stellen sich als schwierig heraus?
  - Wie erkennt man kritische Themen?
  - Wie stellt man diese Themen am besten dar?
- Welche Auswirkung können offene Punkte auf zukünftige Releases haben?
  - Wie sieht der Zeitliche Verlauf von kritischen Themen aus?
- Werden Arbeitspakete termingerecht abgearbeitet?
  - Wie oft werden Pakete umgeplant?
  - Wie oft werden Termine eingehalten?

Fragestellungen zur Qualität (Qualitätssicherung)

- Wie kann durch kontinuierliche Beobachtung der SOLL-/IST Werte und des Freeze-Point Status die Qualitätssicherung gewährleistet werden?
  SOLL → was soll erreicht werden (Umfang anhand z.B. Planungselemente),
  IST → welcher Umfang (bzw. Features) ist ungesetzt und einsatzbereit pro Entwicklungszyklus
- Wurden die Freezepoints erfüllt? Wenn nein, warum nicht?

Im Zuge von großen Software Projekten wird der Projektfortschritt oft in Form verschiedener Etappen durch Software Releases unterteilt. Dabei sollen diese Zwischenziele gewisse Anforderungen erfüllen, um eine grobe Funktionstüchtigkeit verschiedener Softwaremodule zu bewerkstelligen. Mehrere Iterationsschleifen werden durchlaufen und somit der Reifegrad des

Softwareprodukts erhöht, was durch ansteigenden Software Release gekennzeichnet wird. Bei einem späteren Software Release wird die vollständige Serienreife eines Produkts erreicht und die Systemfreigabe an den Kunden kann erfolgen.

Der KPI „Iteration Durations“ beschäftigt sich mit dieser Vorgehensweise und unterteilt das Softwareprojekt in verschiedene Etappen. Dabei wird jeder Software Release in eine Reihe von Freezes unterteilt, bei dem bestimmte Anforderungen erfüllt sein müssen, siehe Abbildung 4.28. In der Berechnungsmethodik werden die Bedingungen für einen erfolgreichen Abschluss (100% Erfüllungsgrad) eines Freezes erläutert. Jeder Freeze muss dabei eindeutig gekennzeichnet sein und enthält neben dessen Namen auch den zugehörigen Software Release und das zugehörige Datum, [6], [47].

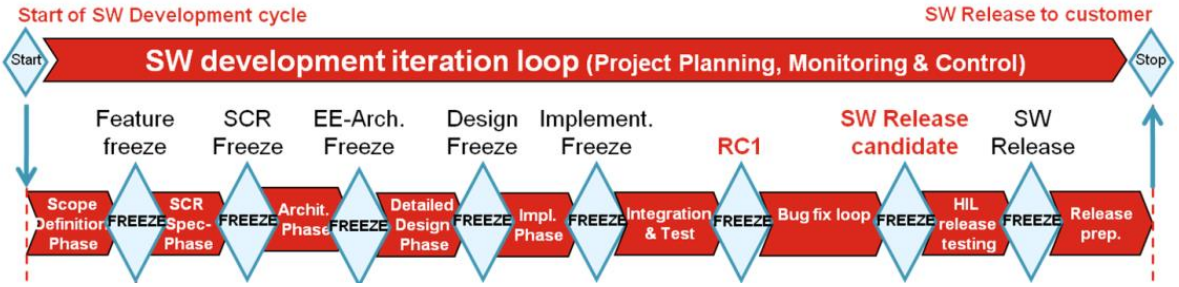


Abbildung 4.28: Iteration Duration, SW development loop, [53]

In der Auswertung dieses KPIs wird daher auf der Ordinate der Software Release aufgetragen und auf der Abszisse nach den auftretenden Freezes gruppiert (*over freezes* in der *settings*-Bar) oder die Freezes nach zeitlicher Abfolge dargestellt (*over time* in der settings-bar), vergleiche Abbildung 4.31 und Abbildung 4.32, [6], [47].

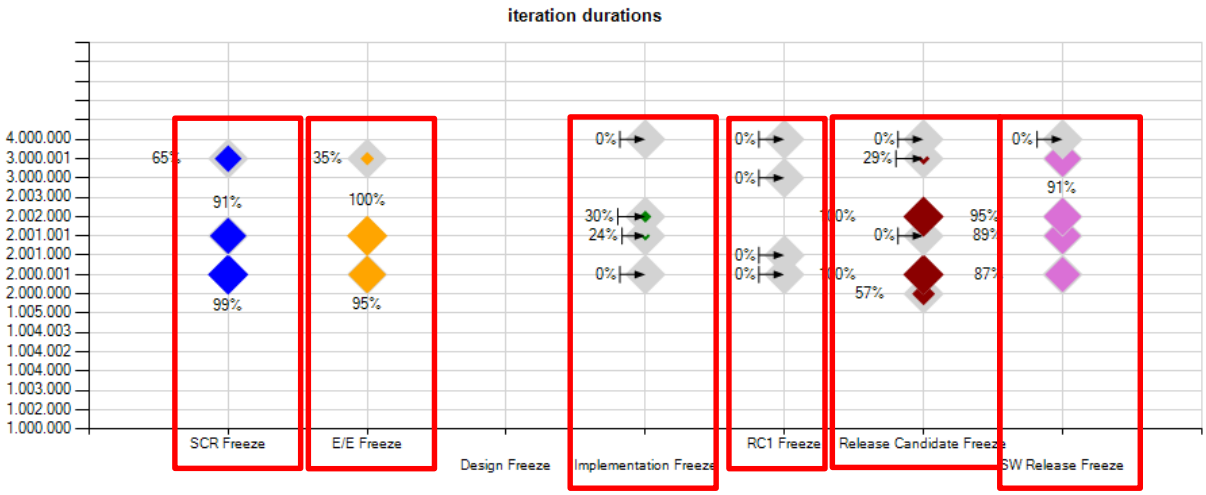


Abbildung 4.29: Iteration Durations, Anzeige *over freezes*, [6], [47]

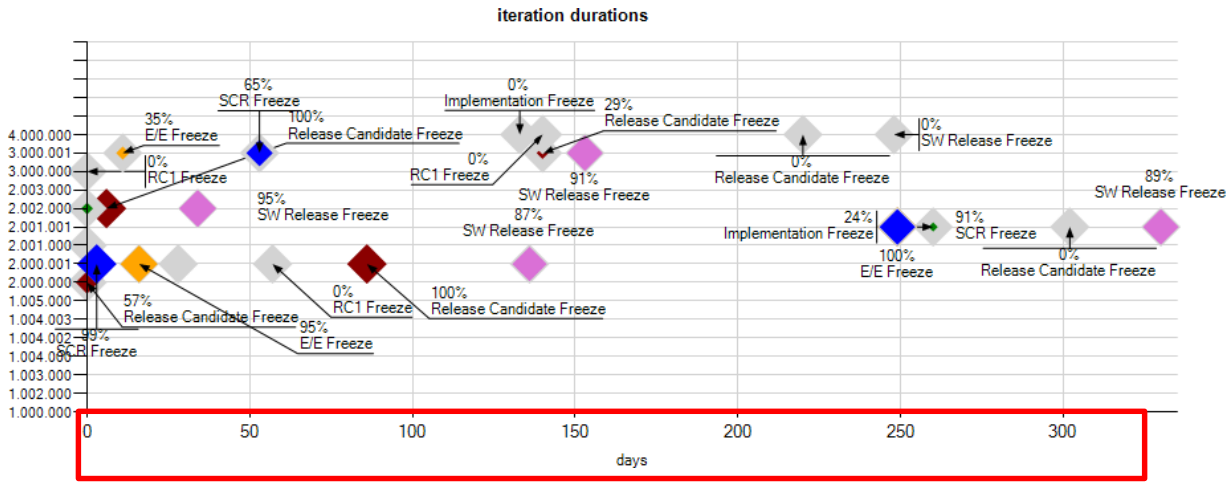


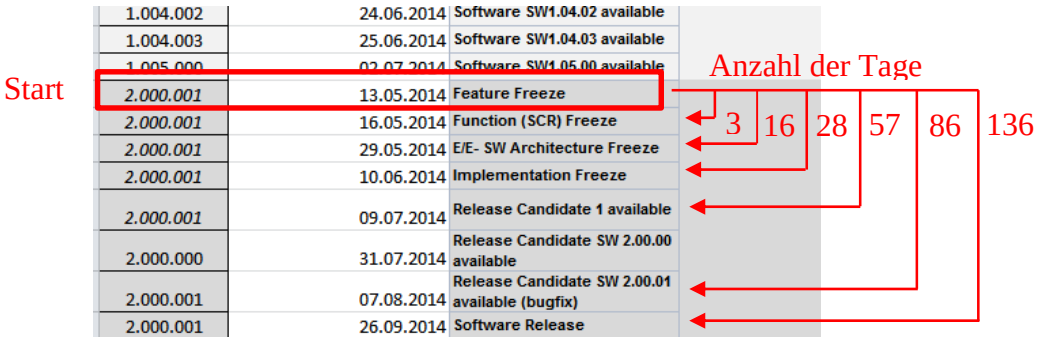
Abbildung 4.30: Iteration Durations, Anzeige *over time*, [6], [47]

4.7.1 Berechnungsmethodik/Algorithmus

4.7.1.1 Berechnung des Eintrages ___ -*Freeze datediff*

Zu jedem Software Release kann in der zweiten Ansicht die zeitliche Abfolge der Freezes im Diagramm angezeigt werden (*over time*). Die Zeitdifferenz (in der Tabelle __ -*Freeze datediff* gekennzeichnet) wird wie folgt berechnet, [6], [47]:

- Das früheste Datum pro Software Release bestimmt den Startzeitpunkt der Tage. Im Falle des Software Releases 2.000.001 ist das der 13.05.2014.
- Alle darauf folgenden Freezes ergeben sich aus der Datumsdifferenz ausgehend vom Startdatum, vergleiche Abbildung 4.31 mit den ersten Einträgen in Tabelle 4.5.



1.004.002	24.06.2014	Software SW1.04.02 available
1.004.003	25.06.2014	Software SW1.04.03 available
1.005.000	02.07.2014	Software SW1.05.00 available
2.000.001	13.05.2014	Feature Freeze
2.000.001	16.05.2014	Function (SCR) Freeze
2.000.001	29.05.2014	E/E- SW Architecture Freeze
2.000.001	10.06.2014	Implementation Freeze
2.000.001	09.07.2014	Release Candidate 1 available
2.000.000	31.07.2014	Release Candidate SW 2.00.00 available
2.000.001	07.08.2014	Release Candidate SW 2.00.01 available (bugfix)
2.000.001	26.09.2014	Software Release

Abbildung 4.31: Iteration Durations, Berechnung der Datumsdifferenz in der Meilensteinliste, [6], [47]

Tabelle 4.5: Iteration Durations, Berechnung der Datumsdifferenz, Ausschnitt aus der Tabelle, [6], [47]

sw release	project	SCR-Freeze c	SCR-Freeze c	SCR-Freeze p	E/E Architect	E/E Architect	E/E Architect
1.000.000	Beispielprojekt A						
1.002.000	Beispielprojekt A						
1.003.000	Beispielprojekt A						
1.004.000	Beispielprojekt A						
1.004.002	Beispielprojekt A						
1.004.003	Beispielprojekt A						
1.005.000	Beispielprojekt A						
2.000.000	Beispielprojekt A						
2.000.001	Beispielprojekt A	16.05.2014 0(	3	99	29.05.2014 0(	16	95
2.001.000	Beispielprojekt A						
2.001.001	Beispielprojekt A	21.11.2014 0(	249	91	21.11.2014 0(	249	100
2.002.000	Beispielprojekt A						
2.003.000	Beispielprojekt A						
3.000.000	Beispielprojekt A						
3.000.001	Beispielprojekt A	16.03.2015 0(	53	65	02.02.2015 0(	11	35
4.000.000	Beispielprojekt A						

4.7.1.2 Berechnung des *SCR-Freezes*

Für die Berechnung der *SCR-Freezes* wird die Meilensteinliste auf den Eintrag *Function (SCR) Freeze* in der Spalte *meilenstein* durchsucht und mit der *SCR*-Datenbank wie folgt für jeden Software-Release kombiniert, [6], [47]:

- *SCR Freeze percent* ergibt sich als Quotient in Prozent folgender aufsummierter *SCR* IDs.
- Als Dividend werden *SCR* IDs mit folgenden Eigenschaften aufsummiert:
  - Mit Status (*state* in der *SCR*-Datenbank) *Requirement Specified*
  - Vom Typ (*Requirements_Type* in der *SCR* Datenbank) *Functional* oder *Non-Functional*
  - Nur jene projektbezogene *SCR* IDs, die mit der *CI*-Datenbank via *Concerns* zum betreffenden Datum verbunden sind
- Als Divisor werden *SCR* IDs mit folgenden Eigenschaften aufsummiert:
  - Mit beliebigem Status
  - Vom Typ (*Requirements_Type* in der *SCR* Datenbank) *Functional* oder *Non-Functional*
  - Nur jene projektbezogene *SCR* IDs, die mit der *CI*-Datenbank via *Concerns* zum betreffenden Datum verbunden sind

