



Software-Engineering-Praxis

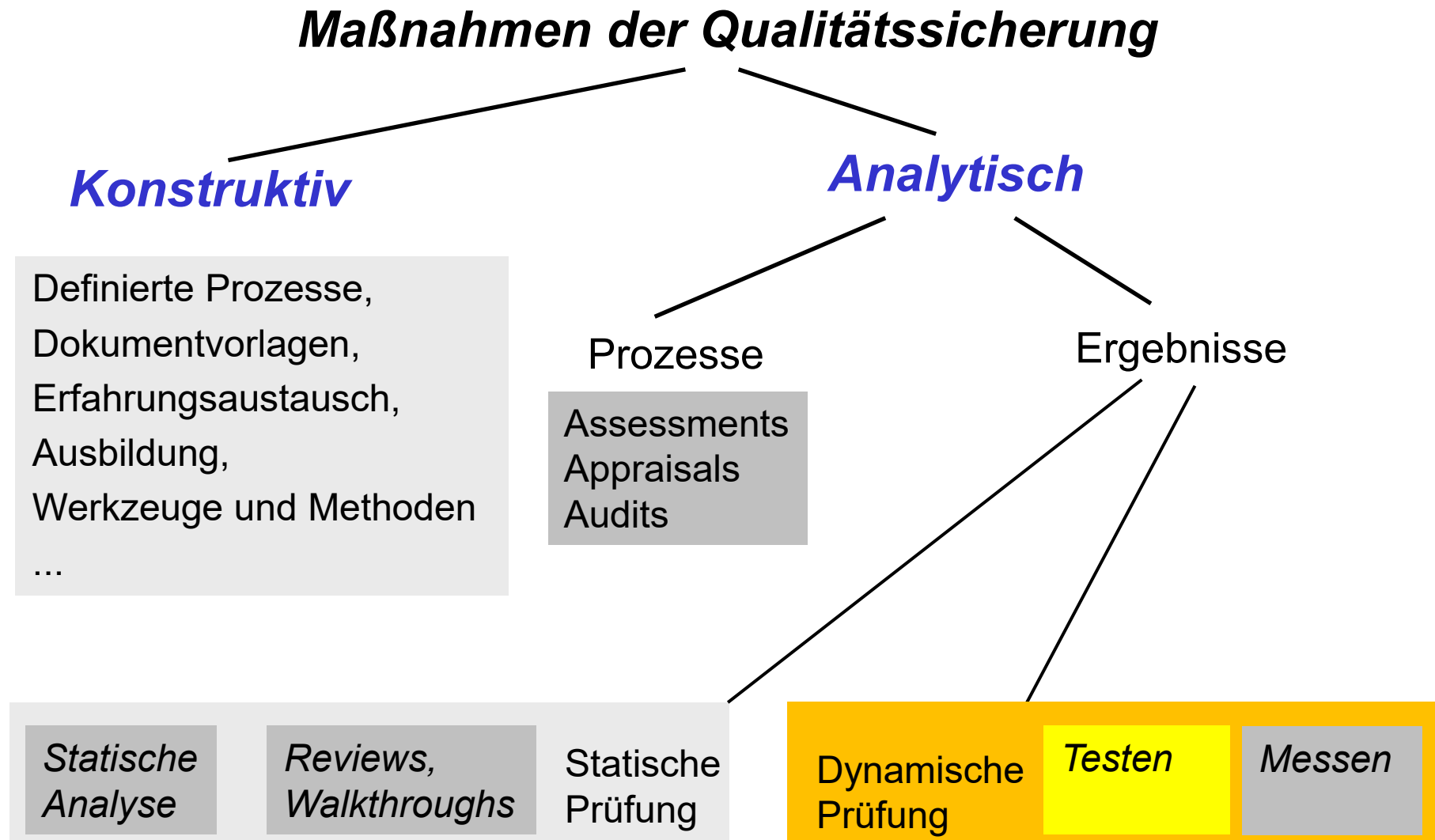
Prof. Dr. Gerd Beneken

Kapitel 12.1

Einführung in Softwaretest

Produktrisiken mindern

Qualitätssicherung Begriffe



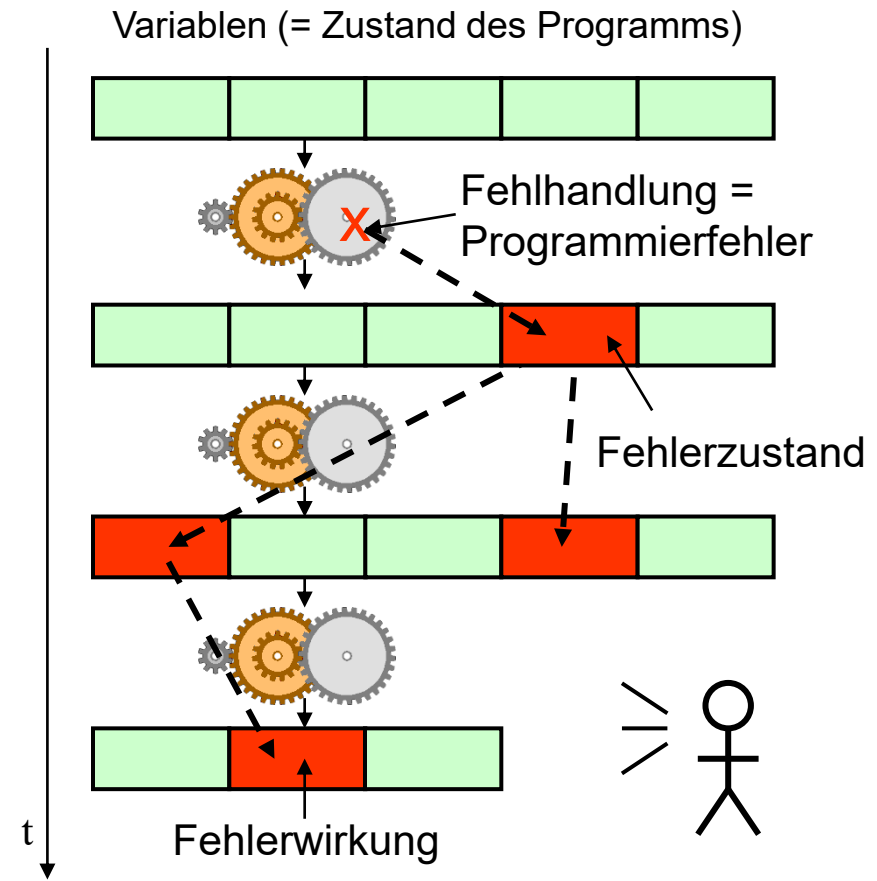
Was ist Testen?

- **Systematische Ausführung** eines **Testobjekts**
- Ziele
 - **Verifikation** der Korrektheit auf Grundlage der Spezifikation
"Das System richtig entwickelt"
 - **Validierung** des System relativ zu den Kundenanforderungen
"Das richtige System entwickelt"
 - Regressionsfähigkeit (um leicht ändern zu können)
- Nicht-Ziele
 - Finden der Ursachen von Fehlern (Fehlhandlung, Fehlerzustand)
 - = Debugging
- Testobjekt = Komponente, integriertes Teilsystem, oder System, das dem Test unterzogen wird
- Voraussetzung für Tests: eine Spezifikation / ein Kunde (Festlegung des Sollverhaltens)

Fehlerfortpflanzung: Von der Fehlhandlung zur Fehlerwirkung

1. Programmierer macht Fehler
(=**Defect**, **Error**)
2. Bei Ausführung: Fehler infiziert
Programmzustand
(=**fault**, **Infection**)
3. Infektion pflanzt sich fort im Programm-
zustand
4. Infektion erzeugt eine Fehlerwirkung
(=**Failure**)

