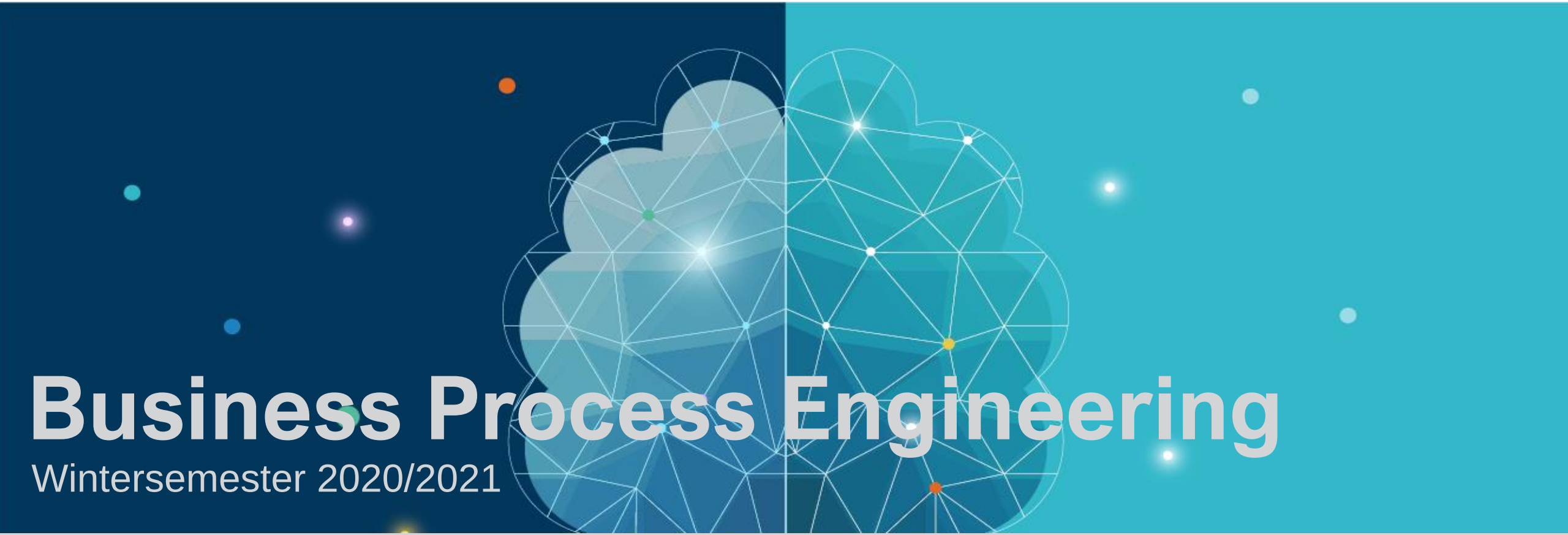
UNIKASSEL VERSITÄT

octaVIA
SAP-Kompetenz und Branchenwissen



Business Process Engineering

Wintersemester 2020/2021

Software Testing & Qualitätssicherung

Agenda

- Organisatorisches
- ISTQB
- Was ist Testen?
- Die 7 Grundsätze des Software Testing
- Testprozess
- Die Psychologie des Testens
- Teststufen
- Black-Box- und White-Box-Testing
- Statisches Testen vs. Dynamisches Testen
- Beispiel Testplan
- TUT, FUT, UAT
- Aufgaben eines Softwaretesters

Organisatorisches

- Abgabe Aufgabe 9
  - Donnerstag 28.01.2021, 14:00 Uhr (1. Abgabe)
  - Sonntag 31.01.2021, 23:59 Uhr (2. Abgabe)