4.7.1.3 Berechnung des *E/E-Architecture-Freezes*

Für die Berechnung der *E/E-Architecture-Freezes* wird die Meilensteinliste auf den Eintrag *E/E- SW Architecture Freeze* in der Spalte *meilenstein* durchsucht und mit der *SCR*-Datenbank wie folgt für jeden Software Release kombiniert, [6], [47]:

- *E/E Architecture Freeze percent* ergibt sich als Quotient in Prozent folgender aufsummierter *SCR* IDs
- Als Dividend werden *SCR* IDs mit folgenden Eigenschaften aufsummiert:
  - Mit Status (*state* in der *SCR*-Datenbank) *Requirement Specified*
  - Vom Typ (*Requirements_Type* in der *SCR* Datenbank) *Design* oder *Interface*

    - Nur jene projektbezogene *SCR* IDs, die mit der *CI*-Datenbank via *Concerns* zum betreffenden Datum verbunden sind
- Als Divisor werden *SCR* IDs mit folgenden Eigenschaften aufsummiert:
    - Mit beliebigem Status
    - vom Typ (*Requirements_Type* in der *SCR* Datenbank) *Design* oder *Interface*
    - Nur jene projektbezogene *SCR* IDs, die mit der *CI*-Datenbank via *Concerns* zum betreffenden Datum verbunden sind

4.7.1.4 Berechnung des *Design-Freezes*

Für die Berechnung der *Design-Freezes* wird die Meilensteinliste auf den Eintrag *Design Freeze* in der Spalte *meilenstein* durchsucht und mit der MR (*Module Requirements*)-Datenbank wie folgt für jeden Software-Release kombiniert, [6], [47]:

- *Design Freeze percent* ergibt sich als Quotient in Prozent folgender aufsummierter MR IDs
- Als Dividend werden MR IDs mit folgenden Eigenschaften aufsummiert:
    - Mit Status (*state* in der *MR*-Datenbank) *Requirement Specified*
    - Mit Parent (*Parent* in der *MR*-Datenbank) Einträgen
    - Nur jene projektbezogene MR IDs, die mit den Linktables „SCR_Child“ zum betreffenden Datum verbunden sind
- Als Divisor werden *MR* IDs mit folgenden Eigenschaften aufsummiert:
    - Mit beliebigem Status
    - Nur jene projektbezogene MR IDs, die mit Linktables „SCR_Child“ zum betreffenden Datum verbunden sind

4.7.1.5 Berechnung des *Implementation-Freezes*

Für die Berechnung der *Implementation-Freezes* wird die Meilensteinliste auf den Eintrag *Implementation Freeze* in der Spalte *meilenstein* durchsucht und mit der *SCR*-Datenbank wie folgt für jeden Software-Release kombiniert, [6], [47]:

- *Implementation-Freeze percent* ergibt sich als Quotient in Prozent folgender aufsummierter *SCR* IDs
- Als Dividend werden *SCR* IDs mit folgenden Eigenschaften aufsummiert:
    - Mit Status (*state* in der *SCR*-Datenbank) *Requirement Implemented*
    - Nur jene projektbezogene *SCR* IDs, die mit der *CI*-Datenbank via *Concerns* zum betreffenden Datum verbunden sind
- Als Divisor werden *SCR* IDs mit folgenden Eigenschaften aufsummiert:
    - Mit beliebigem Status
    - Nur jene projektbezogene *SCR* IDs, die mit der *CI*-Datenbank via *Concerns* zum betreffenden Datum verbunden sind

4.7.1.6 Berechnung des *RC1-Freezes*

Für die Berechnung der *RC1-Freezes* wird die Meilensteinliste auf den Eintrag *Release Candidate 1* in der Spalte *meilenstein* durchsucht und mit der *TC*-Datenbank wie folgt für jeden Software-Release kombiniert, [6], [47]:

- *RC1-Freeze percent* ergibt sich als Quotient in Prozent folgender aufsummierter *TC* IDs
- Als Dividend werden *TC* IDs mit folgenden Eigenschaften aufsummiert:

    - Mit Status (*state* in der *TC*-Datenbank) *TC Completed* oder *TC failed*
    - Der *Test Platform* (in der Datenbank mit *Test_Platform* gekennzeichnet) *MiL* oder *MiL and SiL* zugeordnet sind
    - Nur jene projektbezogene *TC* IDs, die mit *RelationshipTestIssue* aus den *CI-* , *SCR*- und *MR*-Datenbanken zum betreffenden Datum verbunden sind
- Als Divisor werden *TC* IDs mit folgenden Eigenschaften aufsummiert:
    - Mit beliebigem Status
    - Der *Test Platform* (in der Datenbank mit *Test_Platform* gekennzeichnet) *MiL* oder *MiL and SiL* zugeordnet sind
    - Nur jene projektbezogene *TC* IDs, die mit *RelationshipTestIssue* aus den *CI-*, *SCR*- und *MR*-Datenbanken zum betreffenden Datum verbunden sind

4.7.1.7 Berechnung des *Release Candidate-Freezes*

Für die Berechnung der *Release Candidate-Freezes* wird die Meilensteinliste auf den Eintrag *Release Candidate* (Jedoch nicht *Release Candidate 1* oder *Release Candidate 2*) in der Spalte *meilenstein* durchsucht und mit der *CI*-Datenbank wie folgt für jeden Software-Release kombiniert, [6], [47]:

- *Release Candidate-Freeze percent* ergibt sich als Quotient in Prozent folgender aufsummierter *CI* IDs
- Als Dividend werden *CI* IDs mit folgenden Eigenschaften aufsummiert:
    - Mit Status (*state* in der *CI*-Datenbank) *Change Completed* oder *Change Implemented*
    - Die Klassifikation (*classification* in der *CI*-Datenbank) „*Finding*“ aufweisen
    - Den gleichen Software Release (*Planned_SW_Release_String* in der *CI*-Datenbank) wie jene aus der Meilensteinliste beinhalten
- Als Divisor werden *CI* IDs mit folgenden Eigenschaften aufsummiert:
    - Mit beliebigem Status
    - Die Klassifikation (*classification* in der *CI*-Datenbank) *Finding* aufweisen
    - Den gleichen Software Release (*Planned_SW_Release_String* in der *CI*-Datenbank) wie jene aus der Meilensteinliste beinhalten

4.7.1.8 Berechnung des *Software Release-Freezes*

Für die Berechnung der *Software Release-Freezes* wird die Meilensteinliste auf den Eintrag *Software Release* in der Spalte *meilenstein* durchsucht und mit der *TC*-Datenbank wie folgt für jeden Software-Release kombiniert, [6], [47]:

- *SW Release-Freeze percent* ergibt sich als Quotient in Prozent folgender aufsummierter *TC* IDs
- Als Dividend werden *TC* IDs mit folgenden Eigenschaften aufsummiert:
    - Mit Status (*state* in der *TC*-Datenbank) *TC Completed* oder TC failed
    - Der *Test Platform* (in der Datenbank mit *Test_Platform* gekennzeichnet) *HiL* zugeordnet sind
    - Nur jene projektbezogene *TC* IDs, die mit *RelationshipTestIssue* aus den *CI-*, *SCR*- und *MR*-Datenbanken zum betreffenden Datum verbunden sind
- Als Divisor werden *TC* IDs mit folgenden Eigenschaften aufsummiert:
    - Mit beliebigem Status
    - Der *Test Platform* (in der Datenbank mit *Test_Platform* gekennzeichnet) *HiL* zugeordnet sind

- Nur jene projektbezogene *TC* IDs. die mit *RelationshipTestIssue* aus den *CI*-, *SCR*- und *MR*-Datenbanken zum betreffenden Datum verbunden sind

4.7.2 Beispiel zur Berechnung

Anhand eines Beispiels soll die Vorgehensweise für die Handhabung dieses KPIs verdeutlicht werden. In der *settings*-Bar werden folgende Einstellungen getroffen, [6], [47]:

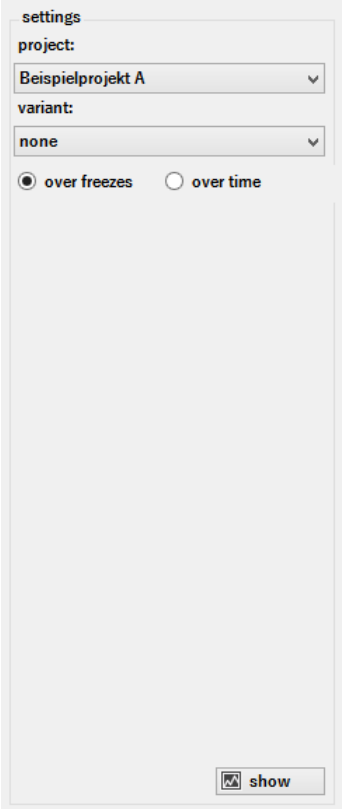


Abbildung 4.32: Iteration Durations, Benutzereingabe, [6], [47]

Als Ergebnis werden pro Freeze Rauten angezeigt, die den Erfüllungsgrad zu einem Software Release illustrieren. Je größer die farbigen Rauten im Verhältnis zum grauen 100% Vergleichsmarker sind, desto höher ist der Erfüllungsgrad in Prozent. In Abbildung 4.29 ist ersichtlich, dass sich in der Meilensteinliste kein *Design Freeze* befindet und somit die Spalte ausgelassen wird. Das zweite Diagramm erhält man durch Anwählen des *over time*-Buttons, siehe Abbildung 4.30. Die Tabelle, sowie die prozentualen Erfüllungsgrade ändern sich hierbei nicht, es werden jedoch die Freeze Points zeitlich fortlaufend im Diagramm aufgetragen. Die Diagramme zu diesem Beispiel sind analog zu Abbildung 4.29 und Abbildung 4.30 in der Einleitung, [6], [47].

4.8 KPI: Software Tracking, [6], [47]

Beantwortete Fragestellungen:

Weitere Fragestellungen

- Wie hoch ist der Verknüpfungsgrad von Items?
- Wie viele Reviews wurden durchgeführt?

Dieser KPI widmet sich dem Verknüpfungsgrad von Anforderungen auf System- und Modulebene. Im PTC MKS-System können Planungselemente durch Verknüpfungen auf unterschiedliche Ebenen oder Äste im V-Modell verweisen. Dies geschieht wie schon in Kapitel 3 beschreiben durch Einträge wie *RelationshipTestIssue*, *Concerns* oder *Parent* des betroffenen Planungselements. Die Quantifizierung dieser Verweise ist besonders im Interesse der Qualitätssicherung, da nun Aussagen über die Verweisqualität der projektspezifischen Planungselemente getroffen werden können. Es werden in den Diagrammen Anforderungen mit und ohne den jeweiligen Verweis gegenübergestellt und pro Systemfreigabe gruppiert. Da dieser KPI sehr umfangreich ist, wurde der Diagrammbereich in drei Abschnitte unterteilt, vergleiche Abbildung 4.33. Zusätzlich ist noch ein Filter nach *Sub Components* implementiert, falls beispielsweise nur Planungselemente des *„Boot Loaders* benötigt werden (auch Mehrfachauswahl erlaubt), [6], [47].