Nicht jeder Programmierfehler erzeugt eine Fehlerwirkung



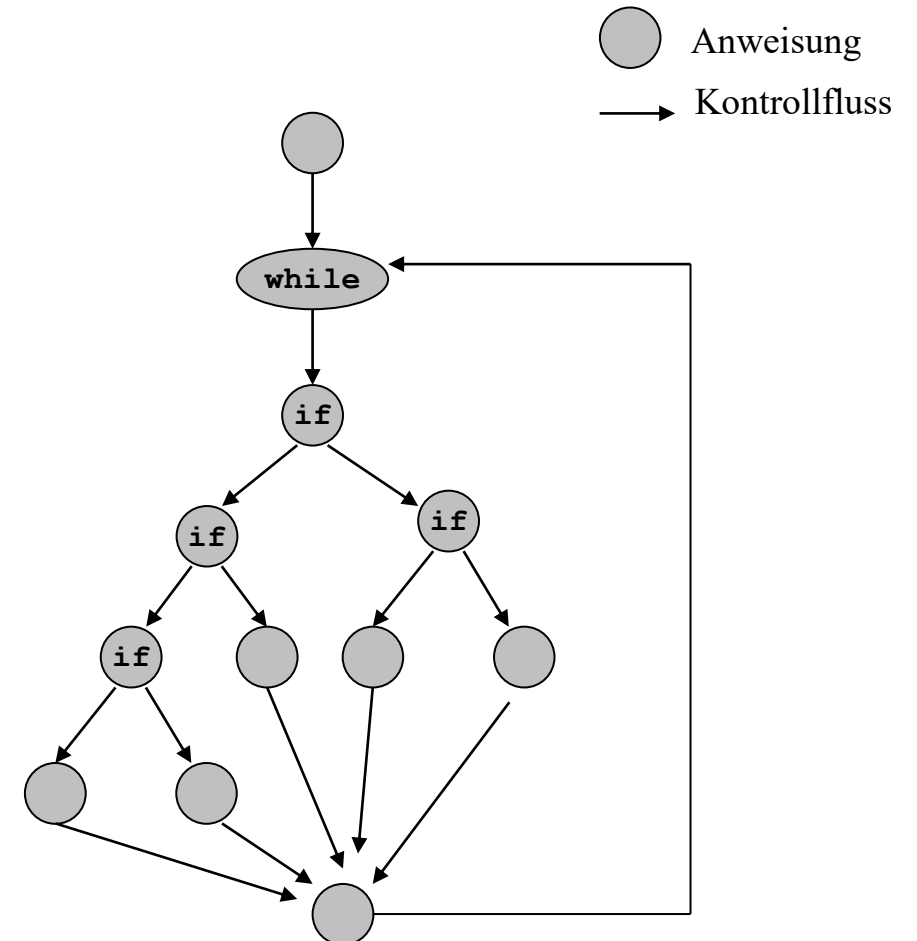
Warum Software Test?

- Durchschnitt: **1-3 Fehler pro 1000 Anweisungen** im Quelltext bei ausgelieferter Software
- Beispiel aus BMBF Studie (2001):
 - Wert aller Software-Systeme in Deutschland: **25 Mrd. €**
 - Quelltext: ca. **1,25 Mrd. LOC**
 - Bei 3 Fehlern pro 1000 LOC: **3,75 Millionen Fehler**
 - Behebung eines Fehlers: 13 Std. durchschnittlich
 - Bei einem Stundensatz von 100 €: **4,8 Mrd. €**, = 18% des Wertes
- Schaden durch fehlerhafte Software US-Autoindustrie (2000): 1,8 Mrd. \$
- Derartige Fehler im Airbag-Controller oder AKW Steuerung?

Unabhängig vom Testaufwand: Vollständiger Test ist nicht möglich

- Grafik = Kontrollflussgraph
- Beispiel:
 - 4 **if** Anweisungen (insgesamt 5 mögliche Durchläufe)
 - **while** – schleife mit bis zu 10 Durchläufen
- Mögliche Varianten =
$$5^{10} + 5^9 + 5^8 \dots + 5$$

$$= 12207030$$



-> Über Tests können keine Eigenschaften „bewiesen“ werden



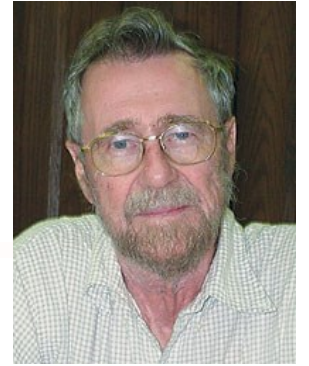
Software-Engineering-Praxis

Prof. Dr. Gerd Beneken

Kapitel 12.2

Teststufen

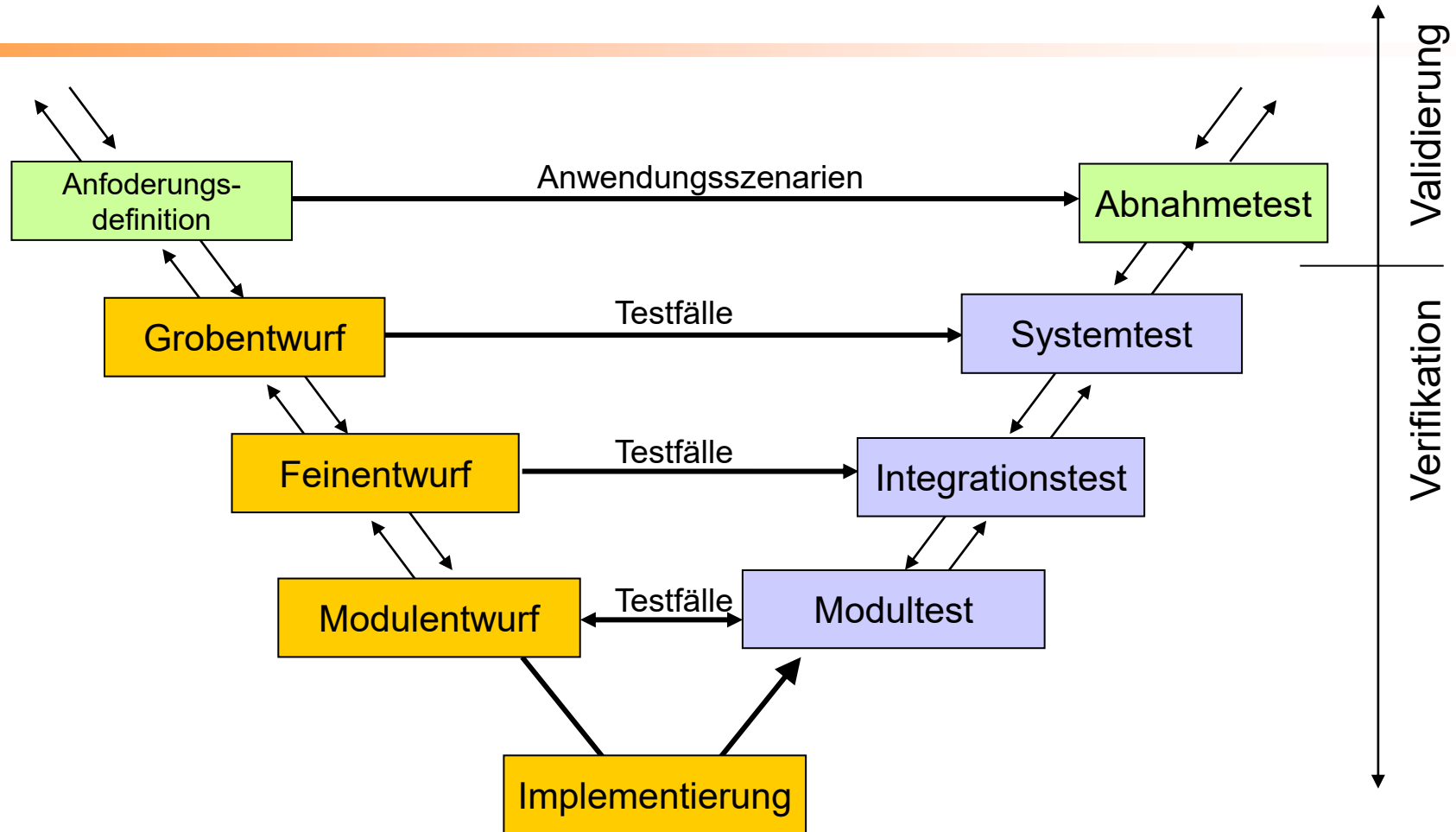
Dijkstra's Law



E. W. Dijkstra

***Testing can show the presence
but not the absence of errors***

Teststufen (V-Modell nach B.W. Boehm, 1979)



Modul/Unit/Komponenten-Test (enthalten in der DoD)



K.Beck



E.Gamma

- Testobjekt = kleine Komponenten, Klassen, Module
 - wird (weitgehend) isoliert getestet
- Testumgebung = Umgebung eines Entwicklers
 - Test im Rahmen der Entwicklung, **vor Checkin**
 - Automatisiert über einen / mehrere **Testtreiber** (JUnit, NUnit, TestNG,...), regressionsfähig
 - Häufig Dummy-Umgebungskomponenten (Mock-Objekte)
 - Häufig nur wenige Dummy-Daten, ein Client
- Testziele:
 - Nachweis der Korrektheit / Vollständigkeit der Implementierung
 - Nachweis der Robustheit (Negativ-Test!)
 - Ggf. erster Effizienzeindruck (Achtung!! nicht zu früh mit Tuning beginnen)
- Häufige Form: Test Driven Development (Test-First)

