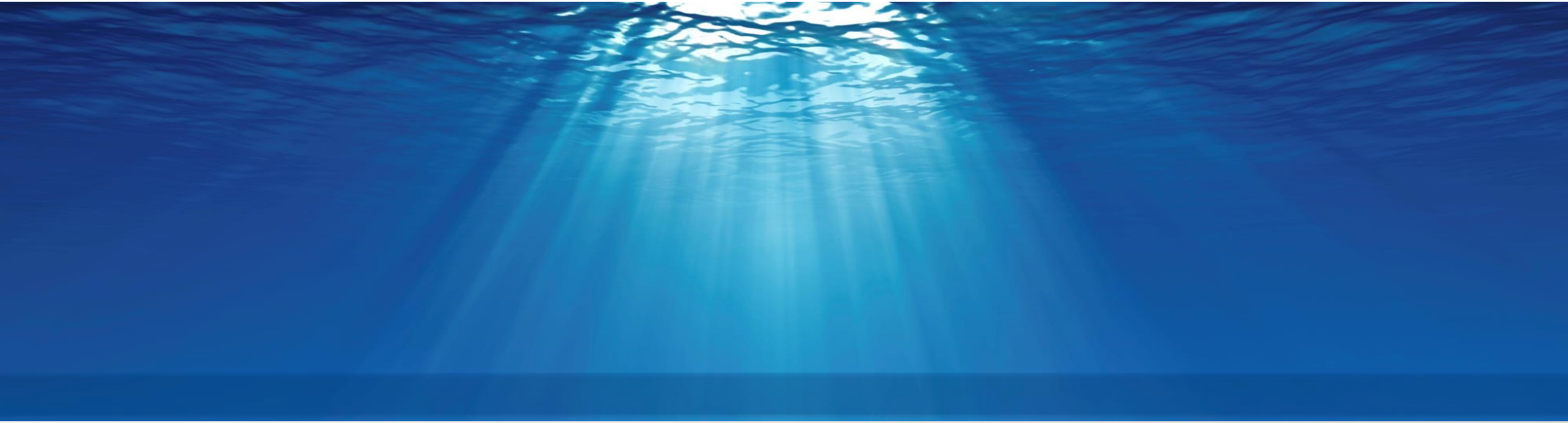
UNIKASSEL VERSITÄT

octaVIA
SAP-Kompetenz und Branchenwissen



SAP-Kompetenz und Branchenwissen aus einer Hand

Business Process Engineering - Seminar WS 2019/2020

Felix Bodewald

Feedback Lastenheft

- Korrekte Rechtschreibung und Grammatik sind wünschenswert
- BPMN sollte im Lastenheft vorhanden sein
- Prototyp wird aus Lastenheft erstellt, d.h. Screenshots sind nicht verfügbar
  - Lastenheft -> kommt vom Auftraggeber
  - Prototyp -> wird vom Dienstleister (z.B. OctaVIA AG) erstellt

Testing/ Qualitätsmanagement

Agenda

- Einführung Software Testing
- Software Development Life Cycle
- Testmanagementmethoden
- Software Testing Life Cycle
- Die 7 Grundsätze des Software Testens
- Die Aufgaben eines Software Testers
- Beispiele Testdokumente
- Testarten
- Testmodelle
- Testmanagementtools

Software Testing

Was ist Testing?

Ein Prozess, um die Richtigkeit, Vollständigkeit und Qualität einer Entwicklung zu bestimmen

Wer sind die Tester?

Entwickler

Tester

Anwender
(Beta-Testing)

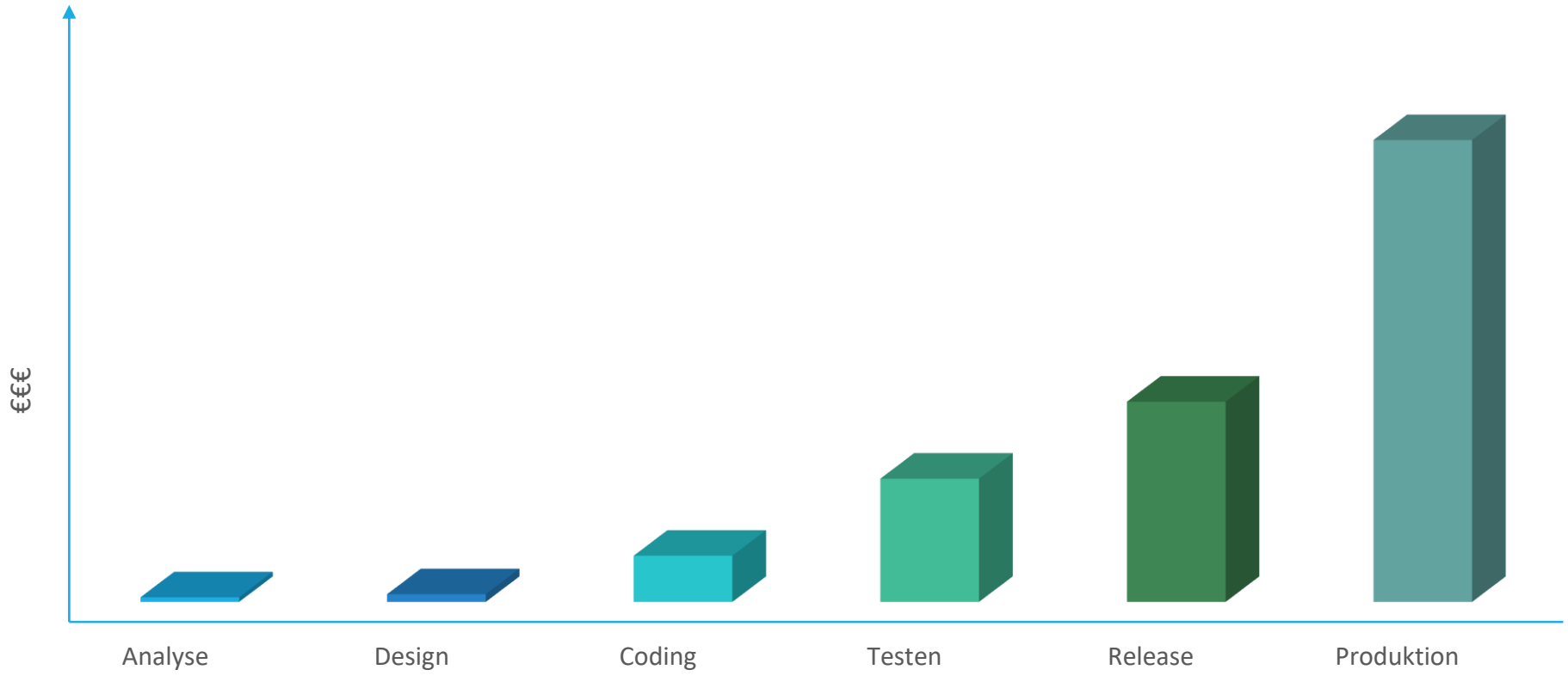
Warum wird getestet?