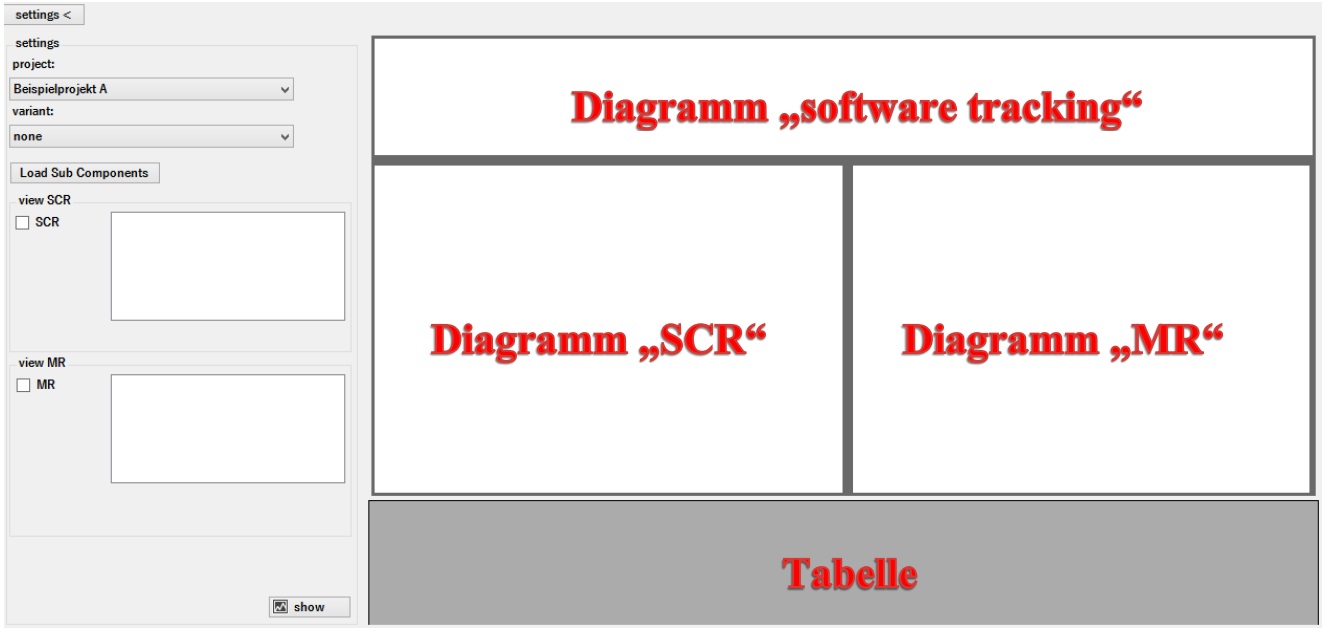


Abbildung 4.33: Software Tracking, GUI, [6], [47]

4.8.1 Diagramm „software tracking“

Pro Systemfreigabe des ausgewählten Projektes werden für *SCR* und MR folgende Balkendiagramme angezeigt, [6], [47]:

- *target SCR* oder *target MR*: Anzahl aller *SCRs*/*MRs* (Status beliebig)
- *performance SCR* oder *performance MR*: Anzahl abgeschlossener *SCRs*/*MRs* (Status *Requirement Implemented*)

Die Berechnungsmethodik dieser Einträge wird im folgenden Abschnitt erklärt.

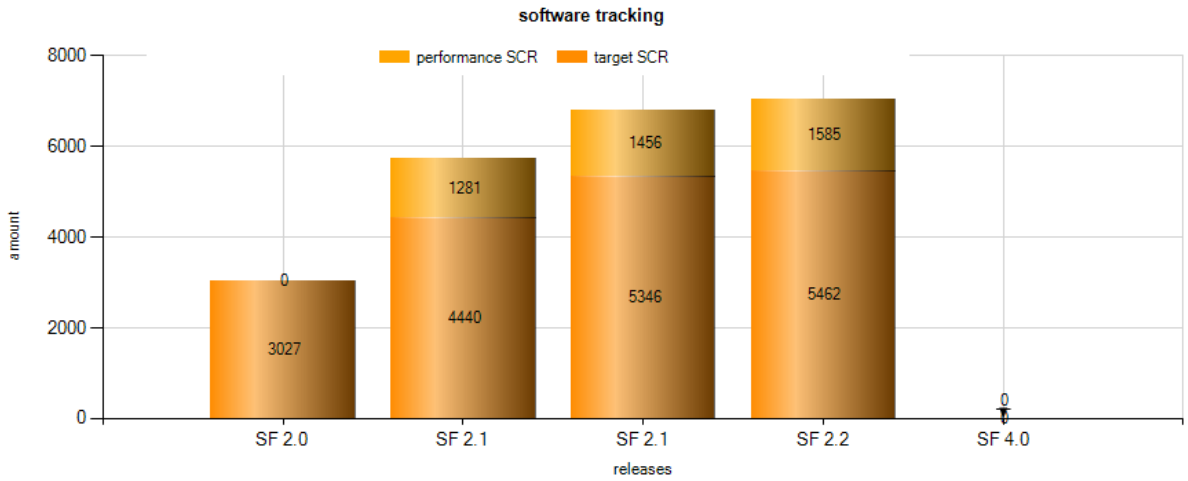


Abbildung 4.34: Software Tracking, Output Diagramm "software tracking", [6], [47]

4.8.2 Diagramm *SCR* bzw. *MR*

Die beiden unteren Diagramme illustrieren pro Systemfreigabe („SF“) vier Balken mit folgenden Eigenschaften, [6], [47]:

- *review not done* bzw. *review done*: Gegenüberstellung der Anzahl jener *SCRs*/*MRs* ohne Einträge in der *review done*-Spalte mit der Anzahl jener *SCRs*/*MRs* mit vorhandenen Einträgen in der *review done*-Spalte
- *empty relationship test issues*bzw. „relationship test issues“: Gegenüberstellung der Anzahl jener *SCRs*/*MRs* ohne Einträge in der *RelationshipTestIssue*-Spalte mit der Anzahl jener *SCRs*/*MRs* mit vorhandenen Einträgen in der *RelationshipTestIssue*-Spalte
- „empty parent entries“ bzw. „parent entries“: Gegenüberstellung der Anzahl jener *SCRs*/*MRs* ohne Einträge in der *Parent*-Spalte mit der Anzahl jener *SCRs*/*MRs* mit vorhandenen Einträgen in der *Parent*-Spalte
- Nur für das *SCR*-Diagramm relevant: *empty child entries* bzw. *child entries*: Gegenüberstellung der Anzahl jener *SCRs* ohne Einträge in der *Child*-Spalte mit der Anzahl jener *SCRs* mit vorhandenen Einträgen in der *Child*-Spalte

Die Berechnungsmethodik dieser Einträge wird im unteren Abschnitt erklärt.

Abbildung 4.35 zeigt die gruppierte Darstellung der Balken pro Systemfreigabe am Beispiel des *SCR*-Diagramms.

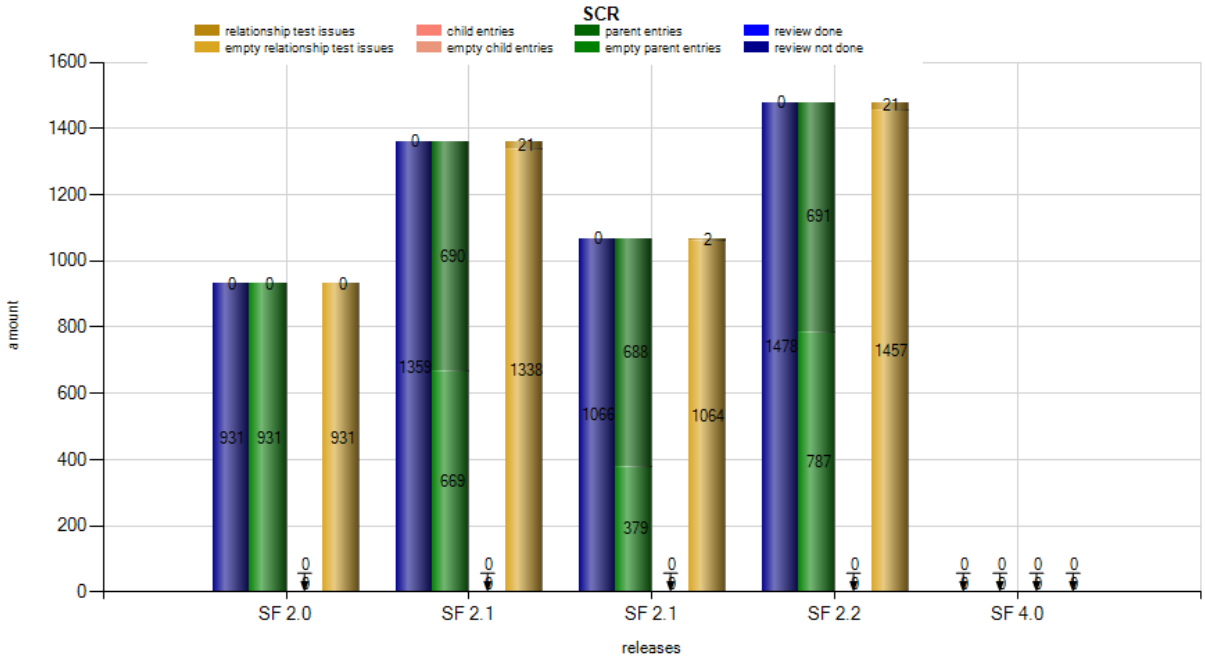


Abbildung 4.35: Software Tracking, Diagramm *SCR/MR*, [6], [47]

4.8.3 Tabelle zu Software Tracking

Die Tabelle zeigt pro Systemfreigabe alle im Diagramm benötigten Informationen zusätzlich numerisch nochmals an, [6], [47]:

- *release*: alle Systemfreigaben des Projektes
- *date*: Datum der jeweiligen Systemfreigabe
- *target SCR* bzw. *target MR*: siehe oben
- *performance SCR* bzw. *performance MR*: siehe oben
- *degree of fulfilment SCR* bzw. *degree of fulfilment MR*: Quotient in Prozent aus *performance SCR* bzw. *performance MR* und „ *target SCR* bzw. *target MR*
- alle anderen Spalteneinträge (*empty parent entries*, *parent entries*, usw.): siehe oben

release	date	target SCR	performance SCR	degree of fulfilment SCR	empty parent entries SCR
SF 2.0	27.10.2014 00:0...	931	0	0	931
SF 2.1	12.05.2015 00:0...	1359	192	14	669
SF 2.1	16.02.2015 00:0...	1100	17	2	379
SF 2.2	26.06.2015 00:0...	1478	307	21	787
SF 4.0	25.03.2016 00:0...	0	0	0	0

Abbildung 4.36: Software Tracking, Tabelle, [6], [47]

4.8.4 Berechnungsmethodik/Algorithmus

4.8.4.1 Berechnung von *target SCR oder target MR*

Die Anzahl an *target SCR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, welche zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-

Box) enthalten und mit den *CIs* via *Concerns* miteinander verbunden sind.

Die Anzahl an *target MR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, die zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-Box) enthalten und mit den *SCRs* via *Child* miteinander verbunden sind, [6], [47].

4.8.4.2 Berechnung von *performance SCR* oder *performance MR*

Die Anzahl an *performance SCR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, welche zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-Box) enthalten, mit den *CIs* via *Concerns* verbunden sind und sich im Status *Requirement Implemented* befinden.

Die Anzahl an *performance MR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, die zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-Box) enthalten, mit den *SCRs* via *Child* verbunden sind und sich im Status *Requirement Implemented* befinden, [6], [47].

4.8.4.3 Berechnung von *review done SCR* oder *review done MR*

Die Anzahl an *review done SCR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, welche zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-Box) enthalten, mit den *CIs* via *Concerns* verbunden sind und Einträge in der *ReviewDone*-Spalte besitzen.

Die Anzahl an *review done MR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, die zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-Box) enthalten, mit den *SCRs* via *Child* verbunden sind und Einträge in der *ReviewDone*-Spalte besitzen, [6], [47].

4.8.4.4 Berechnung von *review not done SCR* oder *review not done MR*

Wie oben beschrieben, mit dem Unterschied, dass die leeren Einträge der *ReviewDone*-Spalte gezählt werden (gilt für *SCRs* und MRs), [6], [47].

4.8.4.5 Berechnung von *relationship test issues SCR* oder *relationship test issues MR*

Die Anzahl an *relationship test issues SCR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, welche zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-Box) enthalten, mit den *CIs* via *Concerns* verbunden sind und Einträge in der *RelationshipTestIssue*-Spalte besitzen.

Die Anzahl an *relationship test issues MR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, welche zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-Box) enthalten, mit den *SCRs* via *Child* verbunden sind und Einträge in der *RelationshipTestIssue*-Spalte besitzen, [6], [47].

4.8.4.6 Berechnung von *empty relationship test issues SCR* oder *empty relationship test issues MR*

Wie oben beschrieben, mit dem Unterschied, dass die leeren Einträge der *RelationshipTestIssue*-Spalte gezählt werden (gilt für *SCRs* und *MRs*), [6], [47].

4.8.4.7 Berechnung von *parent entries SCR* oder *parent entries MR*

Die Anzahl an *parent entries SCR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, welche zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-Box) enthalten, mit den *CIs* via *Concerns* verbunden sind und Einträge in der *Parent*-Spalte besitzen.

Die Anzahl an *parent entries MR* Einträgen ergibt sich durch das Aufsummieren all jener IDs, die zur betreffenden Systemfreigabe die angewählten *Sub Components*-Einträge (in der *settings*-Box) enthalten, mit den *SCRs* via *Child* verbunden sind und Einträge in der *Parent*-Spalte besitzen, [6], [47].

4.8.4.8 Berechnung von *empty parent entries SCR* oder *empty parent entries MR*

Wie oben beschrieben, mit dem Unterschied, dass die leeren Einträge der *Parent*-Spalte gezählt werden. (gilt für *SCRs* und *MRs*), [6], [47].

