GTB
German Testing Board
Software.Testing.Excellence.

Kapitel 6

Basiswissen Softwaretest Certified Tester

Testwerkzeugunterstützung für das Testen

HS@GTB
2019
Version 3.1

Nach dieser Vorlesung sollten Sie ...

- den Begriff „Testwerkzeug“ erklären können,
- den Zweck der Werkzeugunterstützung für den Test erklären können,
- Typen von Testwerkzeugen benennen und ihre grundlegende Funktionalität beschreiben können,
- Werkzeuge zur Unterstützung der Entwickler beim Testen kennen,
- Nutzen und Risiken der Werkzeugunterstützung und Testautomatisierung kennen,
- die bei den Testausführungstechniken angewandten skriptbasierten Techniken wiedergeben können,
- einen Überblick darüber haben, was bei der Auswahl und Einführung eines Testwerkzeugs zu beachten ist,
- die Ziele einer Pilotphase im Rahmen der Werkzeugeinführung kennen.

Lernziele für den Abschnitt Werkzeugunterstützung für das Testen

(nach Certified Tester Foundation Level Syllabus, deutschsprachige Ausgabe, Version 2018)

- 6.1 Überlegungen zu Testwerkzeugen (K2)
  - FL-6.1.1 Testwerkzeuge gemäß ihrem Zweck und den Testaktivitäten, die sie unterstützen, klassifizieren können. (K2)
  - FL-6.1.2 Nutzen und Risiken der Testautomatisierung identifizieren können. (K1)
  - FL-6.1.3 Sich an besondere Gesichtspunkte von Testausführungs- und Testmanagementwerkzeugen erinnern können. (K1)
- 6.2 Effektive Nutzung von Werkzeugen (K1)
  - FL-6.2.1 Die Hauptprinzipien für die Auswahl eines Werkzeugs identifizieren können. (K1)
  - FL-6.2.2 Sich an Ziele für die Nutzung von Pilotprojekten zur Einführung von Werkzeugen erinnern können. (K1)
  - FL-6.2.3 Erfolgsfaktoren für die Evaluierung, Implementierung, Bereitstellung und kontinuierliche Unterstützung von Testwerkzeugen in einem Unternehmen identifizieren können. (K1)

Werkzeugunterstützung für das Testen

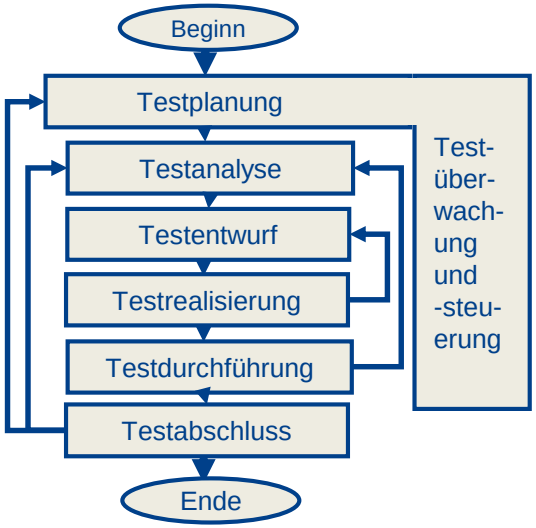
Werkzeugunterstützung für das Testen (1 von 2)

- Klassifizierung von Testwerkzeugen nach
  - unterstützter Aktivität/Phase des Testprozesses (hier im Fokus)
  - unterstützter Teststufe
  - unterstützter Testart
  - unterstützter Testrolle
  - unterstützter Technologie
  - unterstütztem Lizenzmodell (kommerziell/Shareware/Open Source)

  - Beeinflussung des Testobjekts
    - Ja: Intrusives Testwerkzeug (erzeugt sog. „Untersuchungseffekt“)
      Beispiel: Durch Code-Instrumentierung eingefügter Code kann das Performance-Verhalten oder den Speicherbedarf beeinflussen
    - Nein: Nicht-intrusives Testwerkzeug

Werkzeugunterstützung für das Testen (2 von 2)