Qualitätssicherung

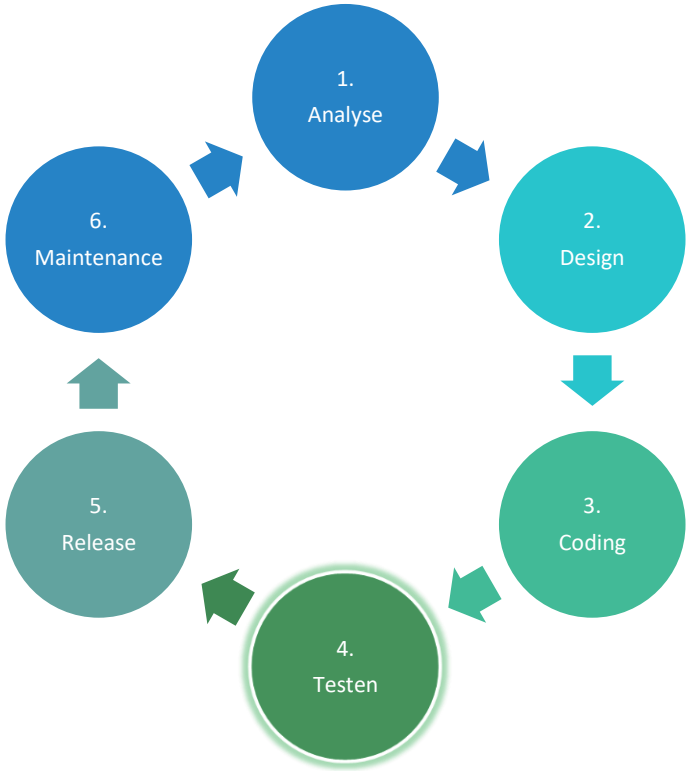
Sicherheit

Wann wird getestet?

So früh wie möglich im *Software Development Life Cycle*



Wie viel kostet ein Fehler?

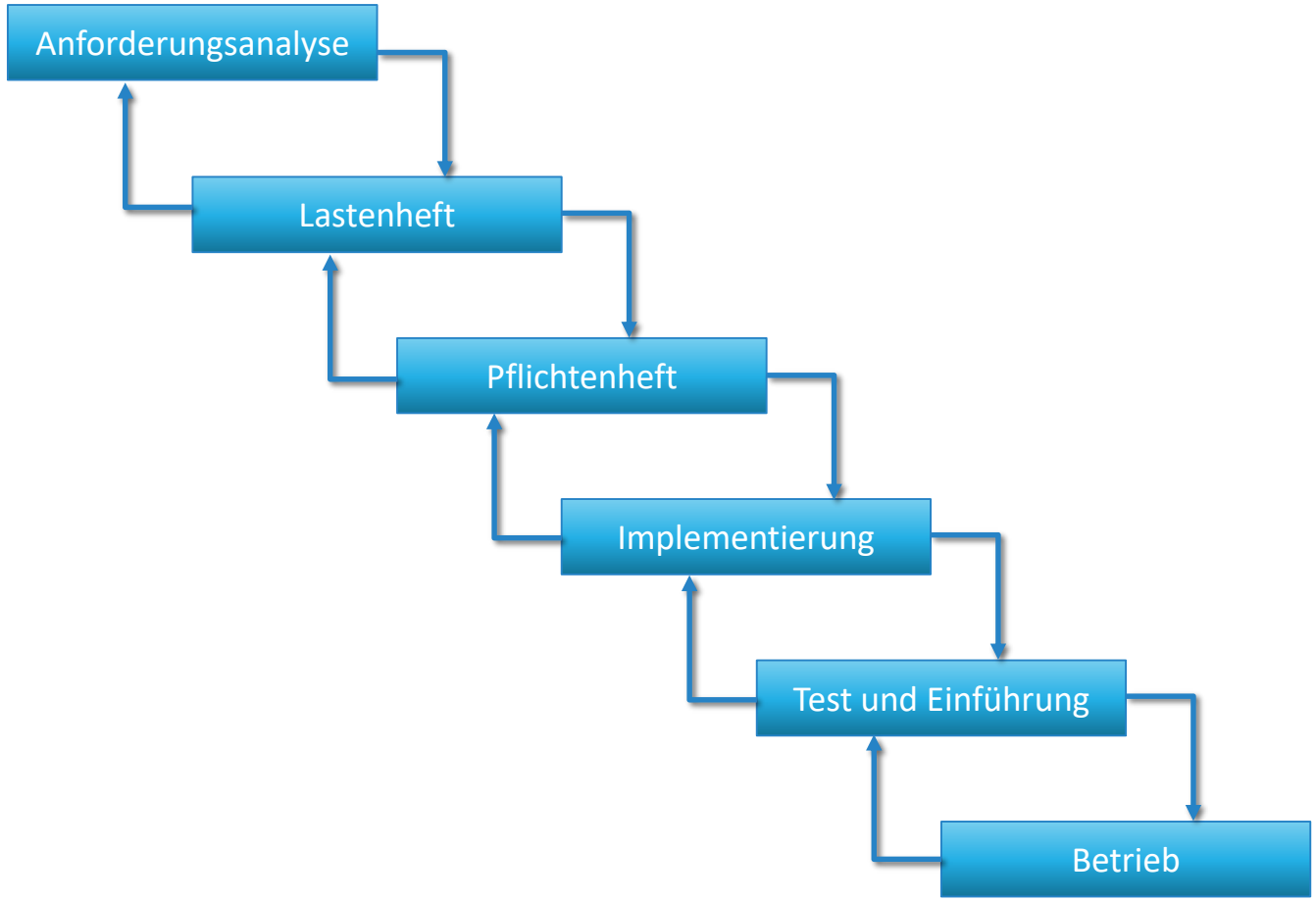


Software Development Life Cycle
- SDLC -

Vorgehen im Projekt

Das Wasserfallmodell (klassisch)