4.8.5 Beispiel zur Berechnung

Anhand eines Beispiels soll die Vorgehensweise für die Handhabung dieses KPIs verdeutlicht werden. In der *settings*-Bar werden folgende Einstellungen getroffen, [6], [47]:

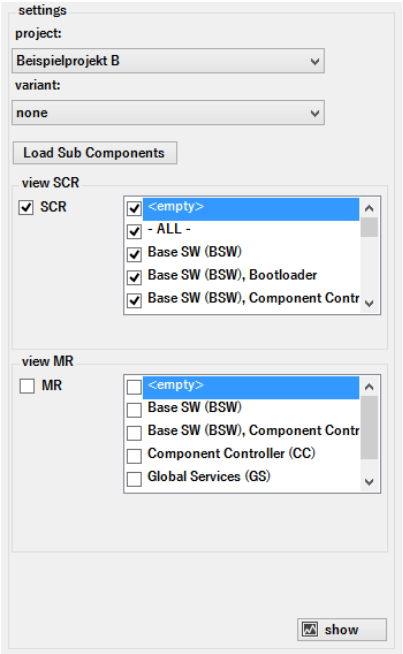


Abbildung 4.37: Software Tracking, Benutzereingabe, [6], [47]

Hinweis: Bevor die Berechnung gestartet werden kann, müssen die *Sub Components* mit dem Button „Load Sub Components“ geladen werden.

Button *SCR* anwählen mit folgenden *Sub Components*, [6], [47]:

- *<empty>* (leere Einträge in den Sub Components)
- *-ALL-*
- *Base SW (BSW)*
- *Base SW (BSW), Bootloader*
- *Base SW (BSW), Component Controller (CC)*

- *Base SW (BSW), Component Controller (CC), Runtime Environment (RTE)*
- *Base SW (BSW), Component Controller (CC), Vehicle Controller (VC)*

Der Berechnungsvorgang wird mit dem *show*-Button gestartet. Die Balkendiagramme und die Tabelle sind in Abbildung 4.39 ersichtlich und können zu besseren Ansicht beliebig vergrößert und verkleinert werden. Als Beispiel soll hier die erste Systemfreigabe *SF 2.1* näher betrachtet werden, [6], [47]:

4.8.5.1 Diagramm „software tracking“

Von 1359 *SCR*-IDs sind 192 im Status *Requirement Implemented*

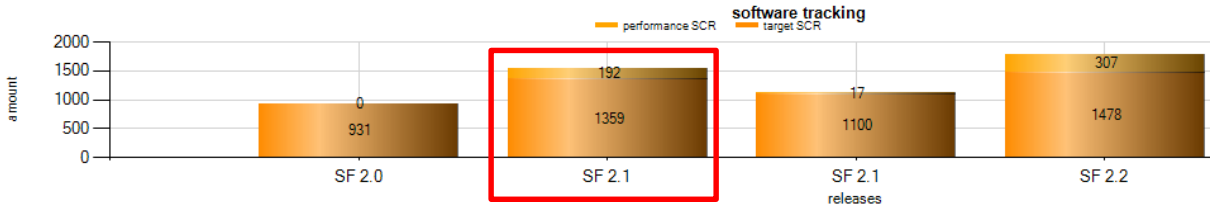
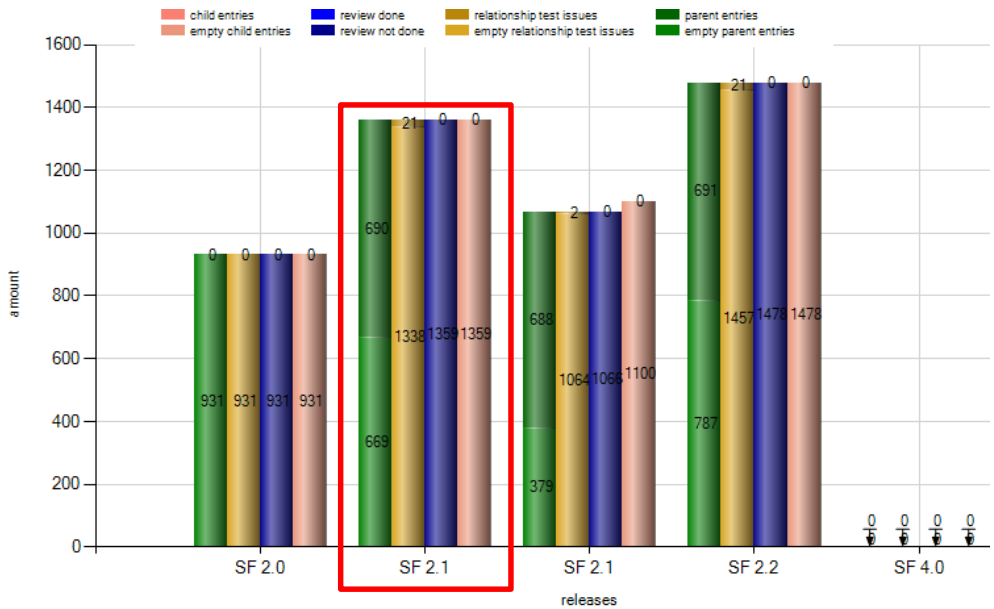


Abbildung 4.38: Software Tracking, Vergleich der abgeschlossenen Anforderungen, [6], [47]

4.8.5.2 Diagramm „SCR“ , [6], [47]

- 1338 *SCR*-IDs ohne (*empty relationship test issues*), 21 *SCR*-IDs mit *relationship test issue*-Einträgen (*relationship test issues*)
- 669 *SCR*-IDs ohne (*empty parent entries*), 690 *SCR*-IDs mit *Parent*-Einträgen *(parent entries*)
- 1359 *SCR*-IDs ohne (*review not done*), 690 *SCR*-IDs mit *review done*-Einträgen (*review done*)
- 1359 *SCR*-IDs ohne (*empty child entries*), 0 *SCR*-IDs mit *Child*-Einträgen (*child entries*)



SF 2.0	27.10.2014 00:0...	931	0	0	931	0	0	931	0
SF 2.1	12.05.2015 00:0...	1359	192	14	669	690	21	1338	0
SF 2.1	16.02.2015 00:0...	1100	17	2	379	688	2	1064	0

Abbildung 4.39: Software Tracking, Anzahl der Planungselemente mit Verweis, [6], [47]

5 Analyse der Ergebnisse von Entwicklungs- und Testdaten

Die nachfolgende Analyse von Beispielprojekten soll exemplarisch anhand von verschiedenen KPIs durchgeführt werden und deren Verknüpfung zueinander veranschaulichen. Dazu werden markante Ausschläge und Auffälligkeiten in Diagrammen untersucht und deren mögliche Ursache erläutert.

5.1 Vergleich von Projekten mit der Product-Platform

Für dieses Fallbeispiel werden die KPIs **„Status of planning elements 2“** und **„Trend of ratios“** benötigt. Der KPI „Trend of ratios“ wird mit *Completed Findings* einmal für die */ProductPlatform*, siehe Abbildung 5.1, und einmal für das *Beispielprojekt A*, Abbildung 5.2, ausgeführt. Der markante Abfall der relativen Häufigkeiten der abgeschlossenen *Findings* in der */ProductPlatform* und der korrelierende Anstieg der abgeschlossenen *Findings* des Beispielprojektes im Oktober 2013 lassen darauf schließen, dass einige noch nicht abgeschlossene Arbeitspakete verschoben worden sind.

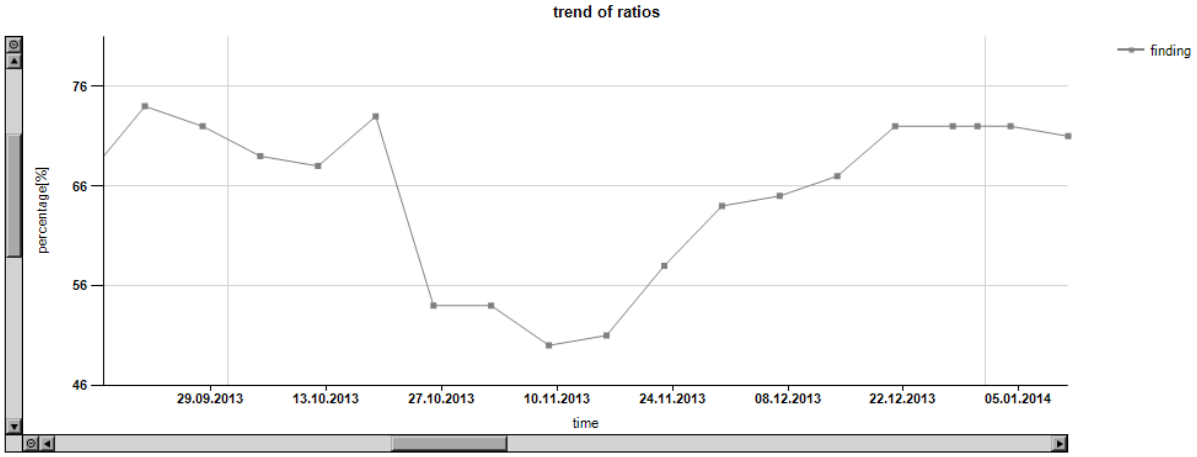


Abbildung 5.1: Vergleich von Projekten mit der Product-Platform, Verlauf 1, [6], [47], vgl. [48], [49]

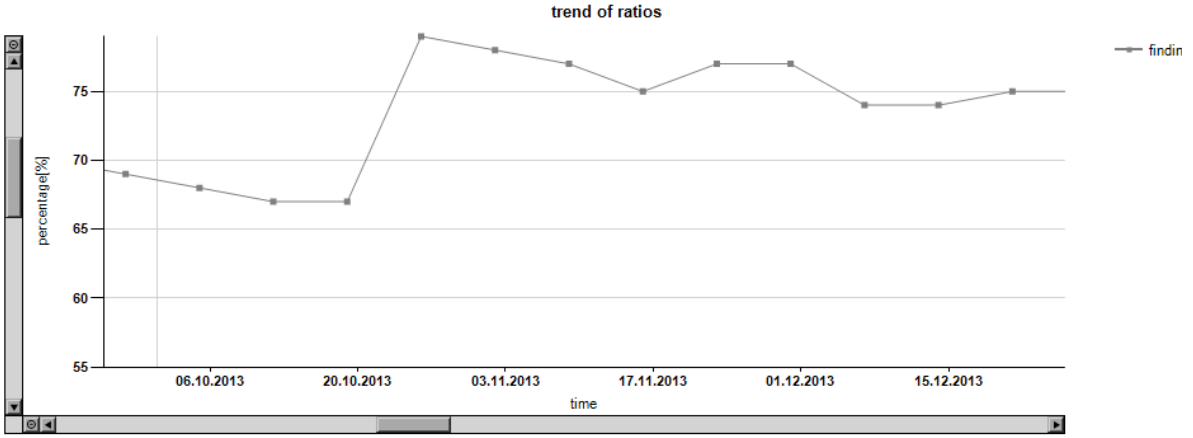


Abbildung 5.2: Vergleich von Projekten mit der Product-Platform, Verlauf 2, [6], [47], vgl. [48], [49]

In der Tabelle von „Status of planning Elements 2“ (mit der Auswahl *Beispielprojekt A* für den

Monat Oktober 2013) ist in der Spalte *last record date* bei einigen Arbeitspaketen der 23.10.2013 genannt, obwohl diese noch nicht abgeschlossen sind. Dies ist ein weiteres Indiz für die Verschiebung einiger Arbeitspakete in die *\ProductPlatform*, siehe Abbildung 5.3.

item id	project	variant	created date	classification	planned SW	completion S	assigned user	defined prio	target date	SIL	State	last record date
360157	Beispielprojekt A		08.10.2013 1	Finding	3.020.060	SW	Max Mustermann	Hi			Change New	23.10.2013 00:00:00
360352	Beispielprojekt A		08.10.2013 1	Finding	3.020.040	3.020.040	Max Mustermann	Med		No	Change Implemented	31.10.2013 00:00:00
360374	Beispielprojekt A		09.10.2013 1	Finding	3.020.000	SW	Max Mustermann	Med			Change New	23.10.2013 00:00:00
360421	Beispielprojekt A		10.10.2013 1	Finding	3.020.040	3.020.040	Max Mustermann	Med		A	Change Implemented	23.10.2013 00:00:00
360781	Beispielprojekt A		14.10.2013 0	Finding	3.020.020	SW	Max Mustermann	Highest			Change New	20.10.2013 00:00:00
360831	Beispielprojekt A		14.10.2013 1	Finding	3.020.040	3.020.040	Max Mustermann	Med		A	Change Implemented	23.10.2013 00:00:00
360843	Beispielprojekt A		14.10.2013 1	Finding	3.020.000	SW	Max Mustermann	Med			Change New	31.10.2013 00:00:00
360911	Beispielprojekt A		14.10.2013 1	Finding	3.020.000	SW	Max Mustermann	Med			Change New	23.10.2013 00:00:00
360938	Beispielprojekt A		15.10.2013 1	Finding	SW	SW	Max Mustermann	Med	21.11.2013		Change New	31.10.2013 00:00:00
362764	Beispielprojekt A		15.10.2013 1	Finding	SW	SW	Max Mustermann	Med	13.11.2013		Change New	31.10.2013 00:00:00
362812	Beispielprojekt A		16.10.2013 0	Finding	3.020.000	SW	Max Mustermann	Med			Change New	23.10.2013 00:00:00

Abbildung 5.3: Vergleich von Projekten mit der Product-Platform 3, [6], [47], vgl. [48], [49]

Dieser markante Ausschlag zu diesem Datum kann vermieden werden, indem die Arbeitspakete mit der Variante *Beispielprojekt A* versehen werden bevor man sie in die */ProductPlatform* verschiebt. Dadurch können Arbeitspakete, die sich bereits in der */ProductPlatform* befinden, trotzdem durch Anwählen der Variante *Beispielprojekt A* in den KPIs berücksichtigt werden.