Integrationstest

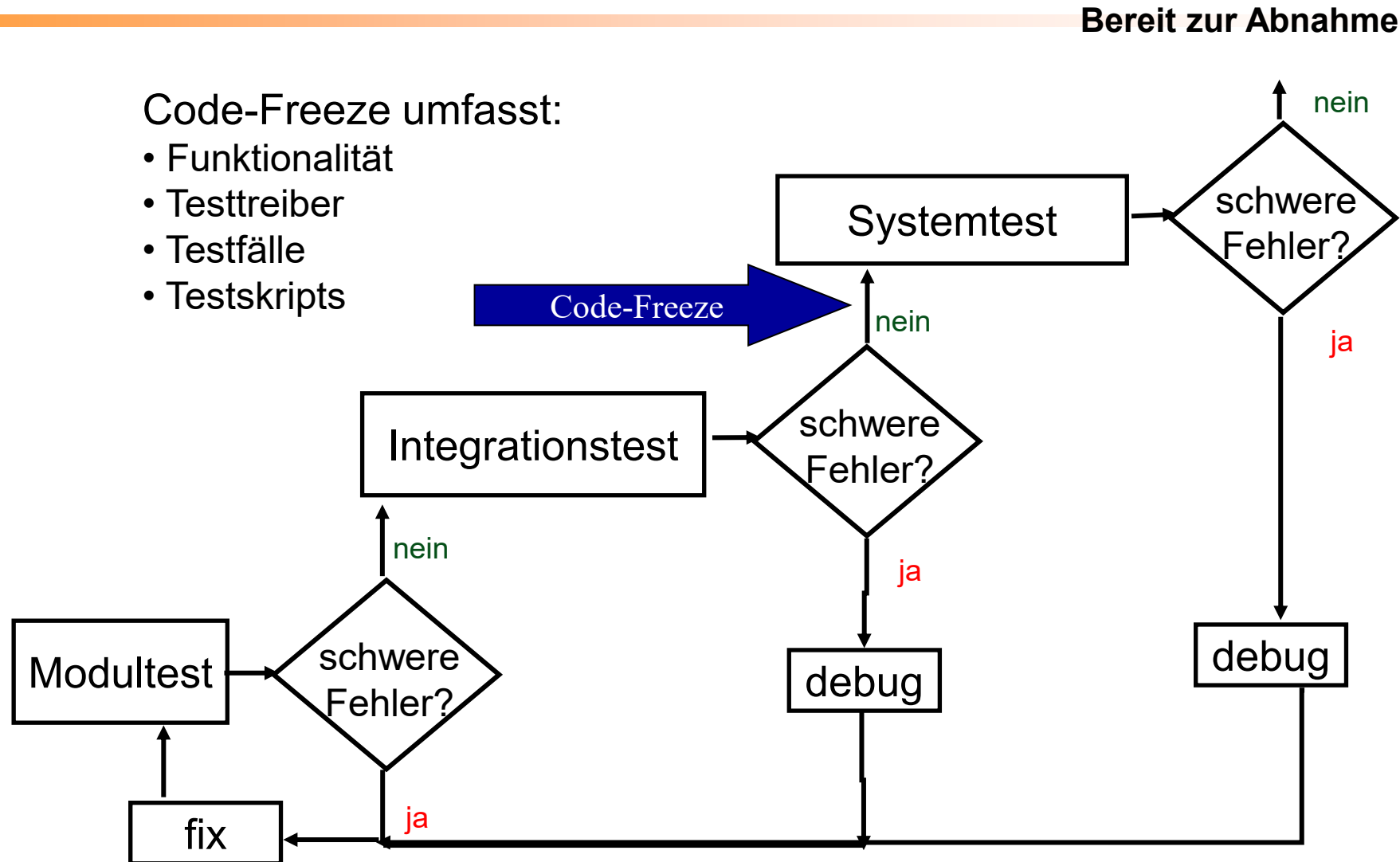
Kann als Teil des Akzeptanztests gesehen werden

- Testobjekt = (Teil-) integriertes System
 - z.B. Softwareanteil vollständig, Hardware noch anders
 - z.B. Software integriert mit Altsystem / Nachbarsystemen
- Testumgebung
 - Integrationsrechner / - umgebung, ggf. Entwicklerrechner
 - (Teil-)automatisiert über Testtreiber / Testsuite (JUnit, BDD)
 - Populär: ATDD/BDD mit Cucumber und anderen Werkzeugen (Given When Then)
 - Ggf. Analysewerkzeuge wie Protokoll-Sniffer / Monitore
- Testziele
 - **Test des Zusammenspiels der Komponenten**
 - **Aufdecken von Schnittstellen/Protokoll-Fehlern**
(z.B. Fehlende Daten in der Übertragung, Fehlerbehandlung fehlt)
 - = großes Problem bei nebenläufigen Systemen
- Integrationstest ggf. in mehreren Stufen, je nach Komplexität des Systems (vgl. Web-Shop vs. Airbus A380)

Systemtest

- Testobjekt = Gesamtsystem unter Produktivbedingungen
 - System wird aus der Perspektive des Kunden betrachtet
- Testumgebung
 - **produktionsnahe Umgebung**, soweit möglich (Datenvolumen, Clientzahl, Hardware, Einsatzfeld)
 - Noch beim **Auftragnehmer**
 - Teilautomatisierung z.B. über
 - GUI-Testtools: Selenium/Appium, Coded UI, Cypress
 - Last-Testtools: Gatling, JMeter, ...
- Testziele
 - Durchtesten der Spezifikation (Anforderungen korrekt umgesetzt?)
 - Durchtesten der nichtfunktionalen Anforderungen (Lasttest, Performancetest, Stresstest, Test auf Datensicherheit, Test der Benutzerfreundlichkeit, ...)

Zusammenspiel der Tests



Abnahmetest bei Werkverträgen

- = Kernelement der Abnahme durch den **Auftraggeber**
- Testobjekt = Gesamtsystem unter **Produktivbedingungen**
- Testumgebung
 - **Produktivumgebung** / Produktionsnahe Umgebung **beim Auftraggeber**
 - Reale Daten, reales Datenvolumen, reale Clientzahl, reale Einsatzbedingungen (bei Hardware z.B. Arktis, Wüste, Urwald, ...)
- Testziele
 - Prüfung der Erfüllung des Vertrags (Nachbesserungen?)
[Voraussetzung: Abnahmekriterien/Akzeptanzkriterien definiert!]
 - Benutzerakzeptanz (Probebetrieb)
 - Feldtest (z.B. durch Alpha-/Beta-Tester, besondere Kunden)
 - Risikominderung bei Einführung des neuen Systems



Software-Engineering-Praxis

Prof. Dr. Gerd Beneken

Kapitel 12.3

Testen – Systematische Planung und Durchführung

Testen: Grundsätzlicher Ablauf

Integrationstest, Systemtest, Abnahmetest

1. Testplanung
2. Testspezifikation
3. Testdurchführung
4. Testprotokollierung und Fehlerverfolgung
5. Auswertung und Test-Management
(Überwachung/Steuerung)