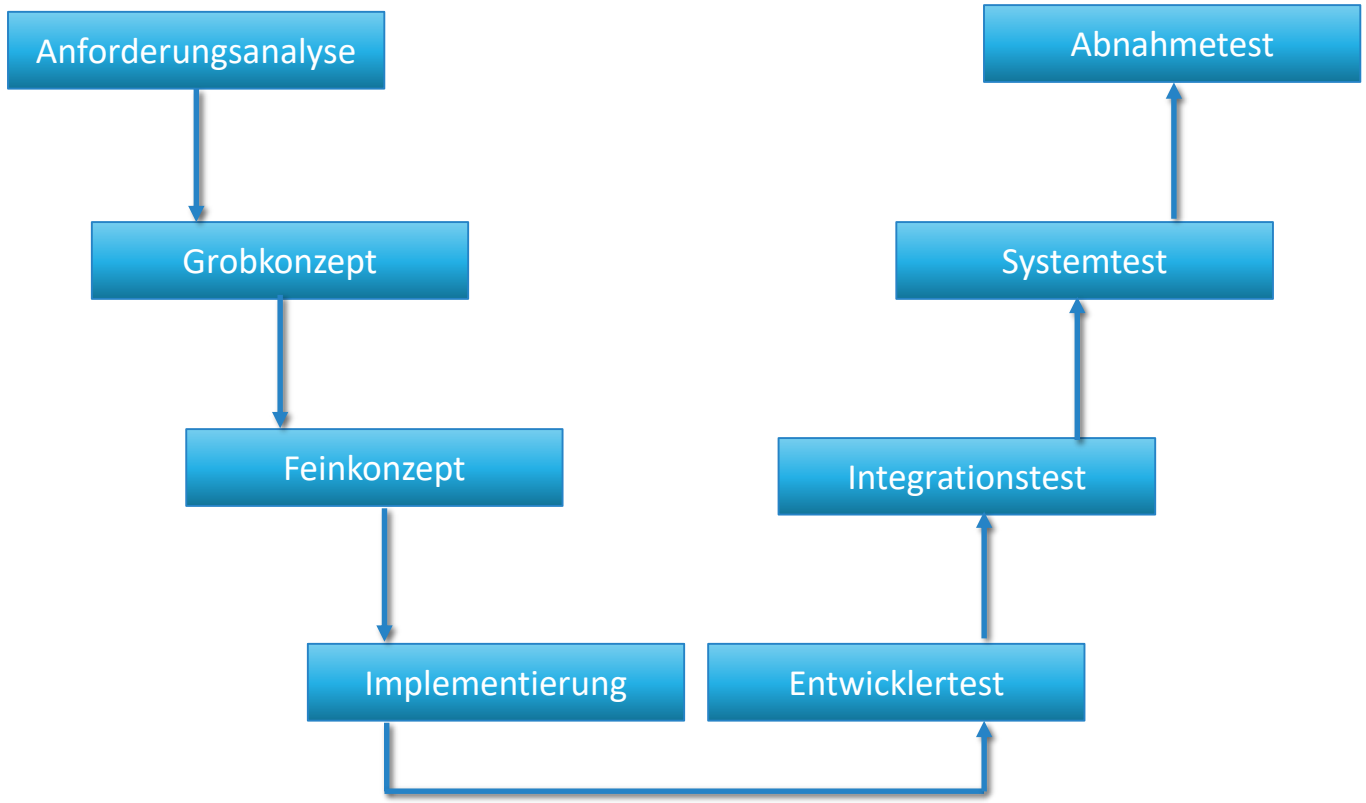
- Die einzelnen Phasen werden linear nacheinander abgearbeitet
  - Lineares Vorgehensmodell
- Beim Übergang von einer Phase zur nächsten gibt es definierte Ergebnisse, die erreicht sein müssen, um in die nächste Phase wechseln zu können
- i.d.R. sechs Phasen
- Erweiterung mit Rücksprungmöglichkeiten, um ein Problem zu beheben
  - Zeit- und Kostenintensiv



Vorgehen im Projekt

Das V-Modell (klassisch)

- Basiert auf dem Wasserfallmodell
- In DE für die Planung und Durchführung von IT-Softwareentwicklungsprojekten der öffentlichen Hand zwingend vorgeschrieben
- Testphasen werden den jeweiligen Realisierungsphasen gegenübergestellt
- Seit 2005: V-Modell XT
  - Orientiert sich mehr an der agilen Softwareentwicklung



Vorgehen im Projekt

Agile Methoden

Scrum

- Bekanntestes agiles Vorgehensmodell/ Rahmenwerk
- Wenige vorgegebene Regeln
- Drei Artefakte
  - Product Backlog
  - Sprintbacklog
  - Produktinkrement
- Ziel: komplexe Lösungen in vielen kurzen Iterationen erstellen
  - Auch die Planung wird iterativ durchgeführt

Sprint = Kurze Iteration fester Länge (meist drei bis vier Wochen)

Jeder Sprint soll eine auslieferbare Softwarelösung erzeugen = ***Inkrement***	Mit jedem Sprint wächst der Funktionsumfang der Softwarelösung

Product Backlog = Verantwortung liegt beim *Product Owner*

Hält alle bekannten Anforderungen und Arbeitsergebnisse fest, die der Erreichung des Projektziels dienen

Sprint Backlog = hält Aufgabenmenge überschaubar, wird währen dem Sprint nicht verändert

Legt fest, welche Anforderungen in einem Sprint umgesetzt werden	Nur Anforderungen, die genau genug formuliert wurden (Definition of Ready)	Kriterien zur Beurteilung der erfolgreichen Fertigstellung des Sprints (Definition of Done)