5.2 Untersuchung von Testfällen

Das hier zu untersuchende Fallbeispiel benötigt die KPIs **„Target performance comparison“** und **„Amount of all states (work/done)“**. Der KPI „Target performance comparison” wird mit der Auswahl *Beispielprojekt A* für Projekt und Variante ausgeführt. Auffällig ist der markante Unterschied der Balkenlänge zwischen den Systemfreigaben *SF 3/4* und *SF 2.x*. Die abgeschlossenen Testfälle gehen rapide zurück und steigen zur nächsten Systemfreigabe wieder an, vergleiche Abbildung 5.4.

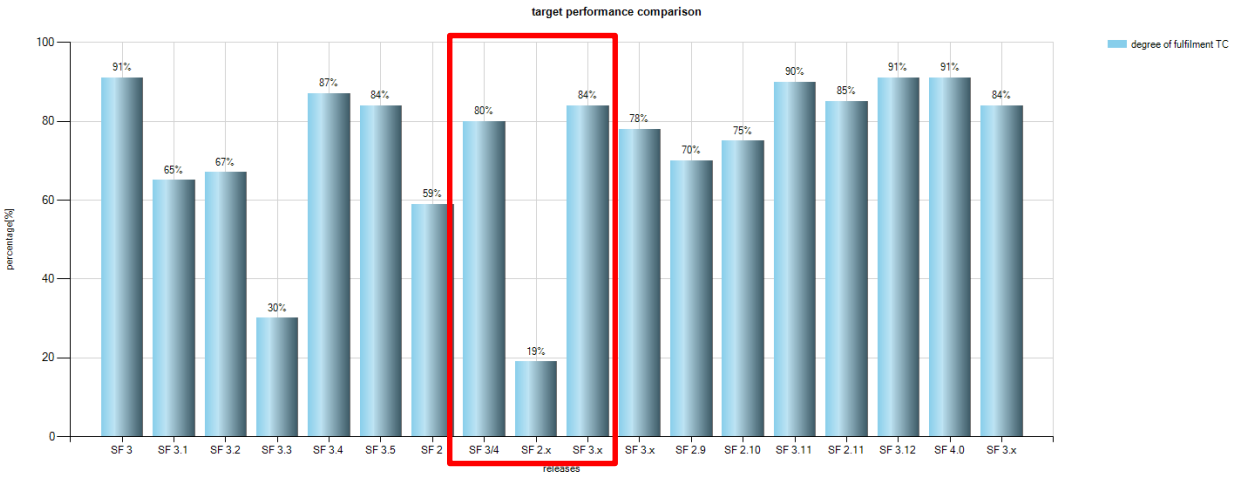
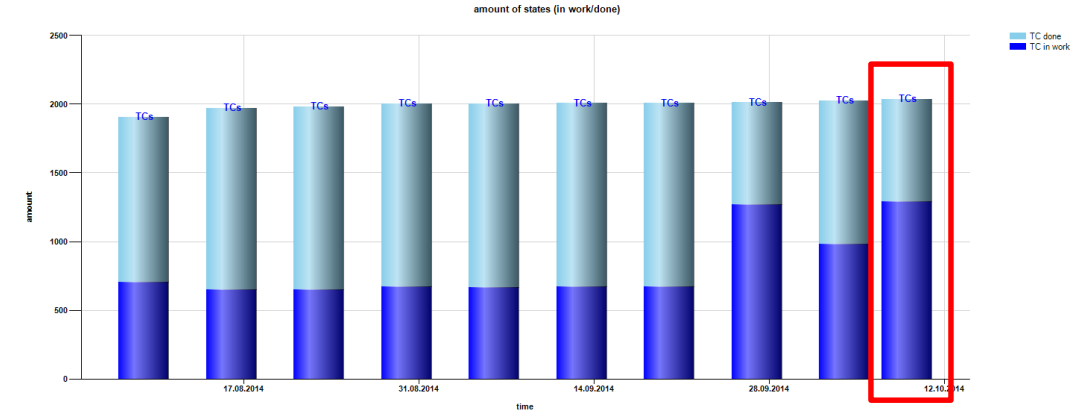


Abbildung 5.4: Untersuchung von Testfällen „Target Performance Comparison“, [6], [47], vgl. [48], [49]

Um einen Zusammenhang dieser Auffälligkeit mit anderen KPIs herzustellen wird der KPI „Amount of all states (work/done)“ im Zeitraum vom 07.08.2014 bis 09.10.2014 ausgeführt, siehe Abbildung 5.5. Am 09.10.2014 (Datum der Systemfreigabe *SF 2.x*) wird die Bearbeitung eines erheblichen Anteils von Testfällen (knapp 300) wiederaufgenommen. Daraus resultiert daher der markante Abfall im vorigen KPI „Target performance comparison“. Zur Systemfreigabe *SF 2.x* mussten eine Vielzahl von Testfällen noch einmal wiederholt werden

und wurden daher von *TC Completed* auf *TC Retest* gesetzt. Nach Tabelle 4.1 zählt *TC Retest* zu den nicht abgeschlossenen Testfällen (*TC in work*) und deshalb wird die relative Länge des dunkelblauen Balkens zum 09.10.2014 erhöht.



snapshot_date	completed TCs	all TCs
09.08.2014	1202	703
16.08.2014	1321	650
23.08.2014	1334	648
30.08.2014	1334	670
06.09.2014	1336	668
13.09.2014	1336	671
20.09.2014	1337	670
27.09.2014	744	1269
04.10.2014	1044	981
09.10.2014	747	1289

Abbildung 5.5: Untersuchung von Testfällen, „Amount of States in work/done“, [6], [47], vgl. [48], [49]

Ähnlicher Sachverhalt tritt auch zwischen Systemfreigabe *SF 3.1* (10.10.2013) und *SF 3.3* (11.03.2014) auf. Auch hier werden in einer Woche mehrere hundert Testfälle wieder neu bearbeitet, vergleiche Abbildung 5.6.

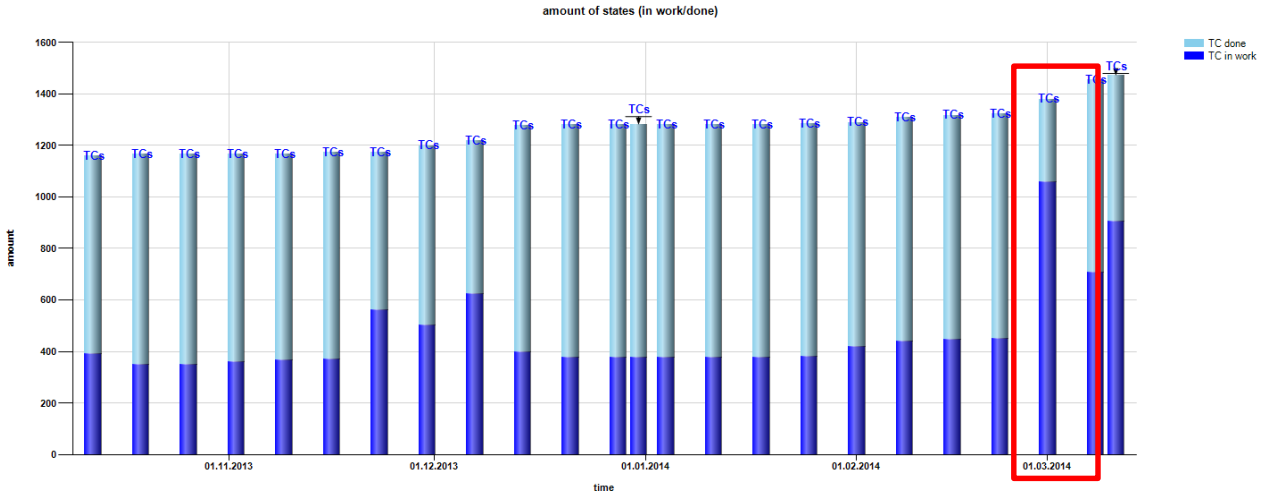
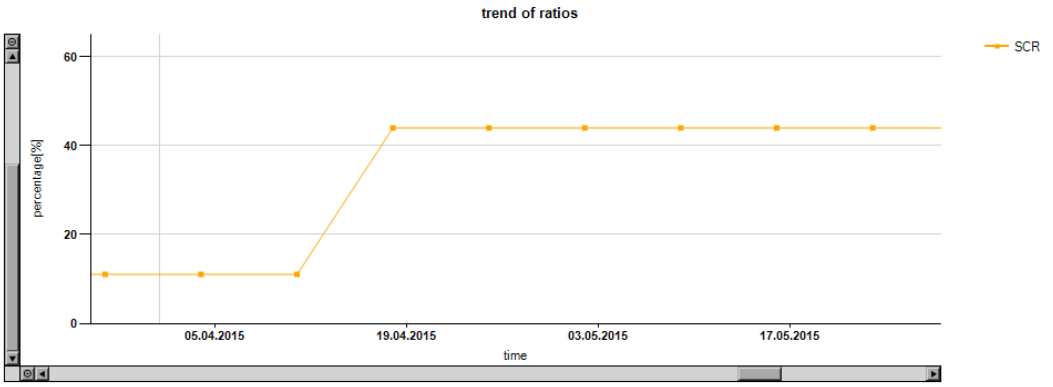


Abbildung 5.6: Untersuchung von Testfällen „Amount of States in work/done“, [6], [47], vgl. [48], [49]

5.3 Zeitlicher Verlauf der Anforderungen

Hierfür wird der KPI **„Trend of ratios“** benötigt, mit dem der Verlauf der Anforderungen (*SCRs*) des Beispielprojektes A inklusive dessen Variante aus der */ProductPlatform* untersucht wird, siehe Abbildung 5.7. Erwähnenswert ist hierbei der nicht stetige Verlauf zwischen

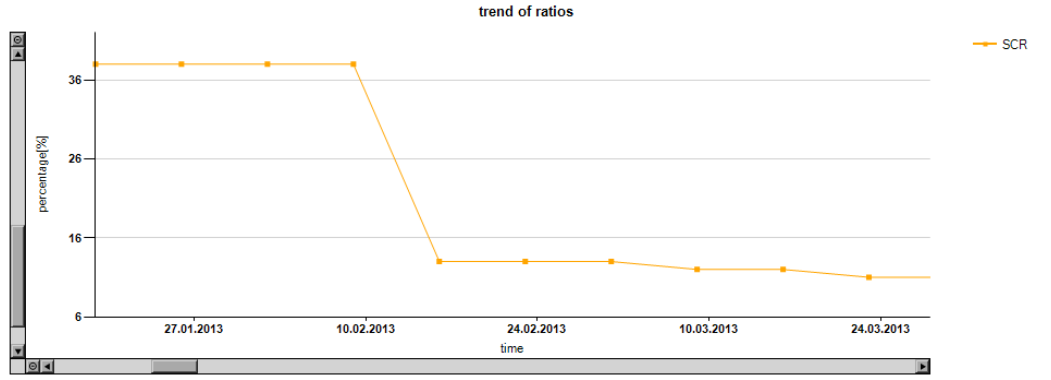
11.04.2015 und 18.04.2015. Bei näherer Betrachtung der zugehörigen Tabelle fällt auf, dass knapp 1200 Anforderungen in dieser Woche implementiert worden sind. Die Ursache für die Vielzahl an implementierten Anforderungen in dieser Woche ist die geplante Systemfreigabe im April 2015. Es ist daher darauf zu achten, implementierte Anforderungen kontinuierlich im ALM-System einzutragen, um solche Unstetigkeit zu vermeiden.



snapshot_date	implemented SCRs	all SCRs	percentage SCRs
21.03.2015	383	3547	11
28.03.2015	386	3547	11
04.04.2015	386	3547	11
11.04.2015	386	3547	11
18.04.2015	1564	3547	44
25.04.2015	1564	3547	44
02.05.2015	1564	3547	44

Abbildung 5.7: Zeitlicher Verlauf von Anforderungen 1, [6], [47], vgl. [48], [49]

Ein vergleichbares Muster ist in der Woche zwischen 09.02.2013 und 16.02.2013 zu verzeichnen. Dies ist dadurch begründet, dass sehr viele neue Anforderungen (*SCRs* im Status *Requirement New*) hinzukommen und deshalb der prozentuale Wert nach unten verläuft.



snapshot_date	implemented SCRs	all SCRs	percentage SCRs
26.01.2013	38	99	38
02.02.2013	38	100	38
09.02.2013	38	101	38
16.02.2013	48	381	13
23.02.2013	48	381	13

Abbildung 5.8: Zeitlicher Verlauf von Anforderungen 2, [6], [47], vgl. [48], [49]

Eine vergleichbare Sachlage ist für den KPI „**Target performance comparison**“ für *Beispielprojekt A* zu erkennen. Hierbei gliedern sich – im Gegensatz zum KPI **„Trend of ratios“** – die abgeschlossenen Anforderungen auf der Abszisse nach zeitlich aufsteigenden Software Releases. Zusätzlich werden nur verlinkte Anforderungen ausgehend von den *Change*

Issues betrachtet. Dennoch fällt auf, dass unter Einbeziehung der */ProductPlatform* mit Variante *Beispielprojekt A* die anfangs gute Performance von 80-100% rapide auf die Hälfte abfällt, siehe Abbildung 5.9. Grund dafür sind die in der */ProductPlatform* neu hinzugefügten Anforderungen zwischen Systemfreigabe 3.5 und Systemfreigabe 2. Wird für diesen KPI nur das *Beispielprojekt A* ohne Product-Platform betrachtet, setzt sich die gute Performance in der Größenordnung von 80-100% bis zur letzten Systemfreigabe *SF 3.x* fort.