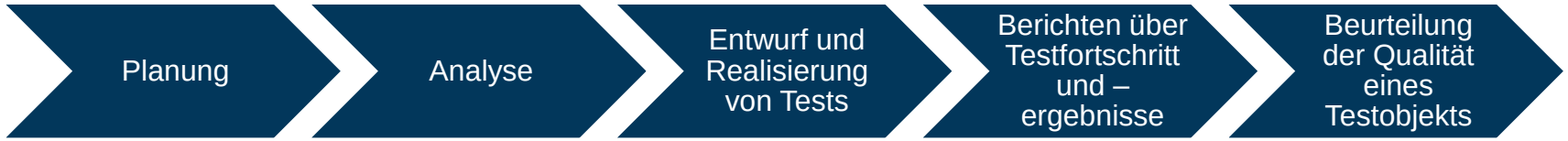
- Aufgabe 10
  - Wird morgen Abend gestellt (29.01.2021)
  - Donnerstag 04.02.2021, 14:00 Uhr (1. Abgabe)
  - Sonntag 07.02.2021, 23:59 Uhr (2. Abgabe)

ISTQB

- **I**nternational **S**oftware **T**esting **Q**ualifications **B**oard
  - Gemeinnützige Zertifizierungsstelle für Softwaretester
- 2002 gegründet und in Belgien registriert
- 2019: 63 nationale Member Boards, die für eine Abdeckung von mehr als 120 Länder sorgen
- Erstellt Lernpläne sowie Regeln zur Akkreditierung und Zertifizierung von Schulungen
- Im deutschsprachigen Raum gibt es drei offizielle Member Boards des ISTQB
  - Swiss Testing Board
  - Austrian Testing Board
  - German Testing Board

Was ist Testen?

- Software die nicht richtig funktioniert führt zu **Problemen**, z.B. Geld-, Zeit- oder Imageverlust, Verletzung, Tod
- Softwaretesten = ein Mittel, die **Qualität von Software zu beurteilen** und das Risiko einer Fehlerwirkung* im Betrieb zu reduzieren
- Es geht nicht um das reine Ausführen von Tests
  - Testen ist ein **Prozess**:



  - Es wird ebenfalls geprüft, ob das System in seiner Einsatzumgebung die Bedürfnisse von Benutzern und anderen Stakeholdern erfüllen wird