Transparenz

Der aktuelle Stand muss für alle sichtbar sein	Im ***Daily Scrum*** wird der aktuelle Stand täglich und gemeinsam besprochen

Scrum

Ereignisse

Sprint Planning

- Die Aufgaben für einen Sprint festlegen, d.h. Erstellen des ***Sprint Backlogs***

Daily Scrum

- Zum Besprechen der Aufgaben die die einzelnen Teammitglieder an einem Tag erledigt haben

Sprint Review

- Zum Besprechen der erledigten Aufgaben und des Inkrements

Sprint Retrospektive

- Nachbesprechung des vergangenen Sprints
- Verbesserungen für den nächsten Sprint besprechen

Rollen

Product Owner

- Führt und verantwortet das ***Backlog***
- Vertritt die Belange des Kunden

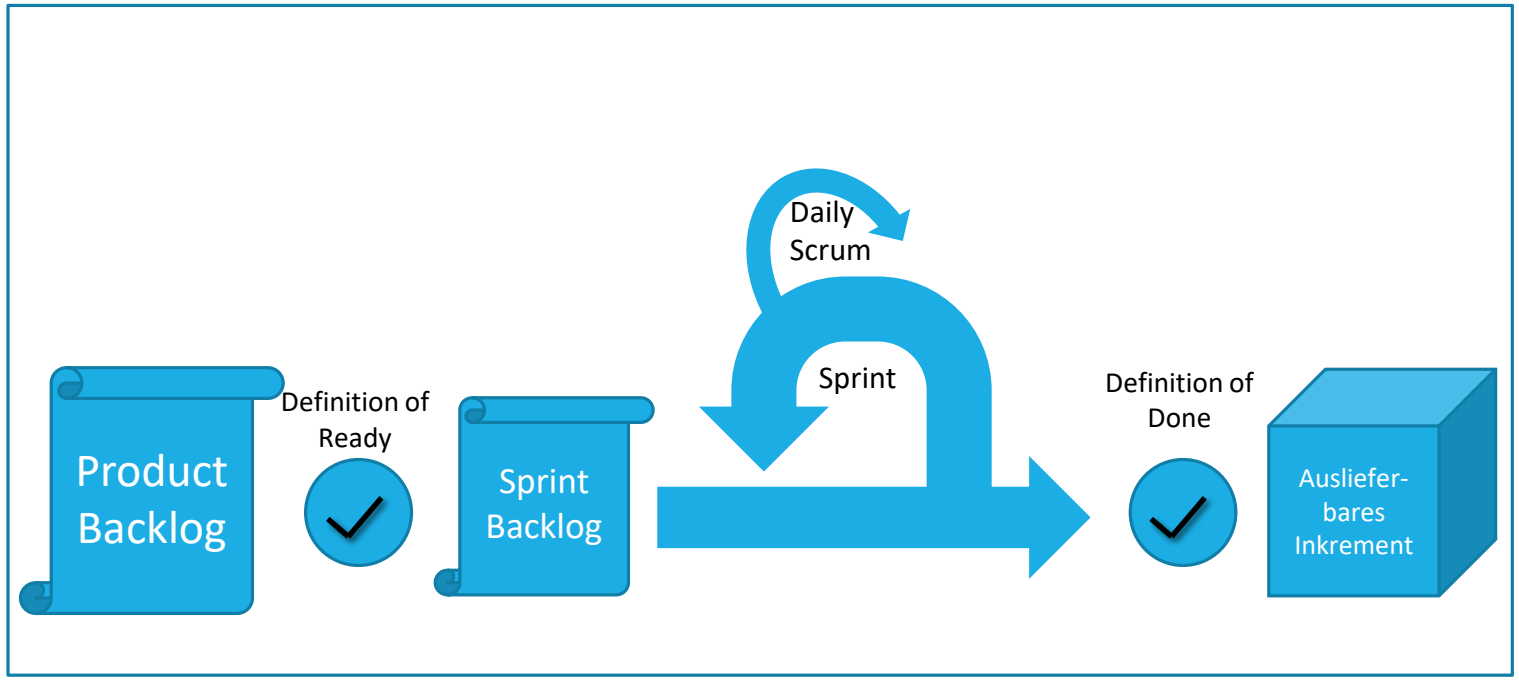
Scrum Master

- Sorgt dafür, dass die ‚Spielregeln' eingehalten werden
- Moderator bei den Meetings
- Kümmert sich um die Behebung von Störungen

Entwicklungsteam

- Für die Lieferung der Softwarefunktionalität verantwortlich
- i.d.R. fünf bis neun Personen
- Organisiert sich selbst

Scrum Prozess



Wichtige Aspekte zum Testen

- Tests müssen wiederholbar sein.
- Tests müssen dokumentiert sein.
- Tests müssen nachvollziehbar sein.
- Tests müssen im geplanten Zeitrahmen durchführbar und wirtschaftlich sein.

Unabhängig Testen

Die Tester sollten eine unabhängige Partei darstellen und nicht im Entwicklungsprozess involviert gewesen sein.

Auch auf ungültigen Input testen

Das System sollte korrekte Mitteilungen generieren, wenn ungültige Tests durchgeführt werden und korrekte Ergebnisse erzielen, wenn der Test gültig ist.

Software während des Tests statisch halten

Die Software sollte während des Testvorgangs nicht verändert werden.

Erwartete Ergebnisse bereitstellen (wenn möglich)

In der Testdokumentation sollten erwartete Ergebnisse spezifiziert werden.

7 Grundsätze des Software Testens

Testen zeigt die Anwesenheit von Fehlerzuständen

- Nur weil keine Fehler gefunden wurden, heißt das nicht, dass keine vorhanden sind

Erschöpfendes Testen ist unmöglich

- Bei komplizierten Entwicklungen ist es unmöglich alle möglichen Kombinationen und Szenarien zu testen.
- Der Testaufwand muss dem Risiko und den Prioritäten angepasst werden.

Frühes Testen

- So früh wie möglich im SDLC testen

Häufung von Fehlern

- Die Mehrheit der Fehler wird durch eine geringe Anzahl von Modulen erzeugt.
- Pareto Prinzip: 80% der Fehler in 20% der Module.