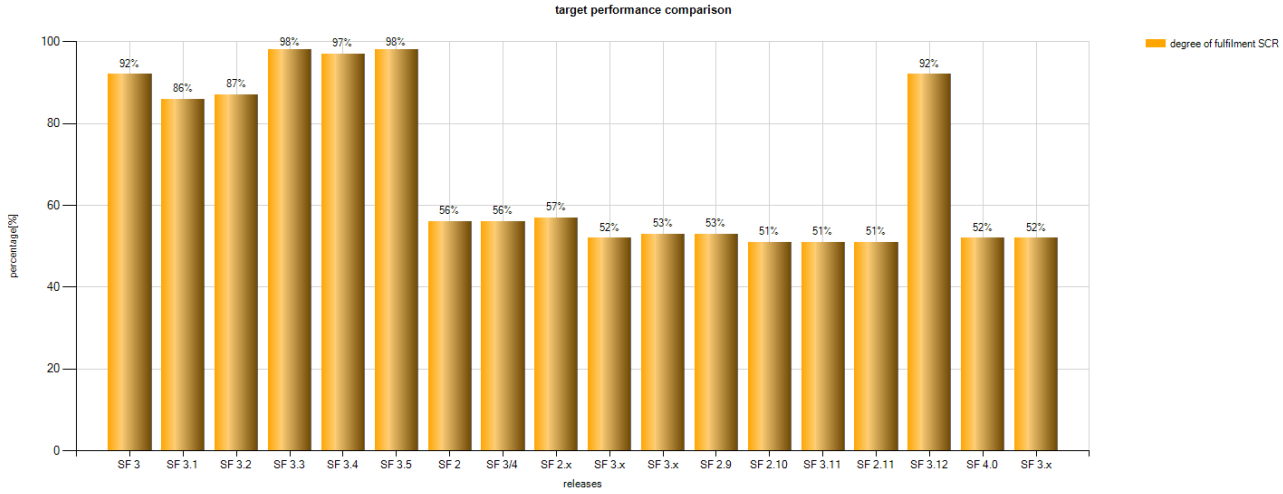


Abbildung 5.9: Verlauf der *SCRs* über Software Releases, [6], [47], vgl. [48], [49]

5.4 Zeitverzug von Arbeitspaketen

In diesem Beispiel soll das Augenmerk auf einzelne Arbeitspakete gelegt werden, weshalb sich der KPI „**Status of planning elements 2**“ dafür eignet. Wiederum wird das *Beispielprojekt A* ausgewählt und der Verlauf in der GUI visualisiert. In der Tabelle sind die Zeitverzögerungen illustriert, vergleiche Kapitel 4.2. Arbeitspakete, die über mehrere Monate nicht bearbeitet wurden – wie beispielsweise das Arbeitspaket mit der ID-Nummer 382749 – sind hierbei ersichtlich. Die geplante Fertigstellung des Arbeitspaketes wurde mit 02.08.2013 veranschlagt, tatsächlich passierte die Implementierung des *CIs* (Status *Change Implemented*) im System aber erst zum 11.06.2014, was zu einer hohen Verzögerung von 313 Tagen (*time lag*) führte. Berechnet wird dieser Verzug aus der Differenz von *Target Date*, der als großes blaues Quadrat in Abbildung 5.10 markiert wird, zum erstem Tag, an dem das Arbeitspaket erstmalig in den Status *Change Implemented* gesetzt wurde.

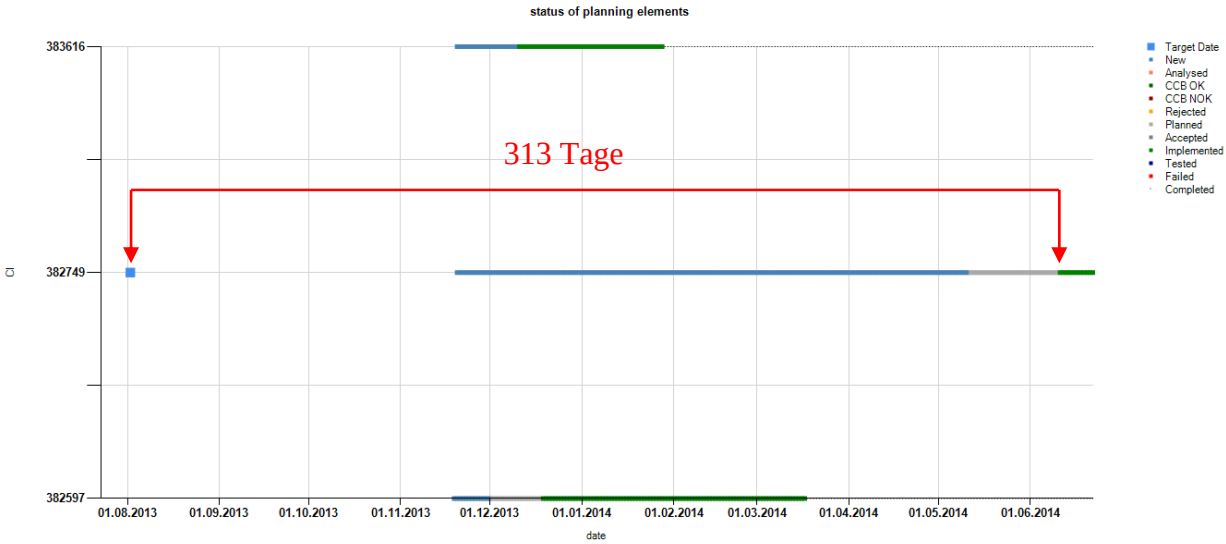


Abbildung 5.10: Zeitverzug von Arbeitspaketen, [6], [47], vgl. [48], [49]

Betrachtet man jedoch das geplante Software Release (*Planned Software Release* = 4.000.020) des Arbeitspakets mit dem Zieldatum für dieses Release in der Meilensteinliste, so beträgt der Verzug lediglich 8 Tage, wie in der unteren Berechnung ersichtlich.

Implementierung des Arbeitspakets 352749: 11.06.2014

Variante 1:

Zieldatum für die Implementierung des Arbeitspaketes in der *CI*-Tabelle: 02.08.2013

Der Zeitverzug beträgt bei dieser Variante 313 Tage (02.08.2013-11.06.2014), vergleiche Abbildung 5.10

Variante 2:

Zieldatum für die Implementierung des Arbeitspaketes in der Meilensteinliste:

Tabelle 5.1: Auszug aus der Meilensteinliste

Software Release	Zieldatum
4.000.020	03.06.2014

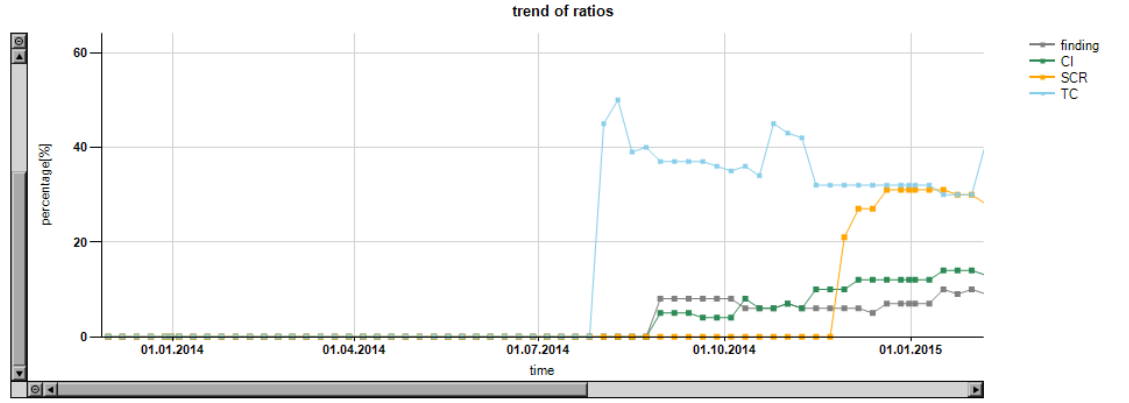
Der Zeitverzug beträgt bei dieser Variante 8 Tage (03.06.2014-11.06.2014), vergleiche Tabelle 5.1 und Abbildung 5.10.

Daher ist eine kritische Betrachtungsweise beider Varianten erforderlich, um eindeutige Aussagen über den Zeitverzug machen zu können.

5.5 Vergleich der abgeschlossenen Planungselemente

Dieses Fallbeispiel behandelt den Soll-/Ist-Verlauf von abgeschlossenen *SCRs*, *CIs* und *TCs* von *Beispielprojekt B*. Hierfür eignet sich der KPI „**Trend of ratios“**, da dieses den zeitlichen Verlauf der implementierten Anforderungen, Arbeitspakete und Testfälle illustriert, siehe

Abbildung 5.11. Bei der Betrachtung des Diagramms fällt auf, dass obwohl weit vor dem 01.01.2014 mit der Planung von Anforderungen begonnen wird, der Zeitpunkt der ersten Implementierung einiger *SCRs* erst am 24.11.2014 (gelbe Linie) geschieht. Der Sprung von 0% auf 21% in der Woche von 22.11.2014 bis 29.11.2014 ist mit der abrupten Implementierung von 1077 Anforderungen erklärt. Die ersten Testfälle werden schon früher abgeschlossen, jedoch auch hier innerhalb einer Woche, was den markanten Anstieg begründet (26.07.2014 bis 02.08.2014, hellblaue Linie).



snapshot_date	completed CIs	all CIs	percentage CIs	completed findings	all findings	percentage findings	implemented SCRs	all SCRs	percent SCRs
15.11.2014 00:0...	27	284	10	4	65	6	1	4668	0
22.11.2014 00:0...	28	285	10	4	65	6	1	4991	0
29.11.2014 00:0...	28	291	10	4	66	6	1078	5056	21
06.12.2014 00:0...	39	313	12	4	72	6	1383	5056	27
13.12.2014 00:0...	39	315	12	4	73	5	1383	5075	27
20.12.2014 00:0...	40	320	12	5	75	7	1554	5090	31

Abbildung 5.11: Vergleich der abgeschlossenen Planungselemente, [6], [47], vgl. [48], [49]

Nach erfolgreichem Abschluss der ersten Testphasen werden nach vier Monaten die zugehörigen Software Anforderungen abgeschlossen. Es werden im *Beispielprojekt B* also zuerst erfolgreiche Testergebnisse abgewartet, ehe die betreffende Anforderung der Systemkomponente abgeschlossen wird.

5.6 Verweisqualität der Anforderungen

In diesem Beispiel werden *Beispielprojekt A* und *Beispielprojekt B* miteinander verglichen. Hierfür eignet sich der KPI **„Software Tracking“**, um die Güte der Verlinkungen beider Projekte zu überprüfen, vergleiche Abbildung 3.3 in Kapitel 3. Es erfolgt eine Gegenüberstellung der Anzahl der Anforderungen, die mit *Concerns* mit den *CIs* verbunden sind. Die „Verweisgüte“ der *SCRs* untersucht also wie viele *SCRs* einen zugeordneten Testfall haben (mit *RelationshipTestIssue*) bzw. wie viele über einen *System Requirement*-Verweis (mit *Parents*) verfügen. Der KPI „Software Tracking“ wird einmal mit *Beispielprojekt A*, allen *Sub Components* der *SCRs* ausgeführt und einmal mit *Beispielprojekt B*, wiederum mit allen *Sub Components*, siehe Abbildung 5.12 und Abbildung 5.13. Die Anzahl der Systemfreigaben (*SF*) hat in diesem Beispiel keinerlei Bedeutung, da lediglich die Quantität der Anforderungen interessiert.