Software-Engineering-Praxis

Prof. Dr. Gerd Beneken

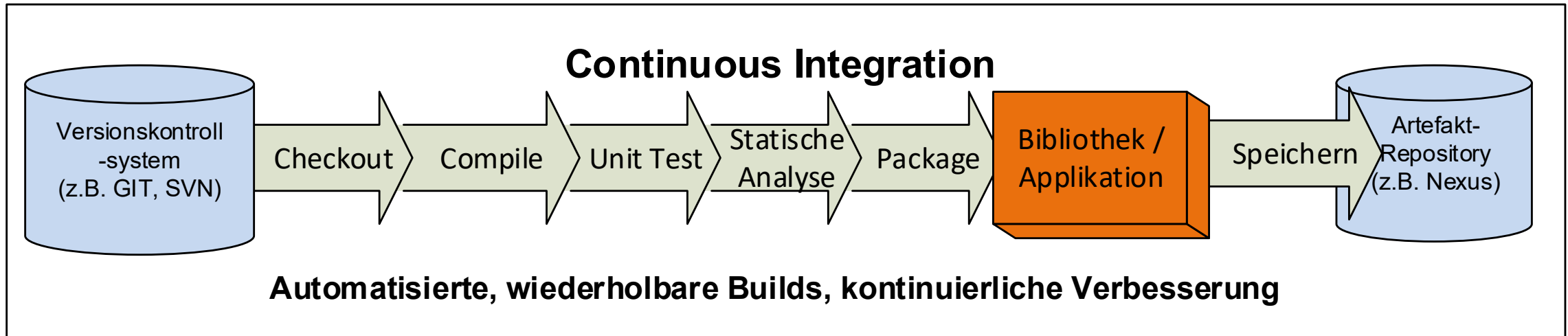
Kapitel 12.3.1

Testplanung

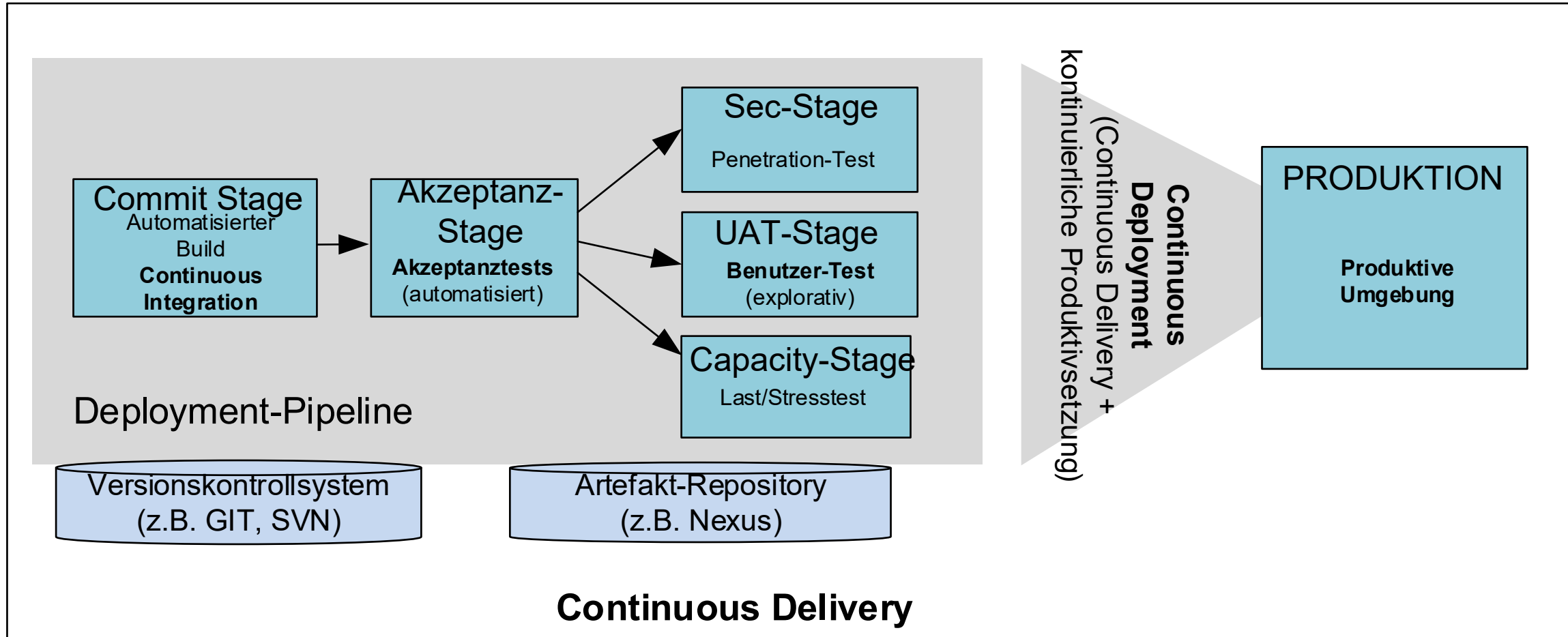
1. Testplanung

- Festlegung der Teststrategie
 - Testziele
(*Welche Risiken bekämpfen?*)
 - Prioritäten
(*Was wird zuerst / intensiv getestet?* Testobjekte?)
 - Umfang der Tests
 - Bestimmung der *Testverfahren und Werkzeuge*
 - Ablauf: Testen mehreren Test/Bugfix Iterationen?
- Bestimmung der Ressourcen
 - Mitarbeiter (Wer, Wann, Wie lange,...?)
 - Räume, Rechner, Netzwerke, Datenbanken, Lizenzen, ...
- Ergebnis = Testkonzept /Test Plan

Testplanung – Continuous Integration anwenden (Buildpipeline)

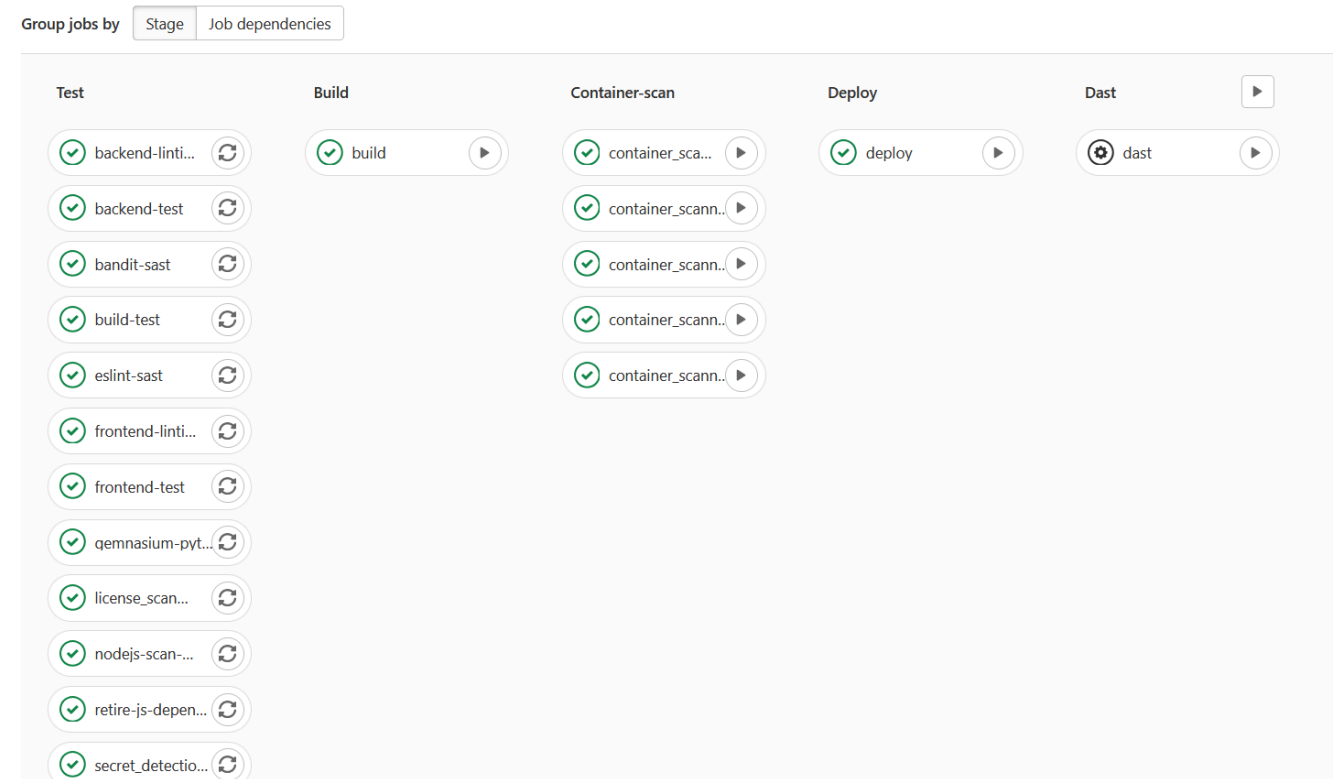


Testplanung: Continuous Delivery verwenden (Buildpipeline)