Wiederholungen haben keine Wirksamkeit

- Alte Testfälle sind für neuere Versionen nicht mehr gültig.

Testen hängt vom Umfeld ab

- Die Art der Applikation bestimmt die Methoden, Techniken und Typen des Testens.

Irrtum: „Keine Fehler“ bedeutet ein brauchbares System

- Ein fehlerfreies System ist nicht automatisch auch ein brauchbares System.
- Ein kompliziertes Design z.B. kann die Anwendung negative beeinflussen.

Teststufen

jede Stufe hat ihre eigene Definition über Art, Umfang und Ziele der Testdurchführung

Entwicklertest

- Alle Tests die von Softwareentwicklern durchgeführt werden
- In agilen Projekten ist dieser Begriff zu vermeiden, besser: *Unit Tests, Komponententests* oder *Modultests*
- Sobald ein Objekt erstellt wurde, wird dieses vom Entwickler isoliert getestet (auch Negativtests)

Integrationstest

- Sobald alle Komponenten erfolgreich getestet wurden, können diese zusammen, d.h. integrativ, getestet werden
- Fehler im Zusammenspiel der Programmteile untereinander aufdecken
- Erste Berechtigungsfehler aufdecken

Systemtest

- Umsysteme bzw. Drittsysteme werden mit in den Test einbezogen, d.h. es werden alle Systeme getestet, die von der Entwickelt beeinflusst werden
- Unterschiede zwischen der Kundenanforderung und dem tatsächlichen Verhalten des Gesamtsystems feststellen

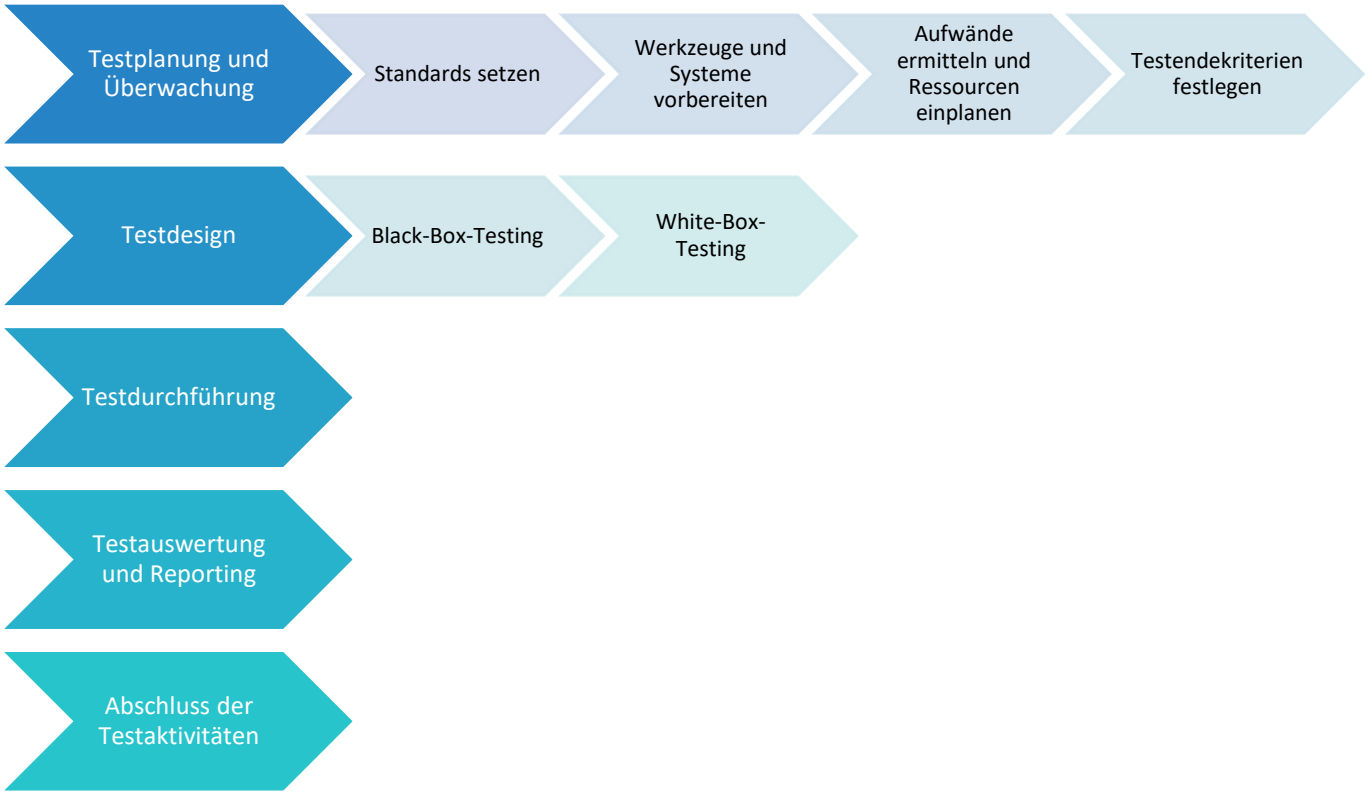
Abnahmetest

- Wird vom Kunden auf dem Konsolidierungs- oder Qualitätssicherungssystem durchgeführt

Regressionstest

- Werden durch das Einspielen von Hinweisen, Support Packages oder Enhancement Packages eingeleitet
- Sicherstellen, dass das System auch nach Veränderungen noch funktioniert wie es soll

Testprozess



Black-Box- vs. White-Box-Testing

- Testdesign -

Black-Box-Testing

Testet das von außen sichtbare Verhalten des Testobjekts

- End-User-Experience
- Funktionstüchtigkeit

Code wird nicht berücksichtigt

Testfälle werden auf Grundlage der fachlichen Spezifikation erstellt

- Output eines gegebenen Inputs testen
- Qualität der Testfälle hängt von den Testdaten ab