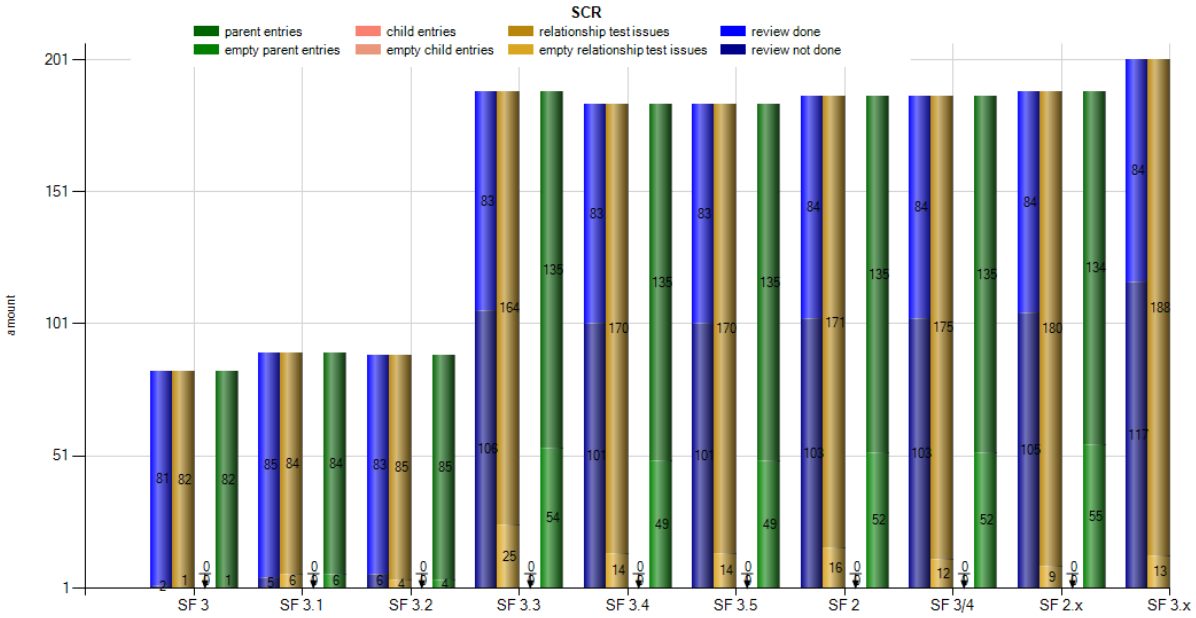


Abbildung 5.12: Verweisqualität der Anforderungen, *Beispielprojekt A*, [6], [47]

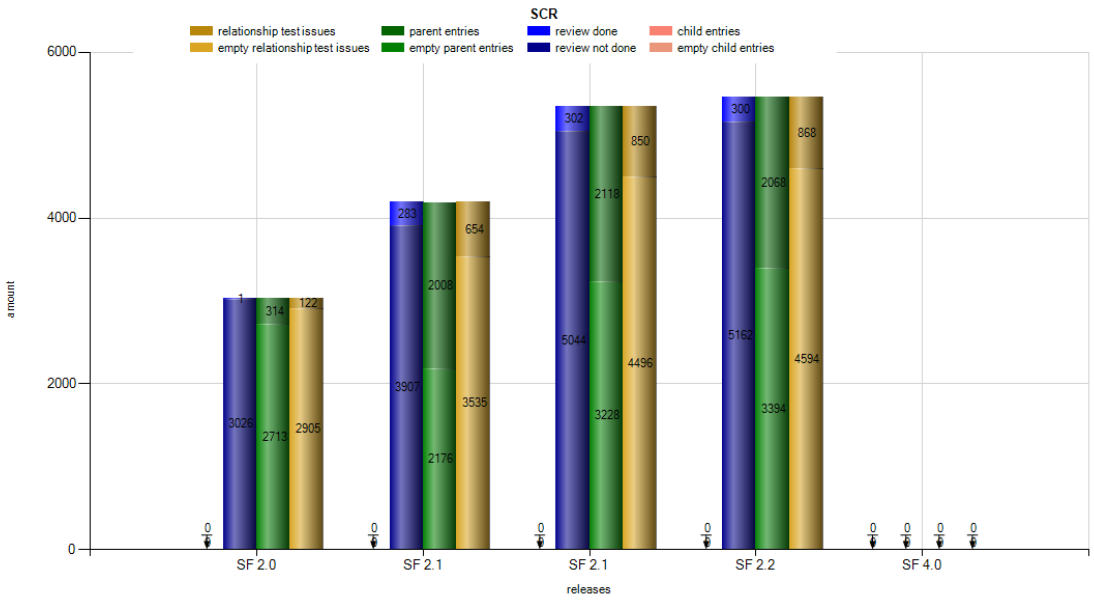


Abbildung 5.13: Verweisqualität der Anforderungen, *Beispielprojekt B*, [6], [47]

Tabelle 5.2 fasst die Erkenntnisse aus beiden Abbildungen zusammen. Dabei werden die relativen Häufigkeiten aus den gestapelten Balken berechnet und in den Spalten *Parents* bzw. *RelationshipTestIssue* eingetragen.

Tabelle 5.2: Vergleich der Verweisqualität beider Projekte

	Parents	***RelationshipTestIssue***	**Anzahl der über die *CIs* verlinkten *SCRs***
Beispielprojekt A	72-95%	87-94%	83-201
Beispielprojekt B	10-40%	4-16%	3026-5362

In Tabelle 5.2 ist ersichtlich, dass *Beispielprojekt A* deutlich weniger verlinkte Anforderungen aufweist als *Beispielprojekt B*, aber umso mehr angehängte Testfälle und System-Anforderungen und deshalb *Beispielprojekt A* insgesamt die höhere Verweisgüte besitzt.

5.7 Freeze-Points von Planungselementen

Dieses Beispiel soll sich dem Verlauf der abgeschlossenen Planungselemente über die Softwarereleases widmen. Es werden die KPIs **„Iteration Durations“** und **„Amount of all states“** benötigt und für das *Beispielprojekt B* ausgeführt. Betrachtet wird der Implementation-Freeze zum Software Release 4.000.000 am 24.11.2015 und ist der Meilensteinliste zu entnehmen.

Tabelle 5.3: *Implementation Freeze* in der Meilensteinliste

4.000.000	**Di 24.11.15**	**Implementation Freeze**

Die im KPI „Iteration Duration“ betrachteten *Implementation Freeze*- Punkte beziehen sich auf die Anforderungen (*SCRs*). Dabei ist ein stagnierendes Verhalten der implementierten Anforderungen über die Softwarereleases 2.001.001 bis 4.000.000 zu verzeichnen, siehe Abbildung 5.14. Der Release 4.000.000 wurde zu rund 33% erfüllt. Bestätigung dieser Erkenntnis liefert Abbildung 5.15 im KPI „Amount of all states“. Der Verlauf der implementierten *SCRs* wird hier mit roten Linien angezeigt. Anders verhalten sich die abgeschlossenen Arbeitspakete, wie in Abbildung 5.16 dargestellt. Die roten Linien verdeutlichen einen merkbaren Anstieg der implementierten *CIs* zum *Implementation Freeze* 4.000.000 am 24.11.2015. Die in der Meilensteinliste definierten *Implementation Freezes* in *Beispielprojekt B* stehen auch in Zusammenhang mit den Arbeitspaketen, da viele zum Software Release 4.000.000 geplant werden.

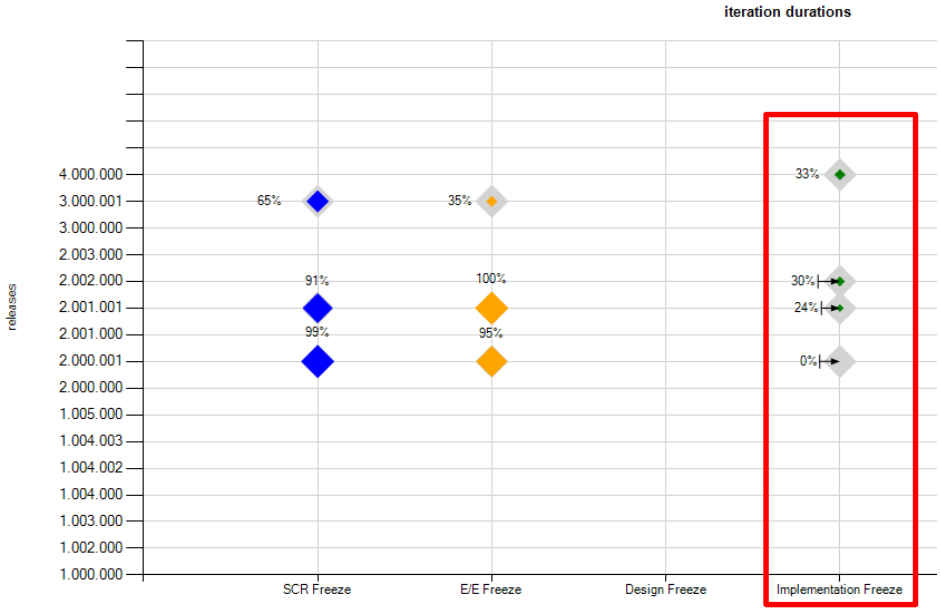


Abbildung 5.14: Freeze-Points Iteration Durations, [6], [47]

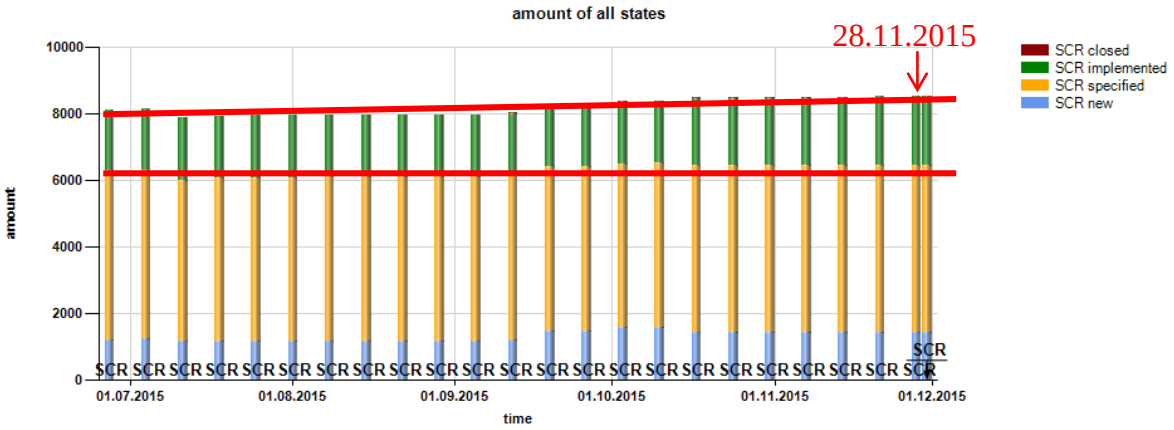


Abbildung 5.15: Implementierte Anforderungen, [6], [47], vgl. [48], [49]

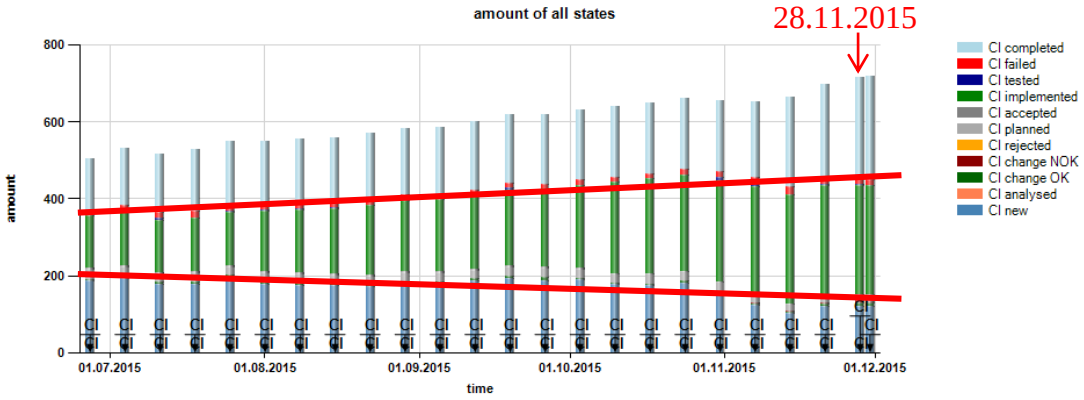


Abbildung 5.16: Implementierte Arbeitspakete, [6], [47], vgl. [48], [49]

6 Zusammenfassung

In der vorliegenden Masterarbeit wurde in den theoretischen Grundlagen zuerst auf die automotive Software eingegangen, um ein Grundverständnis für aktuelle Problemstellungen zu schaffen. Das V-Modell hat sich als Standard der Vorgehensmodelle in der Automobilindustrie herausgestellt und bildet zusammen mit dem Industriepartner vereinbarten Key Performance Indikatoren den Grundstein der Arbeit. Eine Marktanalyse der verfügbaren Analysetools hat ergeben, dass die Anforderungen des Industriepartners nur ungenügend mit bereits am Markt verfügbaren Programmen umsetzbar sind. Diese Online-Analysetools versprechen Flexibilität bei der Erstellung von Metriken oder Key Performance Indikatoren, jedoch konnte kein einziger KPI auf den Online-Plattformen nachgebildet werden. Der Entwicklungsprozess des Industriepartners erfordert eine Anpassung an die am Markt verfügbaren Analysetools, um mit den Entwicklungs- und Testdaten eine systematische Untersuchung von Projekten durchführen zu können. Dieser Kompromiss kann aus technischen und wirtschaftlichen Gründen nicht eingegangen werden, daher war es unabdingbar, ein eigenes Analysetool zu erstellen, um mit diesem eine Entwicklungs- und Testdatenanalyse durchzuführen.