* z.B. eine Error-Meldung

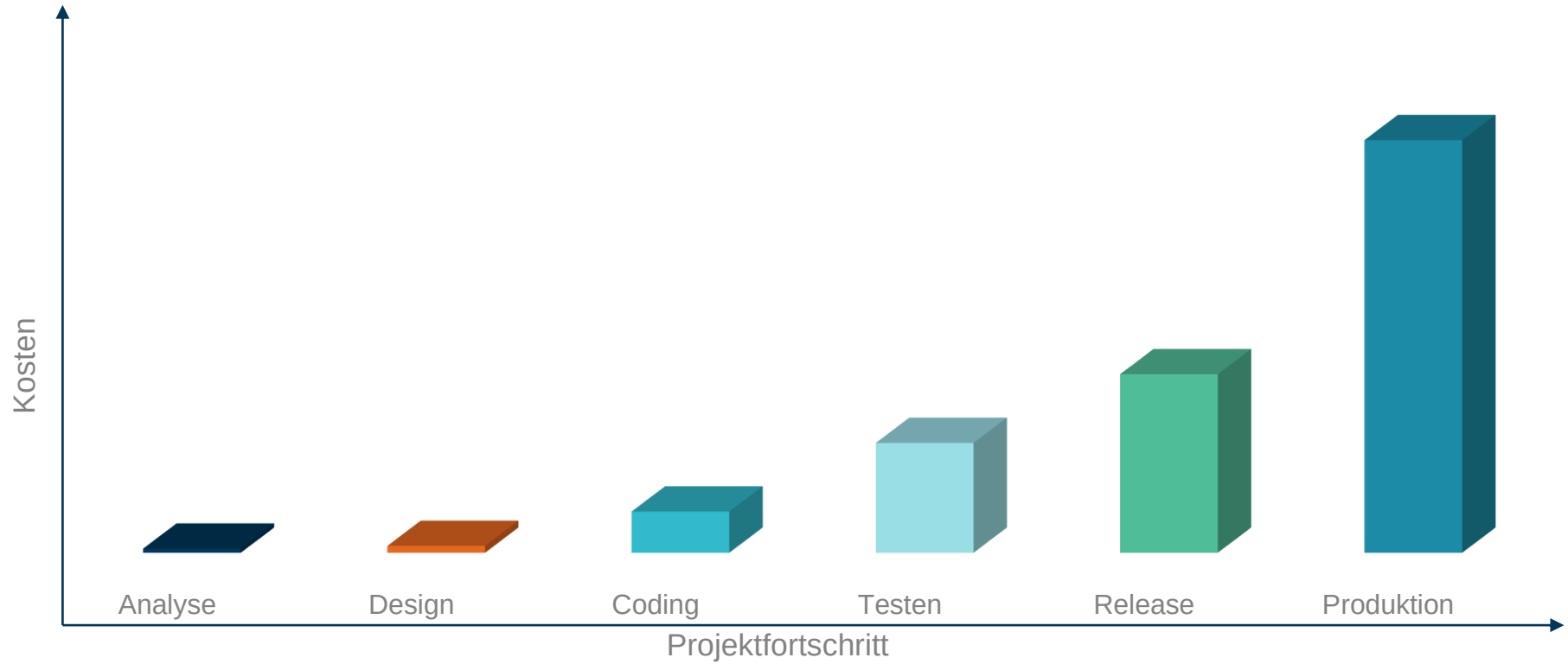
Testen vs. Debugging

Testen

- Findet Fehlerwirkungen (z.B. Error-Meldungen)
- Durch Tester oder Entwickler
- Teilweise automatisierbar

Debugging

- Findet (und korrigiert) Fehlerzustände (z.B. >= statt > im Code)
- Durch Entwickler
- Nicht automatisierbar



Wie viel kostet ein Fehler?

Relative Kosten für Fixen von Fehlern

7 Grundsätze des Software Testens

1	Testen zeigt die Anwesenheit von Fehlerzuständen	• Testen kann nicht beweisen, dass keine Fehlerzustände vorhanden sind.
2	Vollständiges Testen ist unmöglich	• Es ist unmöglich alle möglichen Kombinationen und Szenarien zu testen. • Der Testaufwand muss dem Risiko und den Prioritäten angepasst werden.
3	Frühes Testen spart Zeit und Geld	• So früh wie möglich im Software Entwicklungszyklus testen.
4	Häufung von Fehlern	• Die Mehrheit der Fehler wird durch eine geringe Anzahl von Modulen erzeugt. • Pareto Prinzip: 80% der Fehler in 20% der Module.
5	Wiederholungen haben keine Wirksamkeit	• Wiederholtes Ausführen von Tests bringt i.d.R. keine neuen Erkenntnisse. • Pestizid-Paradoxon: Alte Testfälle sind für neuere Versionen nicht mehr gültig.
6	Testen ist kontextabhängig	• Die Art der Applikation bestimmt die Methoden, Techniken und Typen des Testens.
7	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 negativ beeinflussen.

Wichtige Aspekte zum Testen

- Tests müssen **wiederholbar** sein.
- Tests müssen **dokumentiert** sein.
- Tests müssen **nachvollziehbar** sein.
- Tests müssen **im geplanten Zeit-rahmen durchführbar und wirtschaftlich** sein.

Unabhängig Testen

- Die Tester sollten 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.