White-Box-Testing

Tests werden auf Grundlage des Programmcodes erstellt

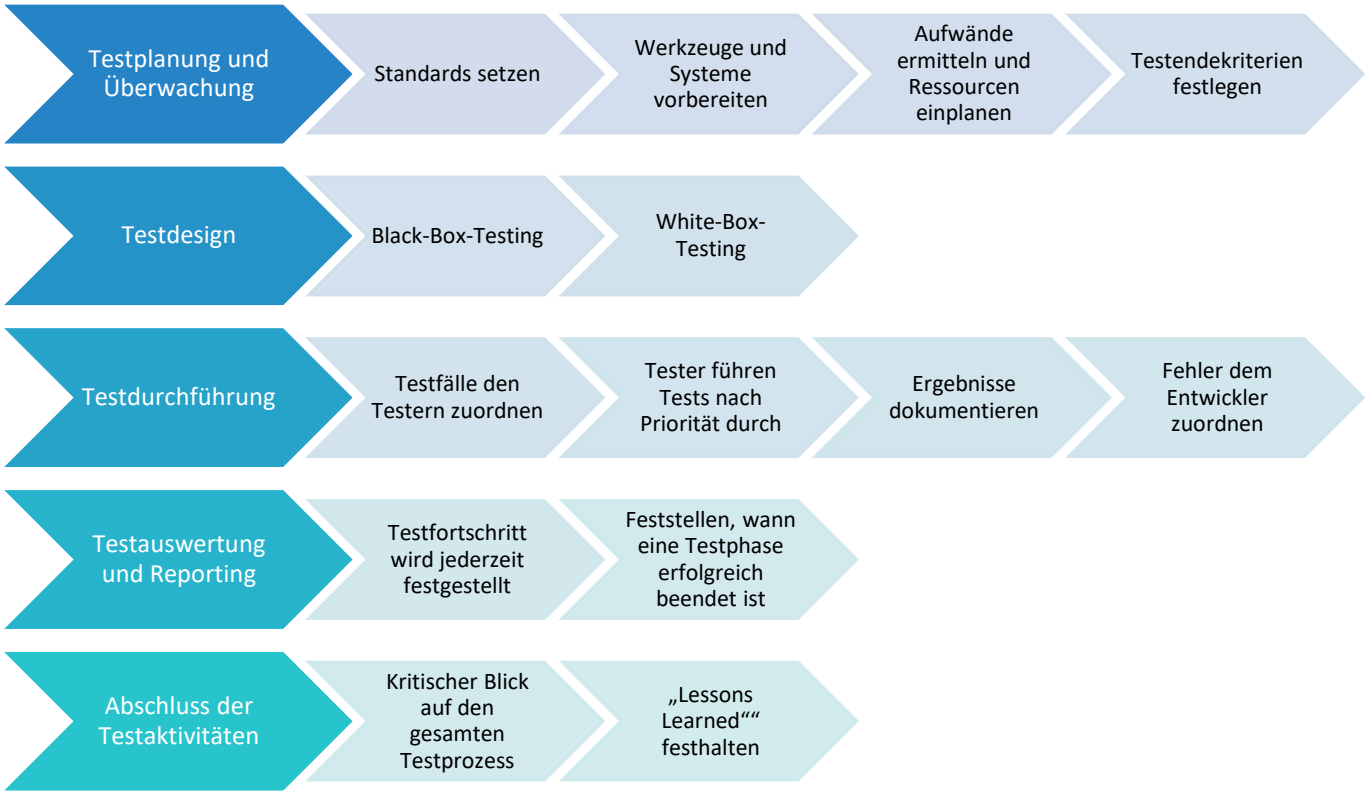
- Der Ersteller braucht Wissen über die Programmiertechnik

Getestet wird im Hinblick auf:

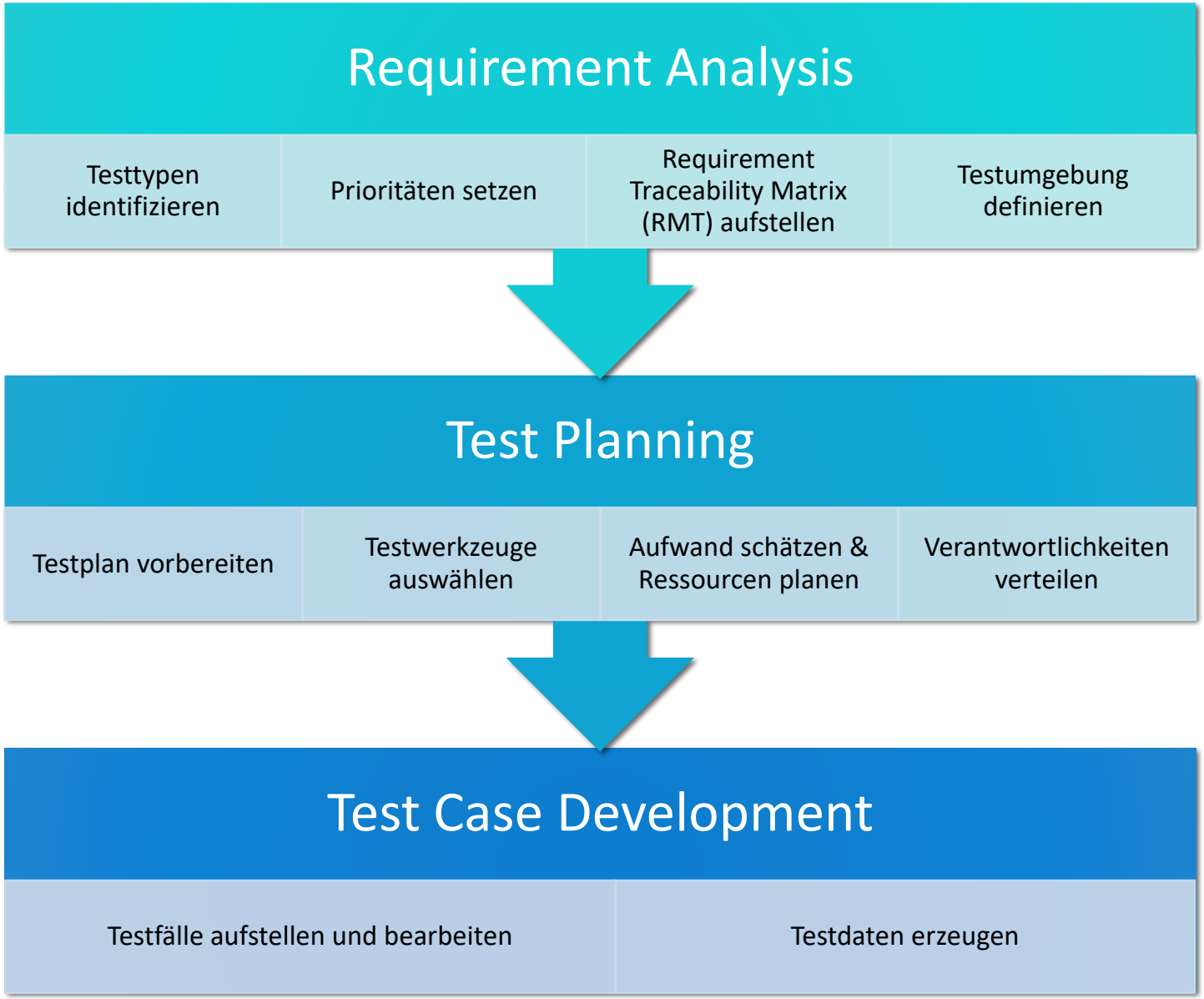
- Sicherheitslücken
- Design und Anwendbarkeit
- Kaputte oder schlecht strukturierte Pfade im Coding-Prozess