Mit den Werkzeugen Visual Basic .NET und SQL wurde ein funktionierender Prototyp eines Analysetools erstellt, der alle vereinbarten KPIs enthält. Dazu wurden pro KPI eigene Klassen programmiert, die voneinander unabhängig fungieren. Jeder Key Performance Indikator enthält eigene maßgeschneiderte SQL-Abfragen an die Microsoft Access Datenbank. Somit ist es möglich, das Analysetool mit beliebig vielen KPIs generisch zu erweitern, falls im Entwicklungsprozess noch zusätzliche Fragestellungen auftauchen und im Tool beantwortet werden sollen. Das Analysetool ist modular aufgebaut, es können demnach beliebig viele Projekte der Datenbank hinzugefügt werden, unter der Voraussetzung, dass die Meilensteinliste aktuell gehalten wird.

Die Vorgehensweisen im Entwicklungsprozess des Industriepartners wurden mit der Untersuchung des Application Lifecycle Management Tools PTC MKS ersichtlich. Mit den Anforderungen (*SCRs*), Testfällen (*TCs*) und Arbeitspaketen (*CIs*) ist ein modifiziertes V-Modell definiert worden, dessen Vorgehensweise bei der Erstellung der KPIs berücksichtigt wurde. In Kapitel 4 wurden die einzelnen KPIs systematisch hinsichtlich deren zu beantwortenden Fragestellungen, deren Berechnungsmethodik sowie deren visuelle Darstellung unterteilt. Die praktische Handhabung wurde mit schematischen Beispielen nähergebracht. Abschließend erfolgte eine Entwicklungs- und Testdatenanalyse mit Datenbanken, welche die Datenstruktur von Industrieprojekten realgetreu nachbildet. Auffälligkeiten in Diagrammen verschiedener Beispielprojekte wurden untersucht und nach deren Ursache geforscht. Dabei sind mehrere KPIs für die Analyse von Auffälligkeiten herangezogen worden, um deren Interaktion miteinander zu veranschaulichen.

Das Analysetool soll direkt an einen SQL-Server angebunden werden, da derzeitige Abfragen wegen der lokalen Hinterlegung am Rechner in Form einer Microsoft Access-Datei im mehrfachen Minutenbereich liegen. Durch den modularen Aufbau des Analysetools müssen projektspezifische Anforderungen, Testfälle oder Arbeitspakete in das Front-End der Datenbank jeweils zu *SCR*-, *TC*-, *CI*-Tabellen gruppiert werden. Microsoft Access erlaubt jedoch nur einen sequenziellen Zugriff auf eine gruppierte Tabelle. Werden nun mehrere Planungselemente in einem KPI abgefragt, müssen diese nacheinander geöffnet und geschlossen werden. Ein weiterer Grund für Leistungseinbußen ist die Restriktion, dass Microsoft Access nur eine maximale Dateigröße von 2GB pro Datenbank zulässt, die Entwicklungs- und Testdaten jedoch diese 2GB-Grenze erheblich überschreiten, [50]. Diese Einbußen können jedoch mit der Direktanbindung an einen SQL-Server vermieden werden.

Abschließend sei angemerkt, dass mit diesem maßgeschneiderten Analysetool qualitativ hochwertige Auswertungen gemacht werden können, welche aber ein Grundverständnis des Entwicklungsprozesses erfordern, um aus den Diagrammen und Tabellen aussagekräftige Schlüsse ziehen zu können.

7 Abbildungsverzeichnis

8 Tabellenverzeichnis

9 Literaturverzeichnis

[1] J. Schäuffele und T. Zurawka, "Automotive Software Engineering", Springer Vieweg, 2013, ISBN 978-3-8348-2469-1.

[2] M. Broy, "Challenges in automotive Software Engineering", New York, 2006, ISBN 1-59593-375-1.

[3] AUTOSAR. [Online]. Available: http://www.autosar.org/. [Zugriff am 2.2.2016].

[4] J. Ludewig und H. Lichter, "Software Engineering: Grundlagen, Menschen, Prozesse, Techniken", dpunkt.verlag, 2013, ISBN 978-3-86491-298-6.

[5] PTC, „ptc.com“ PTC Integrity, [Online]. Available: http://de.ptc.com/application-lifecycle-management/integrity. [Zugriff am 21.12.2015].

[6] M. Ernst, M. Hirz, J. Fabian, F. Fehrer, J. Wolf und M. Steinkellner, "Key Process/Performance Indicators (KPIs) in der automotiven Softwareentwicklung", Forschungsbericht, TU Graz: Institut für Fahrzeugtechnik, 2015.

[7] J. Mössinger, "Software in automotive Systems", Technischer Bericht, IEEE Software, 2010.

[8] K. Reif, "Bosch Autoelektrik und Autoelektronik", Vieweg+Teubner, 2011, ISBN 978-3-8348-1274-2.

[9] ISO, 26262-1:2011, [Online]. Available: http://www.iso.org/iso/catalogue_detail?csnumber=43464. [Zugriff am 2.2.2016].

[10] I. Harris, "Embedded Software for Automotive Applications" in *"Software Engineering for Embedded Systems"*, Elsevier Inc., 2013, ISBN 978-0-12-415917-4.

[11] D. T. Beck, Vector Informatik GmbH, [Online]. Available: https://vector.com/vi_autosar_de.html. [Zugriff am 28.10.2015].

[12] ISO 26262-10:2012, [Online]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54591. [Zugriff am 5.11.2015].

[13] ITWIssen, „http://www.itwissen.info/“ [Online]. Available: http://www.itwissen.info/definition/lexikon/ISO-26262-ISO-26262.html. [Zugriff am 8.11.2015].

[14] VDI Verein Deutscher Ingenieure e.V., VDI 2206, "Entwicklungsmethodik für mechatronische Systeme", Version 04-2006.

[15] R. Höhn, "Das V-Modell XT", Springer, 2008, ISBN 978-3-540-30250-6.

[16] H. Luhn, "A Business Intelligence System", IBM Journal, 1958.

[17] H.-G. Kemper, "Business Intelligence – Grundlagen und praktische Anwendungen", Springer Verlag, 2010, ISBN 978-3-8348-0719-9.

[18] M. Grünwald und T. Dirk, "Business Intellegence", Technischer Bericht, Springer Verlag, 2009.

[19] M. K. Manfred Broy, "Projektorganisation und Management im Software Engineering", Springer-Verlag, 2013, ISBN 978-3-642-29290-3.

[20] IEEE, „IEEE Std 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology“, 1990.

[21] IEEE, „IEEE Std 1003.23-1998 IEEE Guide for Developing User Organization Open System Environment (OSE) Profiles“, 1998.

[22] API Advanced Performance Institute, "What are Key Performance Questions?", Management White Paper, 2010.

[23] E. Wallmüller, "Software-Qualitätsmanagement in der Praxis", Hanser Verlag, 2001, ISBN 978-3-446-21367-8.

[24] A. Sowa, "Metriken – der Schlüssel zum erfolgreichen Security und Compliance Monitoring", Vieweg + Teubner, 2011, ISBN 978-3-8348-1480-7.

[25] S. Erlachner, "Einsatz von Entwicklungs- und Testdaten zur Bewertung automotiver Software", Masterarbeit, TU Graz: Institut für Fahrzeugtechnik, 2015.

[26] I. Peltier Technical Services, Peltier Technical Services, Inc., 2014. [Online]. Available: http://peltiertech.com/Excel/pix2/speedoXPex.gif. [Zugriff am 28.12.2015].

[27] Microsoft Corporation, [Online]. Available: https://support.microsoft.com/de-de/allproducts. [Zugriff am 2.2.2016].

[28] D. Walter und G. Thomas, "Visual Basic 2015 Grundlagen, Profiwissen, Rezepte", Hanser Verlag, 2015, ISBN 978-3-446-44605-2.

[29] W. Doberenz und T. Gewinnus, "Datenbank-Programmierung mit Visual Basic 2012", O'Reilly Verlag, 2013, ISBN 978-3-84833-004-1.

[30] E. Schicker, "Datenbanken und SQL", Springer Vieweg, 2014, ISBN 978-3-8348-2185-0.

[31] R. Adams, "SQL Eine Einführung mit vertiefenden Exkursen", Hanser Verlag, 2012, ISBN 978-3-446-43278-9.

[32] „KPIFire“ [Online]. Available: https://www.kpifire.com/. [Zugriff am 20.12.2015].

[33] „SimpleKPI“, Iceberg Software Ltd., [Online]. Available: https://www.simplekpi.com/. [Zugriff am 20.12.2015].

[34] JIRA, „Atlassian JIRA Software“ [Online]. Available: https://de.atlassian.com/software/jira. [Zugriff am 18.12.2015].

[35] G. Pickl, "Untersuchung und Evaluierung von Analysetols", Bachelorarbeit, TU Graz: Institut für Fahrzeugtechnik, 2015.

[36] J. Rehermann und M. Blumenau, „datapine“ [Online]. Available: http://www.datapine.com/. [Zugriff am 8.2.2016].

[37] MySQL, [Online]. Available: https://www.mysql.de/. [Zugriff am 9.2.2016].

[38] ORACLE, [Online]. Available: http://www.oracle.com/at/corporate/contact/index.html. [Zugriff am 9.2.2016].

[39] Sisense, [Online]. Available: http://www.sisense.com/. [Zugriff am 21.02.2016].

[40]] CA Technologies, [Online]. Available: http://www.ca.com/us/products/ca-agile-central.html?intcmp=headernav. [Zugriff am 22.02.2016].

[41] InetSoft, [Online]. Available: https://www.inetsoft.com/. [Zugriff am 23.02.2016].

[42] M. Klostermann, „MKS Handbook - Introduction 2-“, Firmeninternes Dokument, MAGNA Powertrain GmbH & Co. KG, 2011.

[43] practitest.com, „Practitest“ [Online]. Available: http://www.practitest.com/application-lifecycle-management/. [Zugriff am 21.12.2015].

[44] M. Klostermann, „MKS Items Annotiated Masks and Workflows“, Firmeninternes Dokument, MAGNA Powertrain GmbH & Co. KG, 2013.

[45] M. Klostermann, „Schematical MKS Document Structure“, Firmeninternes Dokument, MAGNA Powertrain GmbH & Co. KG, 2014.

[46] M. Ernst, "Integrated Development Process for Automotive Mechatronic Systems", Doktorarbeit in Bearbeitung, TU Graz: Institut für Fahrzeugtechnik, 2016.

[47] M. Ernst, M. Hirz und J. Fabian, "The Potential of Key Process/Performance Indicators (KPIs) in Automotive Software Quality Management", Paper 2016-01-0046, USA: SAE World Congress, 2016.

[48] M. Ernst, S. Erlachner, M. Hirz und J. Fabian, "Optimisation of Automotive Software Quality Management by Use of Analysis Methods in the Development Process of Mechatronic Systems", Antalya, Türkei: International Conference on Advances in Software, Control and Mechanical Engineering, 2015, ISBN 978-93-84422-37-0.

[49] M. Ernst, S. Erlachner, M. Hirz, J. Fabian und F. Wotawa, "Analysis Methods in the Development Process of Mechatronic Drivetrain Systems with Special Focus on Automotive Software", WMSCI, Orlando, Florida, USA: The 19th World Multi-Conference on Systemics, Cybernetics and Informatics, 2015, ISBN 978-1-941763-25-4.

[50] Microsoft Corporation, „Access 2010-Spezifikationen,“ [Online]. Available: https://support.office.com/de-de/article/Access-2010-Spezifikationen-1e521481-7f9a-46f7-8ed9-ea9dff1fa854. [Zugriff am 10.01.2016].

[51] P. Steve Burbeck, „How to use Model-View-Controller (MVC)“, Technischer Bericht, 1992.

[52] J. Wolf, "Implementierung eines Analysetools", Bachelorarbeit, TU Graz: Institut für Fahrzeugtechnik, 2016.

[53] Magna Powertrain GmbH & Co. KG, Firmeninternes Dokument, KPI-Meeting, 30.04.2015.