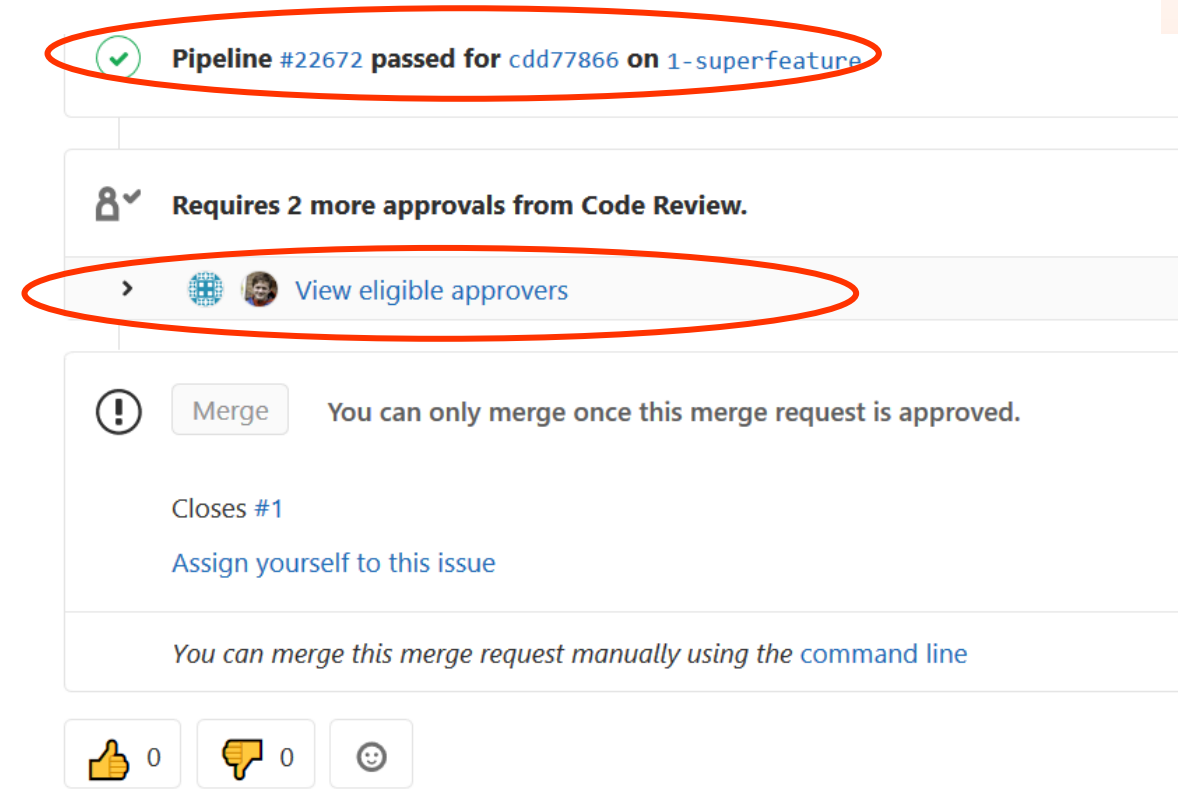
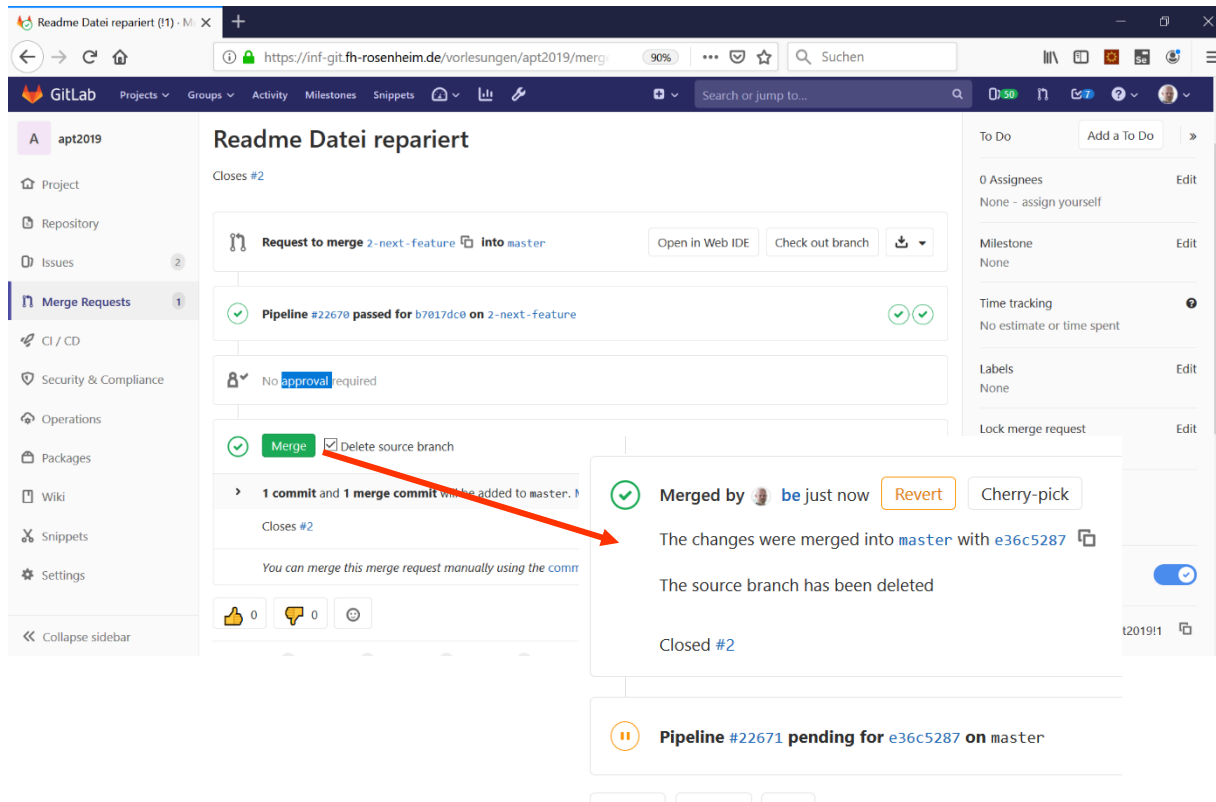


Gitlab Unterstützung in der Build Pipeline (Auto DevOps)

- Testautomatisierung (Unittests, GUI Tests)
- Testcoverage
- Statische Code-Analyse (Techn. Schulden, Sicherheitslücken / Verwundbarkeiten, Code-Duplikate, ...)
- Scan der Open Source Lizenzen
- Scan auf Sicherheitslücken in Bibliotheken
- Scan auf Sicherheitslücken in Containern



Absicherung durch Merge-Requests Manuelles Code Review und Genehmigung



Exploratives Testen

Eigenschaften des Produkts				
		funktioniert	ist robust	ist effizient benutzbar
Was können die Personas mit dem Produkt tun?	Sich als Partner anmelden	Erforschen der ... mit ... um Überraschungen zu finden	Erforschen der ... mit ... um nicht abgefangene Bedienfehler zu finden	Erforschen der ... mit ... um zu aufwendige und unverständliche Dialoge zu finden
	Ein Projekt vorschlagen			
	Projekte verwalten (Prof.)			
	Sich für ein Projekt bewerben			

Testplanung - Vorüberlegungen

	Werkzeug	Erfolgsmaß	Wo/Wann
■ Funktionsumfang	JUnit-Tests am Backend, Datenbank gemockt, mit JaCoCo	80% Zweigüberdeckung	Jeder Push auf jeden Branch, in Pipeline
	Cypress UI-Tests am Frontend	Smoketest, jeder Dialog einmal durchlaufen	Bei Push auf Develop Branch in Pipeline
	User Storys mit Postman	Jede User-Story über Postman-Collection darstellen	Bei Push auf Develop Branch, in Pipeline
	Nutzerakzeptanztest, manuell Explorativ, auf basis Test Charters für die wichtigsten Funktionsbereiche	Nutzerakzeptanztest, manuell explorativ	Vor Inbetriebnahme Von STAGE auf PROD
■ Wartbarkeit	SonarQube verfolgt Testcoverage, Technische Schulden, Code Duplikate (Rosenheim-Profil)	0 Technische Schulden Maximal 10% Duplikate Keine Vulnerabilities / Bugs	Jeder Push auf jeden Branch, in Pipeline Bei Push auf Develop
	Arbeit mit Feature Branches, dort Merge-Request stellen, manuelles Code-Review	Anmerkungen zum Merge Request	Branch, manuell (Merge Request)