Den Quellcode testen

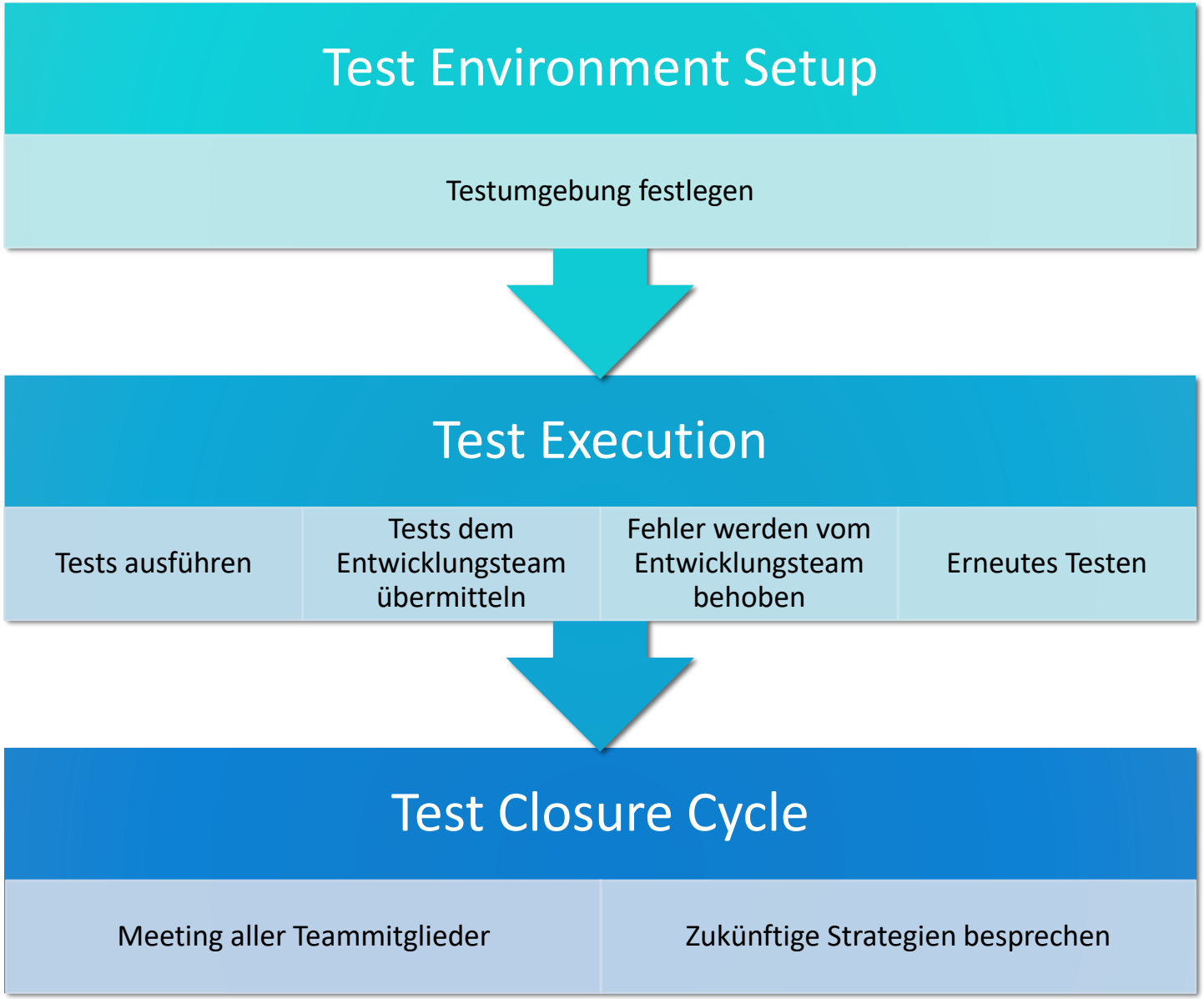
Testprozess



Software Testing Life Cycle – STLC –



Software Testing Life Cycle – STLC –



Beispiel Testplan

Welche Testtypen sind relevant und welche nicht?

Im Umfang	Nicht im Umfang
Plattformübergreifendes Testen	Browserübergreifendes Testen
Performance Testing	
Funktionales Testen	

Projektrisiken und Produktrisiken festlegen

Risiko	Wahrscheinlichkeit	Auswirkung	Minderung
Projektrisiko:			
Senior Teammitglied verlässt das Team plötzlich	5	3	Wissenstransfer Ressourcen Buffer
Produktrisiko:			
Das System lässt sich in der Testumgebung nicht installieren	8	10	Smoke Testing