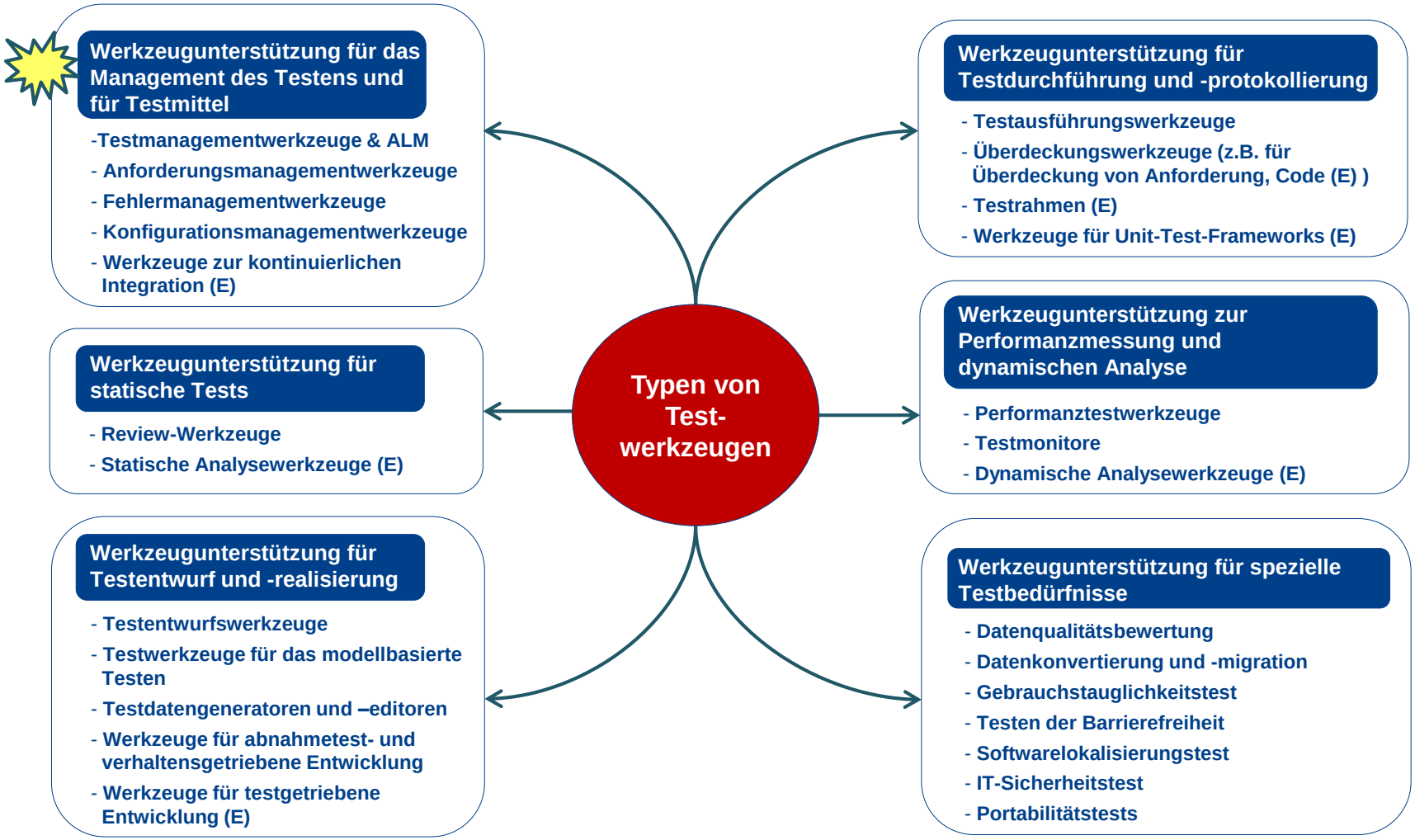
- Unterstützte Aktivität bzw. Phase des Testprozesses
  - Testplanung
  - Testüberwachung und -steuerung
  - Testanalyse
  - Testentwurf
  - Testrealisierung
  - Testdurchführung
  - Testabschluss



- Einzelne Testwerkzeuge unterstützen meist nur einen kleinen Ausschnitt des Testprozesses
- Testwerkzeug-Familien / Testwerkzeug-Suiten decken viele / alle Aktivitäten des Testprozesses ab, als Einheit zu betrachten
- Für ein Testprojekt wird selten die ganze Bandbreite an Testwerkzeugen eingesetzt

Werkzeugunterstützung für das Testen

Typen von Testwerkzeugen (E – auch von Entwicklern genutzte Werkzeuge)