Software-Engineering-Praxis

Prof. Dr. Gerd Beneken

Kapitel 12.3.2

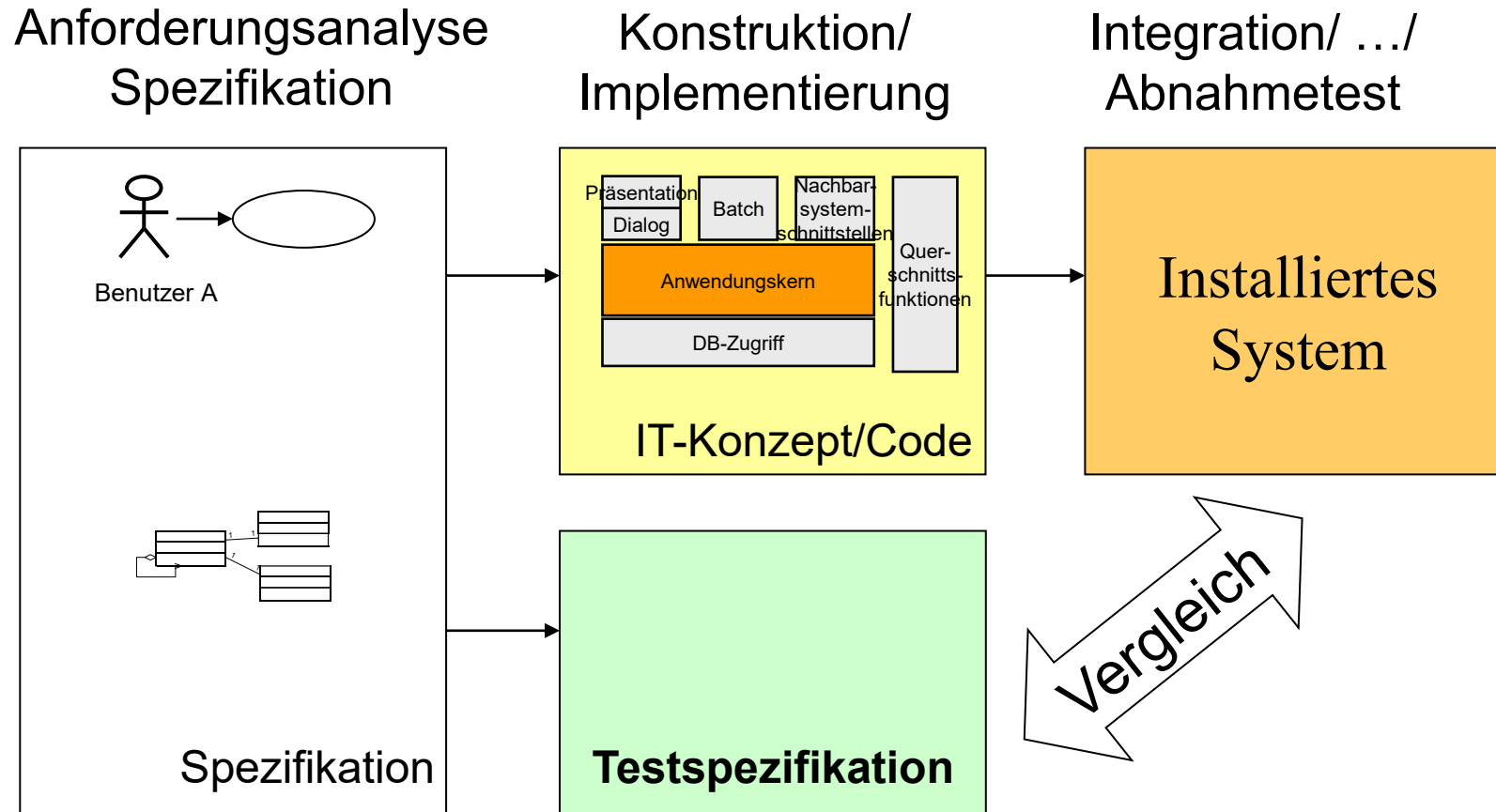
Testspezifikation

2. Testspezifikation

- = Anwendung der festgelegten Testmethoden
 - Blackbox oder Glassbox Tests? Automatisierung?
- Testfall
 - = Schritt für Schritt Anleitung für einen menschlichen Tester
 - = Programm für ein Testwerkzeug (z.B. JUnit, Selenium, ...)
 - Spezifiziert konkrete **Eingaben** (auch Fehleingaben) und die erwartete **Reaktion des Programms und Nachbedingungen**
 - Spezifiziert **Rahmenbedingungen** (Datenbasis, Hardware, Vorbedingungen, ...)
 - Spezifiziert ggf. detaillierte Prüfanweisungen
 - Ggf. Unterscheiden: Logische und konkrete Testfälle
- Testspezifikation = Menge aller Testfälle

Rolle der Testspezifikation

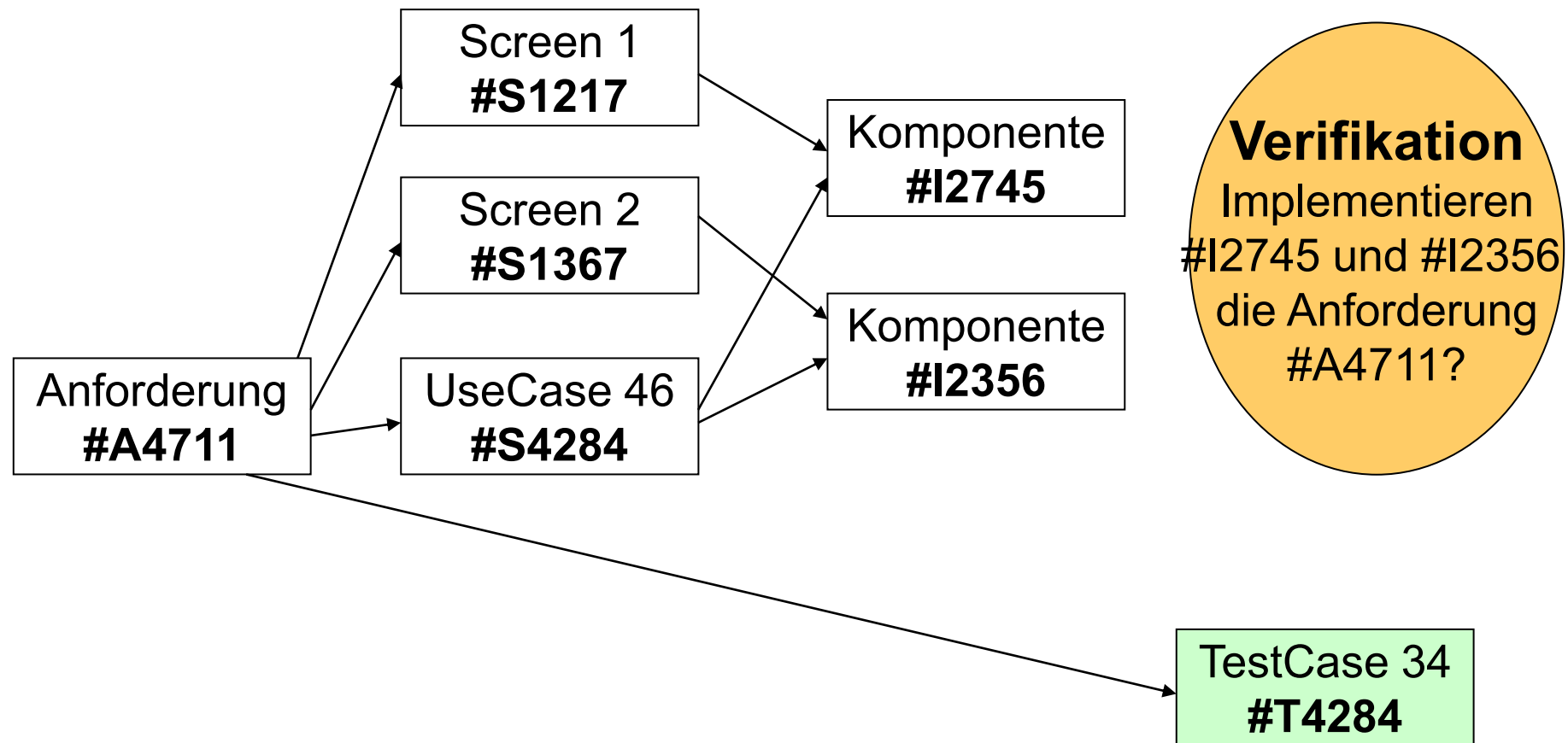
Kette der Anforderungsverfolgung



Rolle der Testspezifikation

Kette der Anforderungsverfolgung

/2



Derartige Ketten sind z.B. bei FDA-Zertifizierung nachzuweisen!

Beispiel für Testfallspezifikation

- **Testname: „Partner anlegen“ (#7)**
- **Vorbedingungen:**
 - Benutzer Petra Partner loggt sich erstmalig in das System ein
 - Petra Partner ist im System noch nicht vorhanden
 - Die Firma "Personalberatung Petra" ist ebenfalls noch nicht angelegt

The screenshot shows a web browser window titled 'Projektvergabesystem Rosenheim'. The page is for 'Registrierung Projektvergabe'. It has two main sections: 'Ansprechpartner' and 'Unternehmen'. The 'Ansprechpartner' section contains fields for 'Vorname', 'Titel', 'Nachname', and a 'Rolle' dropdown menu (currently showing 'Geschäftsführung'). The 'Unternehmen' section contains fields for 'Name', 'Strasse & Hausnummer', 'PLZ', 'Ort', and 'Land'. At the bottom is a blue button labeled 'Registrierung abschließen'. Two yellow sticky notes with red tabs are attached to the right side of the form. The first note points to the 'Rolle' dropdown with the text 'Welche Rollen wollen wir hier unterstützen?'. The second note points to the 'Strasse & Hausnummer' field with the text 'Hier fehlt noch die Möglichkeit, das Logo hochzuladen!'.

Projektvergabesystem Rosenheim

Registrierung Projektvergabe

Ansprechpartner

Vorname Titel

Nachname

Rolle

Geschäftsführung

Unternehmen

Name

Strasse & Hausnummer

PLZ Ort

Land

Registrierung abschließen

Welche Rollen wollen wir hier unterstützen?

Hier fehlt noch die Möglichkeit, das Logo hochzuladen!

Beispiel für Testfallspezifikation

Was genau ist wie zu tun?

Ablauf

- ☐ 1. Eingeben: Vorname "Petra", Nachname "Partner", Titel "Dr.", "Rolle Geschäftsführerin", Email: "petra@personalberatung.example.com"
- ☐ 2. Eingeben: Unternehmensname "Personalberatung Partner" eingeben
- ☐ 3. Eingeben: "Anmelden" drücken
- ☐ 4. Prüfen: Fehlermeldung: "Bitte Straße, Postleitzahl und Ort des Unternehmens ausfüllen"
- ☐ 5. Eingeben: "Hochschulstr. 1", "83024", "Rosenheim"
- ☐ 6. Eingeben: "Anmelden" drücken
- ☐ 7. Prüfen: Unternehmen ist angelegt (Bestätigungsfenster)

Konkrete Daten angeben !