Test Szenario	Testfall	Vorbedingung	Testschritte	Testdaten	Vorausgesagtes Ergebnis	Istergebnis	Bestanden/ Fehlgeschlagen
Login-Funktionalität prüfen	Antwort nach Eingabe von ungültigem Benutzernamen und Passwort prüfen	Applikation muss installiert sein	1. Applikation starten 2. Namen eingeben 3. Passwort eingeben 4. OK Button klicken	Benutzername: Admin Passwort: app123	Login muss NICHT erfolgreich sein	Login NICHT erfolgreich	Bestanden
Login-Funktionalität prüfen	Antwort nach Eingabe von gültigem Benutzernamen und Passwort prüfen	Applikation muss installiert sein	1. Applikation starten 2. Namen eingeben 3. Passwort eingeben 4. OK Button klicken	Benutzername: Admin1 Passwort: app123	Login muss erfolgreich sein	Login NICHT erfolgreich	Fehlgeschlagen
Login-Funktionalität prüfen	Antwort nach Eingabe von gültigem Benutzernamen und Passwort prüfen	Applikation muss installiert sein	1. Applikation starten 2. Namen eingeben 3. Passwort eingeben 4. OK Button klicken	Benutzername: Admin1 Passwort: app123	Login muss erfolgreich sein	Login erfolgreich	Bestanden

Beispiel Testfall

Funktionales- vs. Nicht-Funktionales Testen

Funktionales Testen	Nicht-Funktionales Testen
Wird von Software Testern durchgeführt	Man testet die Performance, Anwendbarkeit und Skalierbarkeit eines Systems
Bezieht sich auf die funktionalen Ansprüche eines Systems	Steht nicht in Beziehung zu spezifischen Funktionen

Statisches Testen – Review Stages

Planning Stage
Manager prüft das zu testende Dokument auf Fehler

Kick-Off Meeting (optional)
alle Beteiligten auf den selben Stand bringen

Vorbereitung
die Teilnehmer des Review Meetings gehen einzeln das Dokument durch, um Fehler zu finden

Review
ein Dokument analysieren und Veränderungen vorschlagen, um die Qualität zu verbessern

Re-Work Phase
Autor ändert das Dokument mit Bezug auf die Vorschläge des Review Meetings

Follow-Up Phase
der Moderator verbreitet das neue Dokument an alle beteiligten Personen, damit diese es überprüfen können

Aufgabe 5

1) Teste den von uns bereitgestellten Prototypen „Prototype zum Debuggen“ und dokumentiere deine Tests in der Jira Vorlage.

- Um die von dir durchgeführten Tests abzubilden, lege Testvorgänge analog des **Templates Jira Testing** an. Denke daran auch die gefundenen Bugs zu dokumentieren.
- Sortiere die Vorgänge in das Backlog.

2) Fixe die von dir gefundenen Bugs im Prototypen.

3) Teste den Prototypen erneut und dokumentiere die Tests in der Vorlage.

-> es sind insgesamt 10 Bugs im Prototyp vorhanden!

Quellen

https://aviation-safety.net/database/record.php?id=19940426-0

https://de.wikipedia.org/wiki/Therac-25#Fallgeschichte

https://web.archive.org/web/20071212183729/http://neptune.netcomp.monash.edu.au/cpe9001/assets/readings/www_uguelph_ca_~tgallagh_~tgallagh.html

https://www.youtube.com/watch?v=goaZTAzsLMk

https://deepsource.io/blog/exponential-cost-of-fixing-bugs/

https://www.testingexcellence.com/seven-principles-of-software-testing/

https://comquent.de/de/die-sieben-grundsaetze-des-softwaretestens/

https://www.youtube.com/watch?v=3bJcvBLJViQ

https://www.youtube.com/watch?v=Wi75S5TTfQ0

https://blog.seibert-media.net/blog/2011/05/16/software-tests-notwendigkeit-arten/

https://www.oliver-lampert.at/glossar/testarten/

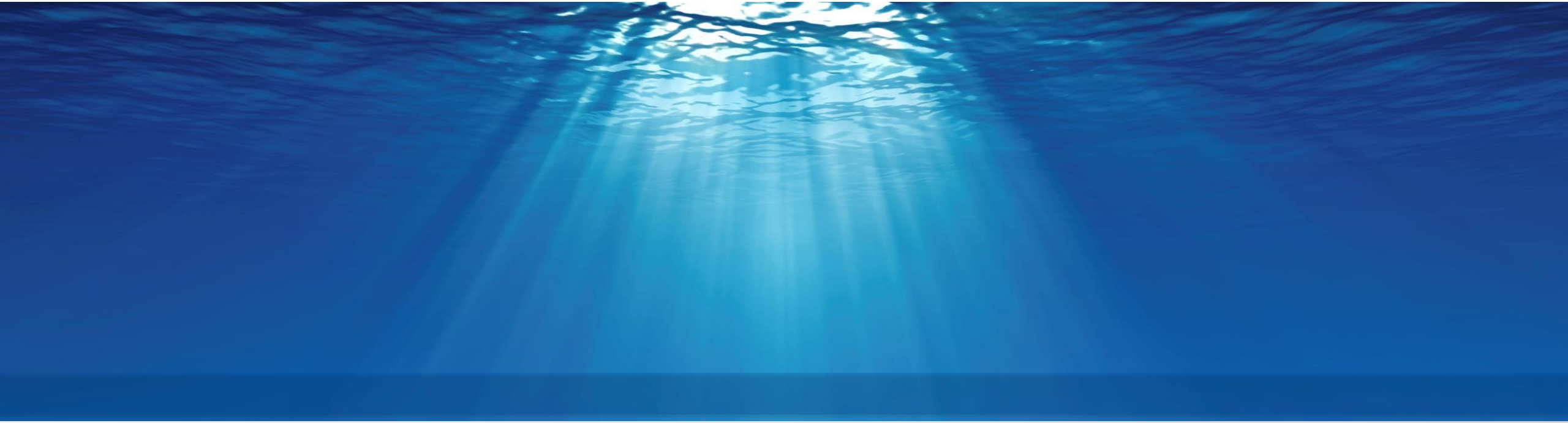
https://karrierebibel.de/kanban/

https://www.atlassian.com/de/software/jira

Huber, Stefan. „SAP-Testmanagement – Tests planen, entwickeln und durchführen“. Bonn: Rheinwerk Verlag, SAP Press. 2015. Print.

UNIKASSEL VERSITÄT

octaVIA
SAP-Kompetenz und Branchenwissen



SAP-Kompetenz und Branchenwissen aus einer Hand

Business Process Engineering WS 2019/2020

Felix Bodewald