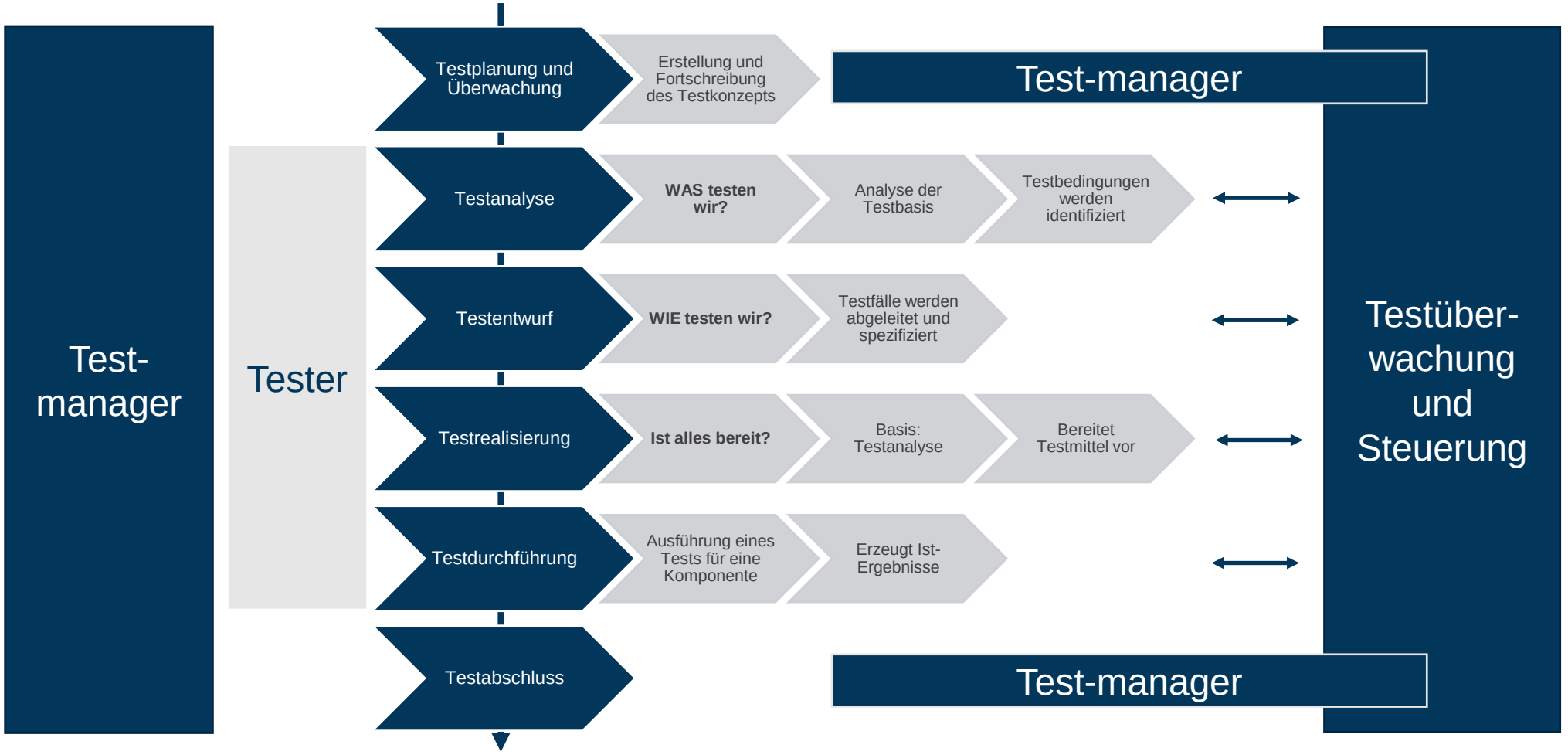
Software während des Tests statisch halten

- Die Software sollte während des Testvorgangs nicht verändert werden.

Erwartete Ergebnisse bereitstellen (wenn möglich)

- In der Testdokumentation sollten erwartete Ergebnisse spezifiziert werden.

Testprozess



Die Psychologie des Testens

- Tester vs. Entwickler
  - Verteilung von Aufwand
  - Unabhängige Sichtweise
  - „Betriebsblindheit“
- Konstruktive Kommunikation zwischen Tester und Entwickler ist essentiell:
  - Kooperation statt Konfrontation
  - Nutzen betonen
  - Objektiv argumentieren
  - Bestätigungen geben und einholen
  - Perspektivwechsel nachvollziehen

Teststufen

Abnahmetest

Test durch den Kunden.
Meist Voraussetzung für rechtswirksame Übernahme.
Häufig: Blackbox-Verfahren

Arten	UAT, vertraglich/regulatorisch, Alpha/Beta
Objekte	System-under-Test Systemkonfigurationen Geschäftsprozesse