Beispiel für Testfallspezifikation

Nachbedingungen / Prüfkriterien

- ☐ In der Partnerliste ist die "Personalberatung Petra" sichtbar
- ☐ An die Benutzerin "Petra Partner" hat eine Email erhalten, die den Kontakt bestätigen soll



Software-Engineering-Praxis

Prof. Dr. Gerd Beneken

Kapitel 12.3.3

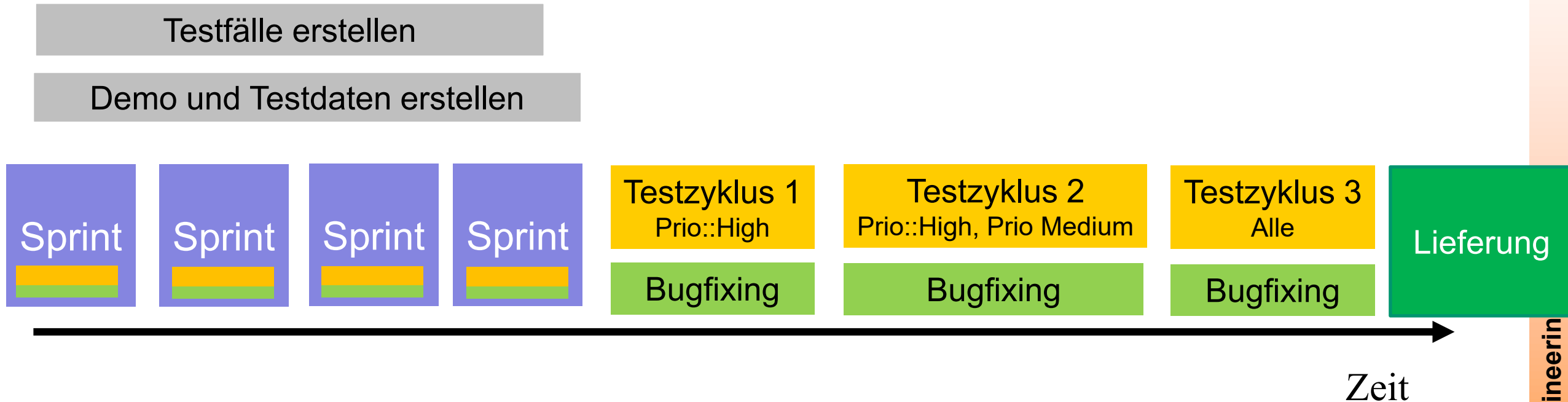
Testdurchführung

3. Testdurchführung

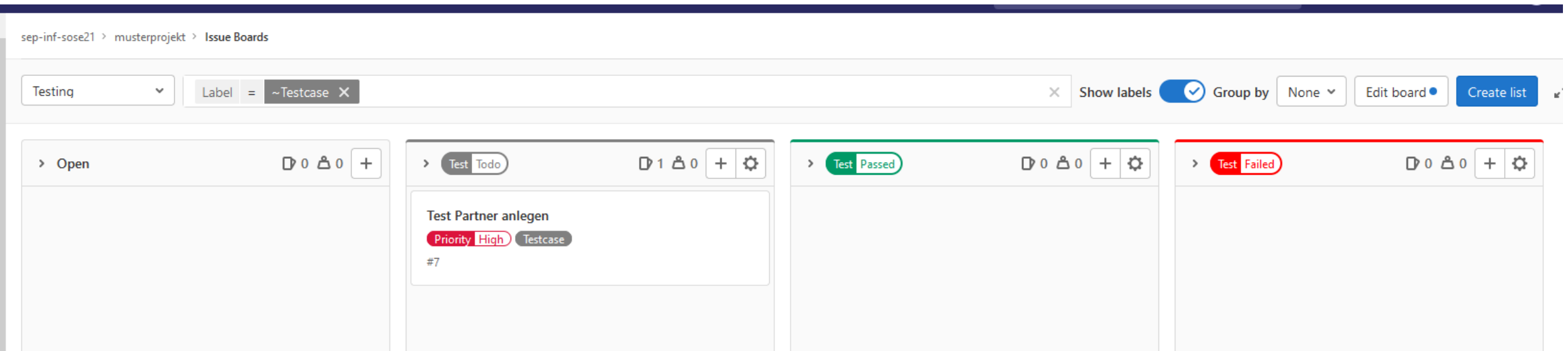
- Herstellung der Rahmenbedingungen
 - Hardware, Datenbasis, Voraussetzungen, ...
- Installation / Beschaffung des Testobjekts
 - Z.B. Baseline aus KM-Werkzeug auschecken und bauen
 - z.B. Letztes Build vom Buildwerkzeug beschaffen
 - Z.B. aktuellen Docker-Container starten
- Häufig iteratives Vorgehen
 - Code Freeze, dann Baseline erzeugen, dann
 - Erster Testlauf nur Prio 1 Testfälle, dann Bugfixing
 - Zweiter Testlauf: Prio 1 und Prio 2 Testfälle, dann Bugfixing
 - Dritter Testlauf: Alle Testfälle, dann Bugfixing
- Keine Weiterentwicklung des Testobjekts und kein Bugfixing während der Tests

Durchführung funktionaler *System*test

Funktionale Richtigkeit z.B. in drei Zyklen



Gitlab: Eigenes Board für das Thema Testen



- Eigenes Board für die Durchführung der Tests für die Testphase
- Sinnvoll ggf. auch für die Stabilisierung in jedem Sprint
- Idee (von den Gitlab Autoren) Schnelle Übersicht durch drei Labels – Todo, Passed, Failed



Software-Engineering-Praxis

Prof. Dr. Gerd Beneken

Kapitel 12.3.4

Testprotokollierung

4. Testprotokollierung

Testprotokollierung

- = Nachweis, dass Test ausgeführt wurde
- Ziele
 - Nachprüfbarkeit
 - Fehlerverfolgung
 - Grundlage für Debugging / Bugfixing
- Informationen:
 - Testobjekt (Version, Komponente, ...)
 - Testende Person / Testendes Werkzeug
 - Umgebung, Rahmenbedingungen
 - Details zum Testablauf
(Fehler, Erfolgreiche Ausführung einzelner Schritte)

Beispiel Testprotokoll

Test Partner anlegen

Vorbedingungen

- Benutzer Petra Partner loggt sich erstmalig in das System ein
- Petra Partner ist im System noch nicht vorhanden
- Die Firma "Personalberatung Petra" ist ebenfalls noch nicht angelegt

Ablauf

- ☒ 1. Eingeben: Vorname "Petra", Nachname "Partner", Titel "Dr.", "Rolle Geschäftsführer
- ☒ 2. Eingeben: Unternehmensname "Personalberatung Partner" eingeben
- ☒ 3. Eingeben: "Anmelden" drücken
- ☒ 4. Prüfen: Fehlermeldung: "Bitte Straße, Postleitzahl und Ort des Unternehmens ausf
- ☒ 5. Eingeben: "Hochschulstr. 1", "83024", "Rosenheim"
- ☒ 6. Eingeben: "Anmelden" drücken
- ☐ 7. Prüfen: Unternehmen ist angelegt (Bestätigungsfenster)

Nachbedingungen / Prüfkriterien

- ☐ In der Partnerliste ist die "Personalberatung Petra" sichtbar
- ☐ An die Benutzerin "Petra Partner" hat eine Email erhalten, die den Kontakt bestätiger


Edited right now by be

Drag your designs here or [cli](#)

The screenshot shows a Jira board with a card titled "Test Partner anlegen". The card has a red "Test Failed" label, a "Priority High" label, and a "Testcase" label. The card number is #7. Below the card, there is a comment thread. The first comment is from "be" and says "@be marked the task 6. Eingeben: 'Anmelden' drücken as completed just now". The second comment is from "be" and says "@be marked the task 5. Eingeben: 'Hochschulstr. 1', '83024', 'Rosenheim' as completed just now". The third comment is from "be" and says "@be · right now Schritt 7: Das Bestätigungsfenster erscheint nicht. Das Neue Unternehmen wird nicht angelegt." The comment has a "1" icon and a smiley face icon. At the bottom of the comment thread, there is a "Write" button and a "Preview" button. The right side of the comment thread has icons for "Owner", "Smiley", "Comment", "Edit", and "More".

Fehler in die Datenbank eintragen

sep-inf-sose21 > musterprojekt > Issues > #8

Open Created just now by  be Owner Close issue ⋮

Partner Anlegen Bestätigungsfenster erscheint nicht

Nach dem Anlegen des Partners wird das Bestätigungsfenster nicht angezeigt. [Testfall](#)





📁 Drag your designs here or [click to upload](#).

Linked issues 🔗 1 +

Blocks

🕒 Test Partner anlegen #7 🕒 Wachstum ×

👍 0 🗨️ 0 😊 Oldest first Show all activity Create merge request ⌵

- 🕒 be  @be changed milestone to %Wachstum just now
- 🔗 be  @be added bug label just now
- 🔗 be  @be added Priority High scoped label just now
- 🔗 be  @be marked this issue as related to #7 right now

Write Preview B I ” </> 🔗 ☰ ☷ ☸ 📎 📅 ↵

Write a comment or drag your files here...