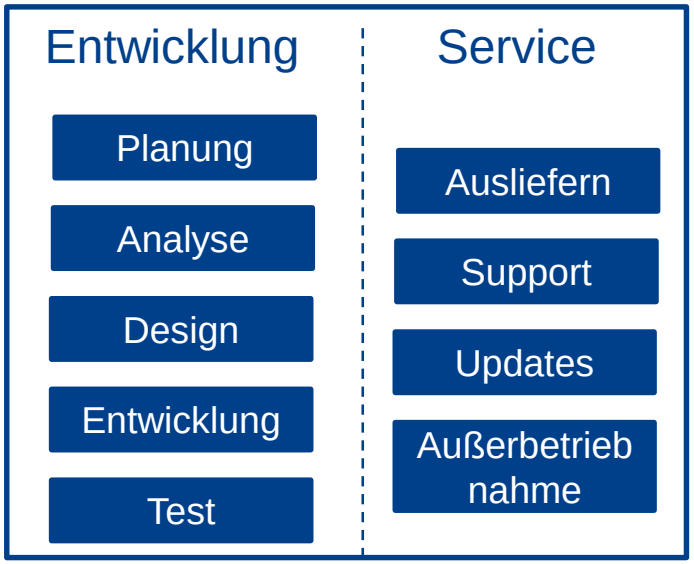


Testmanagement: Werkzeuge für Testplanung und Steuerung

- Anzahl der Testfälle oft drei- bis vierstellig
- Testplanungswerkzeuge für Erfassung, Katalogisierung, Verwaltung und Priorisierung von Testfällen notwendig
- Fortgeschrittene Funktionalität
  - Erfassung / Import von Systemanforderungen (Requirements)
  - Traceability (Rückverfolgbarkeit) von Tests, Testergebnissen und Vorfällen zu Anforderungen
  - Abdeckungsprüfung (z.B. mindestens ein Test pro Anforderung)
  - Testfallstatus (wie oft durchgeführt, mit welchem Resultat)
  - Aufzeichnung von Testergebnissen und Erstellung von Fortschrittsberichten
  - Zeit- und Ressourcenplanung
  - Testfortschrittsüberwachung durch quantitative Analyse (Metriken)
  - Eigenständige Versionskontrolle oder Schnittstelle zu einem externen Konfigurationsmanagementwerkzeug
  - Weitere Schnittstellen zu Testausführungswerkzeugen sowie zu Fehlermanagementwerkzeugen

ALM: Application Lifecycle Management

- ALM-Werkzeug unterstützt den gesamten Lebenszyklus einer Software
  - u.a. Planung, Analyse, Design, Entwicklung, Test, Ausliefern, Support, Updates, Außerbetriebnahme, …
- Trennung in Entwicklungs- und Service-Ansatz möglich
  - Vorteilhaft ist die gemeinsame Betrachtung beider Themen
- Trends
  - Spezialisierung auf relevante Teilthemen
  - Automatisierung sämtlicher Prozesse
  - Follow-the-Sun: 24/7-Erreichbarkeit



Anforderungsmanagementwerkzeuge

- Komplexität und hohe Anzahl von Anforderungen, daher Anforderungsmanagementwerkzeuge für
  - Erfassung, Katalogisierung, strukturierte Ablage, Verwaltung und Änderungsmanagement, sowie Priorisierung von Anforderungen
- Fortgeschrittene Funktionalität:
  - Prüfung auf Konsistenz sowie auf fehlende/undefinierte Anforderungen
  - Rückverfolgung von einzelnen Tests zu Anforderungen, Funktionen und/oder Features
  - Messung des Überdeckungsgrads von Anforderungen, Funktionen und/oder Features durch eine Menge von Tests

Fehlermanagementwerkzeuge

- Unmengen von Fehlermeldungen erfassen, verwalten und verteilen
  - Abweichungen aller Art: Fehlerzustände, Änderungsanforderungen, Fehlerwirkungen, Anomalien, …
- Ermöglichen die Verfolgung der Fehler über die Zeit
- Unterstützen statistische Analysen und liefern Berichte über Fehler
- Fortgeschrittene Funktionalität:
  - Parametrisierbare Fehlerstatusmodelle (Bearbeitungsstatus, Priorität etc.)
  - Parametrisierbare Testfallstatusmodelle (z.B. geplant, spezifiziert, durchgeführt)
  - Workflow-Funktionalität für den Fehlerbehebungsprozess (z.B. Zuordnung von Aufgaben zu bestimmten Personen und/oder Personengruppen für Fehlerbehebung oder Nachtest)
- Häufig gekoppelt mit Testmanagementwerkzeugen
  - Planung für Fehlernachtest, Fehlerstatistik und -analyse
  - Generierung von Testdokumentation