Systemtest

Test des Gesamtsystems gegen Anforderungen (funktional + nicht-funktional)

Basis	Systemspezifikationen Risikoanalyse Anwendungsfälle und Stories
Objekte	Anwendungen System-under-Test Systemkonfigurationen

In der Praxis oft nicht scharf voneinander abgrenzbar

Integrationstest

Arten	Systemkonfigurationstest Komponentenintegrationstest
Objekte	Teilsysteme Datenbanken, APIs

Schwerpunkt auf Schnittstellen

Komponententest

Basis	Code, Datenmodelle
Objekte	Komponente (Unit), Code & Datenbankstrukturen, Datenbanken

Modul, Programm, Unterprogramm, Klasse

Black-Box- vs. White-Box-Testing

- Testdesign -

Black-Box-Testing

Testet das von außen sichtbare Verhalten des Testobjekts

- End-User-Experience
- Funktionstüchtigkeit

Testfälle werden auf Grundlage der fachlichen Spezifikation erstellt

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

Quellcode wird nicht berücksichtigt

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

Quellcode testen

Statisches Testen vs. Dynamisches Testen

Statisch

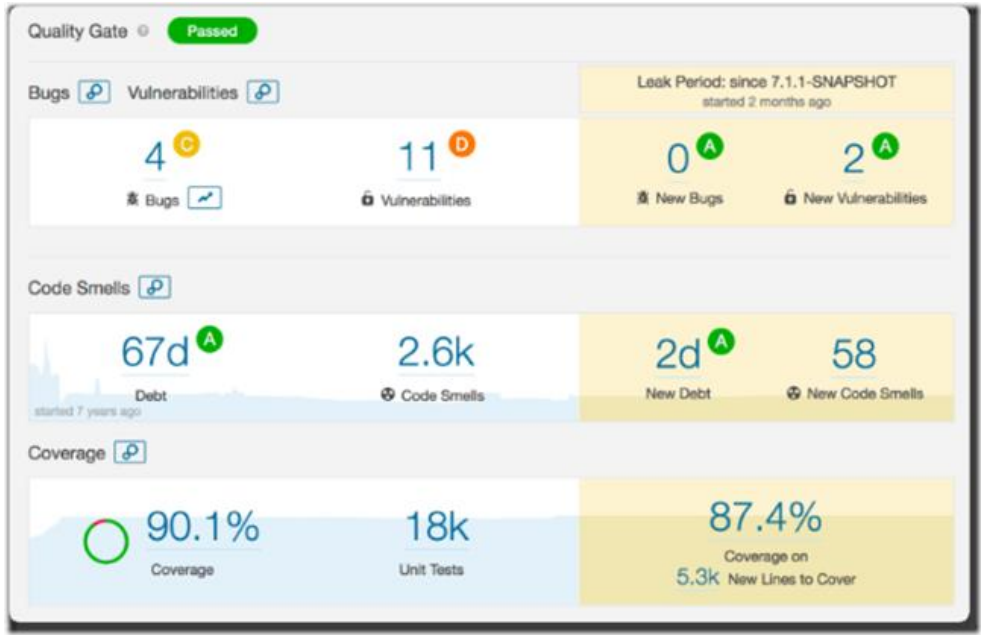
- Finden Fehlerzustände **ohne** Ausführung
- Fokus: Qualität der Arbeitsergebnisse
- Zeitpunkt: einsetzbar bevor ausführbarer Code verfügbar ist
- Auf alle Arbeitsergebnisse sehr früh anwendbar

Dynamisch

- Finden Fehlerwirkungen **durch** Ausführung
- Fokus: Verhaltensweise des Programms
- Zeitpunkt: setzen ausführbaren Code voraus
- Kaum anwendbar auf Spezifikationen, Dokumentationen, Pläne etc.