To Do Add a to do »

0 Assignees Edit
None - assign yourself

Epic Edit
None

Milestone Edit
Wachstum

Iteration Edit
None

Time tracking ?
No estimate or time spent

Due date Edit
None

Labels Edit
Priority High × bug ×

Weight Edit
None

Health status Edit
None

Confidentiality Edit
👁 Not confidential



Software-Engineering-Praxis

Prof. Dr. Gerd Beneken

Kapitel 12.3.5

Testauswertung

5. Testauswertung - Fehlerverfolgung

- Bei gescheiterten Testfällen
 - Tatsächliche Fehler? Fehler in Testspezifikation? Fehler bei Ausführung?
- Bei tatsächlichen Fehlern
 - Festlegung der **Fehlerklasse**
 1. Systemabsturz – Testobjekt nicht verwendbar
 2. Wesentliche Funktion fehlerhaft
 3. Funktionale Abweichung / Einschränkung
 4. Geringfügige Abweichung
 5. Schönheitsfehler
 - Erfassung des Fehlers in einem Bug/Issue Tracker
 - Ggf. weitere Testfälle im Umfeld des Fehlers spezifizieren

Beispiel

Testauswertung - Fehlerverfolgung

- Werkzeug = Bug / Issue Tracker
- Produkte
 - Jira, Redmine, ...
- Ziel: Erfassung / Verfolgung / Mgmt von Fehlern und Änderungswünschen

sep-inf-sose21 > musterprojekt > Issues > #8

Open Created just now by be Owner Close issue

Partner Anlegen Bestätigungsfenster erscheint nicht

Nach dem Anlegen des Partners wird das Bestätigungsfenster nicht angezeigt. [Testfall](#)

Drag your designs here or [click to upload](#).

Linked issues 1 +

Blocks

Test Partner anlegen #7 Wachstum

0 0

Oldest first Show all activity Create merge request

@be changed milestone to %Wachstum just now

@be added bug label just now

Open 1 Closed 0 All 1

Recent searches Label = ~bug

Partner Anlegen Bestätigungsfenster erscheint nicht

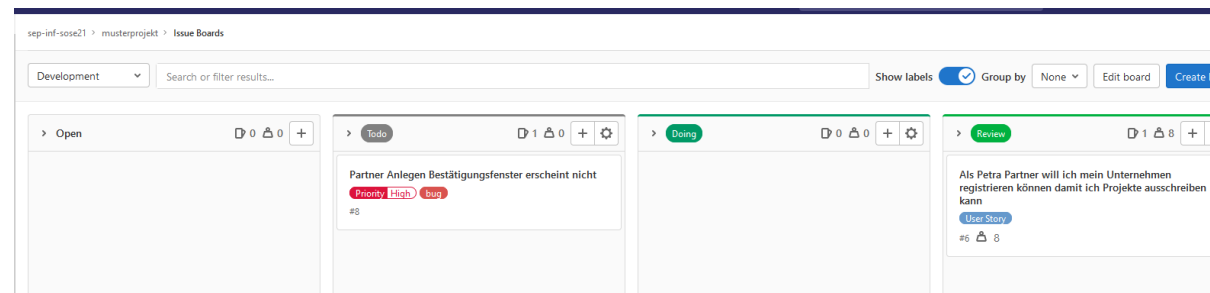
#8 · created 41 minutes ago by be Wachstum Priority High bug

updated 22 minutes ago

To Do	Add a to do	»
0 Assignees	Edit	
None - assign yourself		
Epic	Edit	
None		
Milestone	Edit	
Wachstum		
Iteration	Edit	
None		
Time tracking	?	
No estimate or time spent		
Due date	Edit	
None		
Labels	Edit	
Priority High bug		
Weight	Edit	
None		
Health status	Edit	
None		
Confidentiality	Edit	
Not confidential		

Was machen Sie mit Bug - Meldungen?

- Grundsätzlich: Triage Prozess
- Müssen wir uns sofort darum kümmern?
 - Fehler muss sofort untersucht und behoben werden
 - Patch für das Testobjekt
- Können wir uns später kümmern?
 - Bug wandert in den Product Backlog wie die anderen Anforderungen auch
 - Bug in der Sprint Planung mit eingeplant
- Wir kümmern uns garnicht
 - Fehler wird akzeptiert / Spezifikationslücke / ...

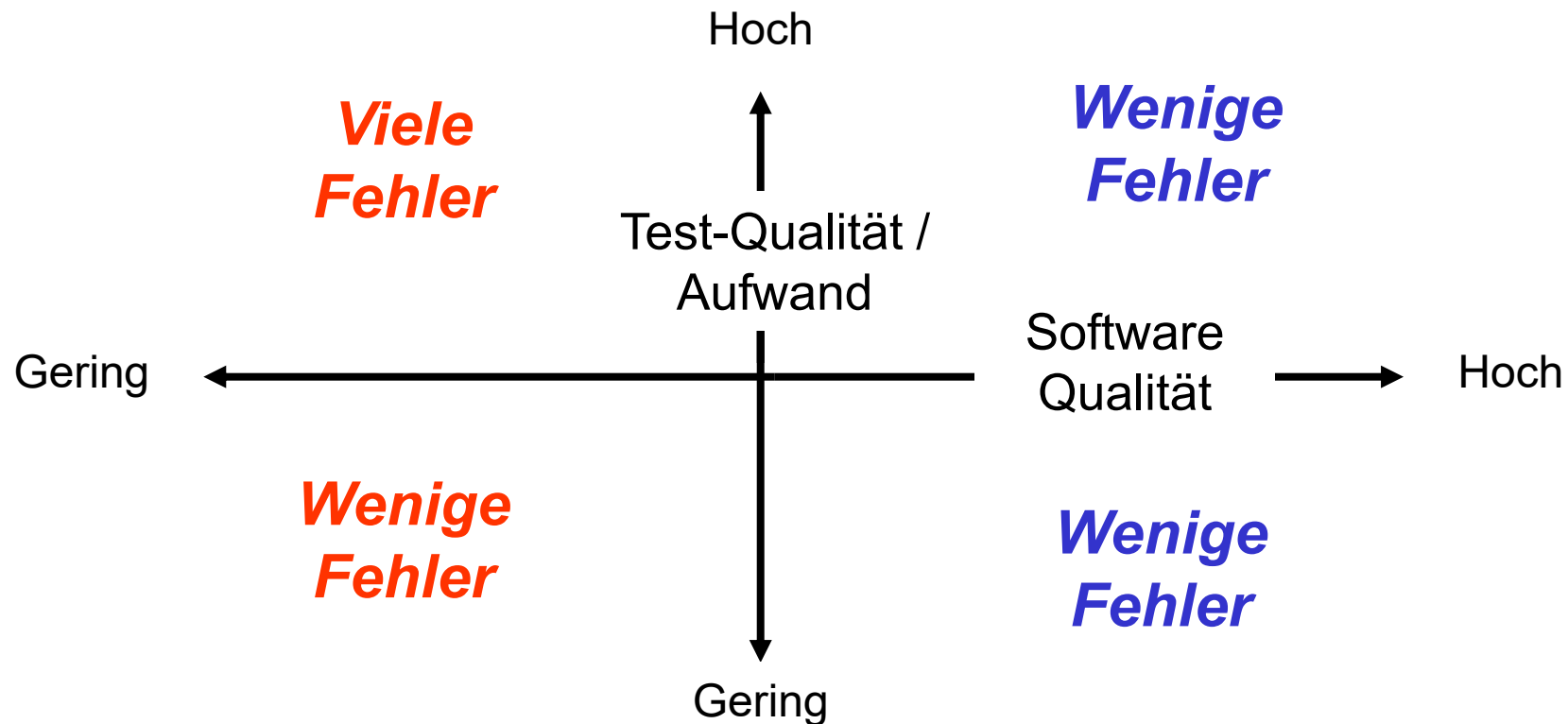


5. Testauswertung - Controlling

- Ziel: Kontrolle und Steuerung
 - Frage: Stabilisiert sich die Software gerade?
 - Frage: Ist der Testprozess effektiv / effizient?
- Statistische Auswertung der Testdurchführung
 - Zahl / Anteil der durchgeführten Testfälle pro Testobjekt
 - Zahl der gefundenen Fehler pro Testobjekt
 - Zahl der gefixten Fehler pro Testobjekt
 - Aufwand pro gefundenem / gefixtem Fehler
 - ...
- Auswertungen mit Code
 - Testüberdeckung (Code Zeilenweise / Pfadweise durchlaufen?)

Testauswertung - Controlling

Was bedeutet „Wenig Fehler“?



Zusammenfassung

- Tests zeigen
 - was funktioniert
 - die vorhandene Qualität
- Testen soll Fehler feststellen
 - bevor sich der Kunde darüber ärgert
 - bevor sie Schaden anrichten

aber nicht deren Ursache finden und diese beheben
- Testvorgehen und -umfang müssen an die möglichen Risiken und die Qualitätsziele angepasst sein