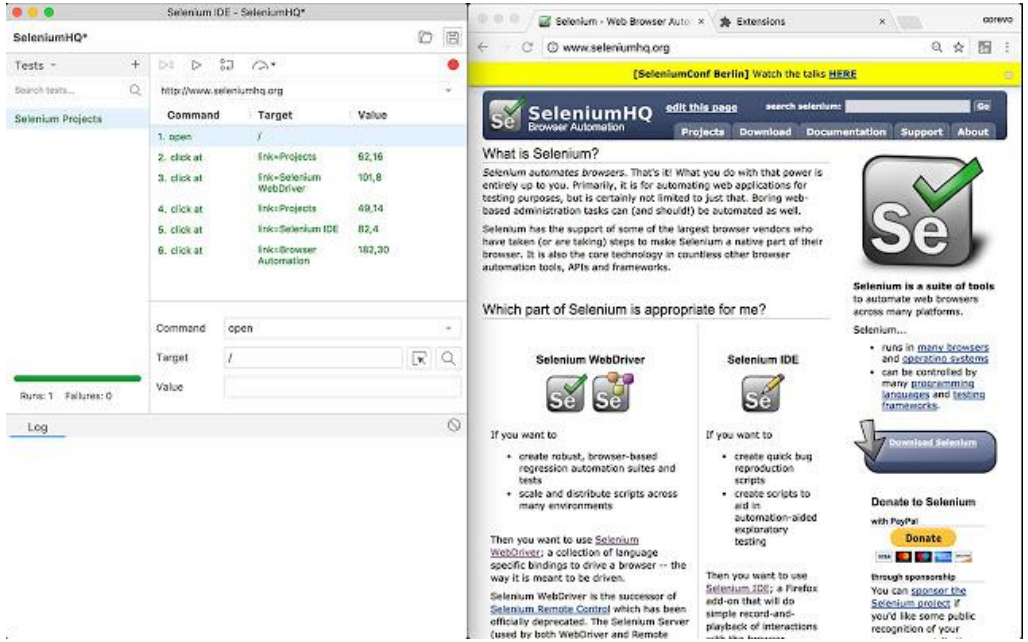
Statisches Testen vs. Dynamisches Testen - Beispiele

Statisch



https://www.sonarqube.org

Dynamisch



https://www.selenium.dev

Statisches Testen

- Statische Tests haben einige Vorteile gegenüber den Dynamischen:

Kostenersparnis durch frühes Finden von Fehlerzuständen

Zeitersparnis durch effizientere Entwicklung

Bessere Kommunikation durch Reviews

Beispiel: 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

Beispiel Testplan

Welche Testtypen sind relevant und welche nicht?	
Im Umfang	**Nicht im Umfang**
Plattformübergreifendes Testen	Browserübergreifendes Testen
Performance Testing	
Funktionales Testen	

Skala: 1-10 (1=Gering, 10=Sehr stark)

Projektrisiken und Produktrisiken festlegen			
Risiko	**Wahrschein-lichkeit**	**Auswirkung**	**Minderung**
Projektrisiko			
Senior Teammitglied verlässt das Team plötzlich	5	3	• Wissens-transfer • Ressourcen Buffer
Produktrisiko			
Das System lässt sich in der Testumgebung nicht installieren	8	10	Smoke Testing

Beispiel Testfall

Test Szenario	Testfall	Vorbedingung	Testschritte	Testdaten	Vorausgesagtes Ergebnis	Ist-Ergebnis	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

Bugfix – Nächstes Release

Test Szenario	Testfall	Vorbedingung	Testschritte	Testdaten	Vorausgesagtes Ergebnis	Ist-Ergebnis	Bestanden/ 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

TUT, FUT und UAT

- Technical User Test (TUT)
  - Vom Entwickler durchgeführt
  - Testet das Backend, d.h. den Code einer Anwendung
- Functional User Test (FUT)
  - Basiert auf der Spezifikation der zu testenden Softwarekomponenten
  - Beispiel: „Login-Funktionalität testen“
- User Acceptance Test (UAT)
  - Der erfolgreiche Abschluss dieser Teststufe ist meist Voraussetzung für die rechtswirksame Übernahme der Software und deren Bezahlung

User Acceptance Test (UAT)

- Test orientiert sich nicht am Code, sondern am Verhalten der Software bei spezifizierten Vorgängen
- Testet, ob die Software wie beabsichtigt funktioniert
- Oft Beta-Tests
- Meistens von Kunden oder Endnutzern durchgeführt
- Konzentriert sich auf Probleme wie Rechtschreibfehler, kosmetische Probleme und Softwareabstürze
- Kann u.U. bereits auf der Produktionsumgebung mit Kopien von Echtdaten durchgeführt werden

Aufgabe!

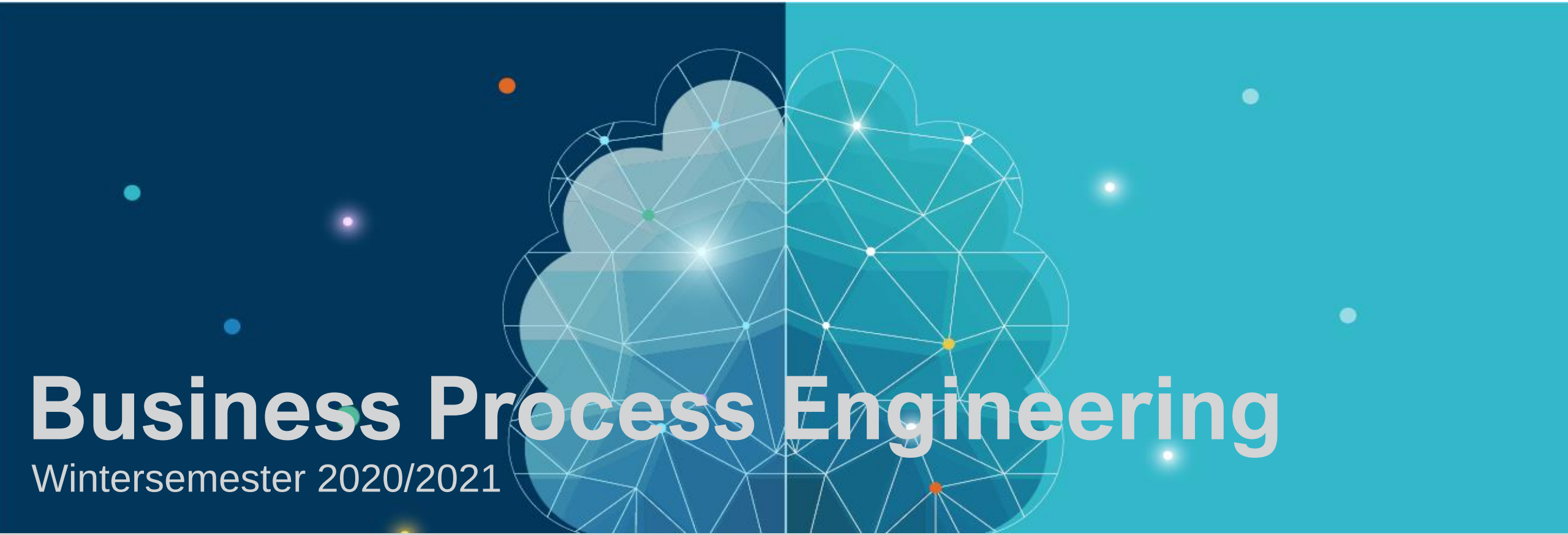
- Füllt die Checklisten TUT und FUT für die in den Aufgaben 8 und 9 entwickelte Software aus.
  - Wird morgen Abend gestellt (29.01.2021)
  - Donnerstag 04.02.2021, 14:00 Uhr (1. Abgabe)
  - Sonntag 07.02.2021, 23:59 Uhr (2. Abgabe)

Quellen

- https://deepsource.io/blog/exponential-cost-of-fixing-bugs/
- https://blog.seibert-media.net/blog/2011/05/16/software-tests-notwendigkeit-arten/
- https://www.oliver-lampert.at/glossar/testarten/
- 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

UNIKASSEL VERSITÄT

octaVIA
SAP-Kompetenz und Branchenwissen



Business Process Engineering

Wintersemester 2020/2021

Dr. Andreas Scharf