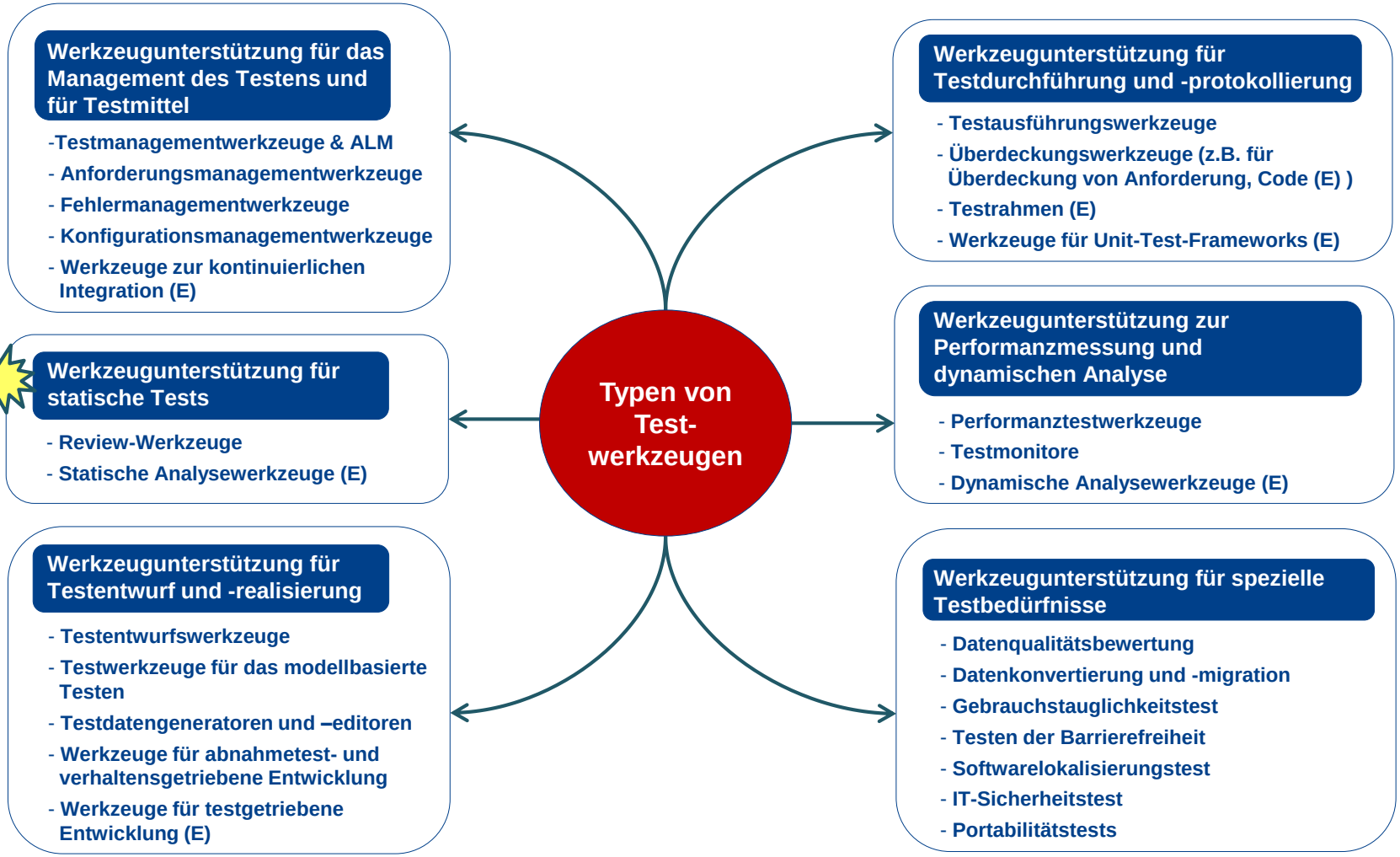
Konfigurationsmanagementwerkzeuge

- Keine Testwerkzeuge im engeren Sinn, können als Testunterstützungs-werkzeuge bezeichnet werden
- Identifikation, Verwaltung, Bereitstellung und Speicherung der Information über Versionen und Konfigurationen der Software und der benötigten Testmittel
- Überwachung und Dokumentation der Änderungen der Software und der Testmittel
- Erlauben die Rückverfolgbarkeit zwischen den Software-Produktkomponenten, den Varianten und den Testmitteln
- Insbesondere für die Verwaltung von mehreren Konfigurationen von Hardware- und Softwareumgebungen geeignet
  - z.B. für verschiedene Betriebssystemversionen, Bibliotheken und Compiler, Browser und Rechner

Werkzeuge zur kontinuierlichen Integration (E)

- Kontinuierliche Integration (continuous integration):
  - Manuelles Einchecken neuen Codes in ein Repository löst vollautomatisiert viele Testschritte aus:
  - Kompilieren und Linken der Software
  - Testdurchführung (statische Analysen und dynamische Tests)
  - Ermittlung von Metriken (z.B. Code-Überdeckung)
  - Bereitstellung einer neuen Version des Produkts mit Qualitätsstatus
- Ziele:
  - nach jeder Änderung schnellstmögliches Feedback an Entwicklung
  - Automatische und objektive Testdurchführung
  - Automatische und reproduzierbare Produkterstellung
- Beispiel-Werkzeug: Jenkins

Typen von Testwerkzeugen (E – auch von Entwicklern genutzte Werkzeuge)



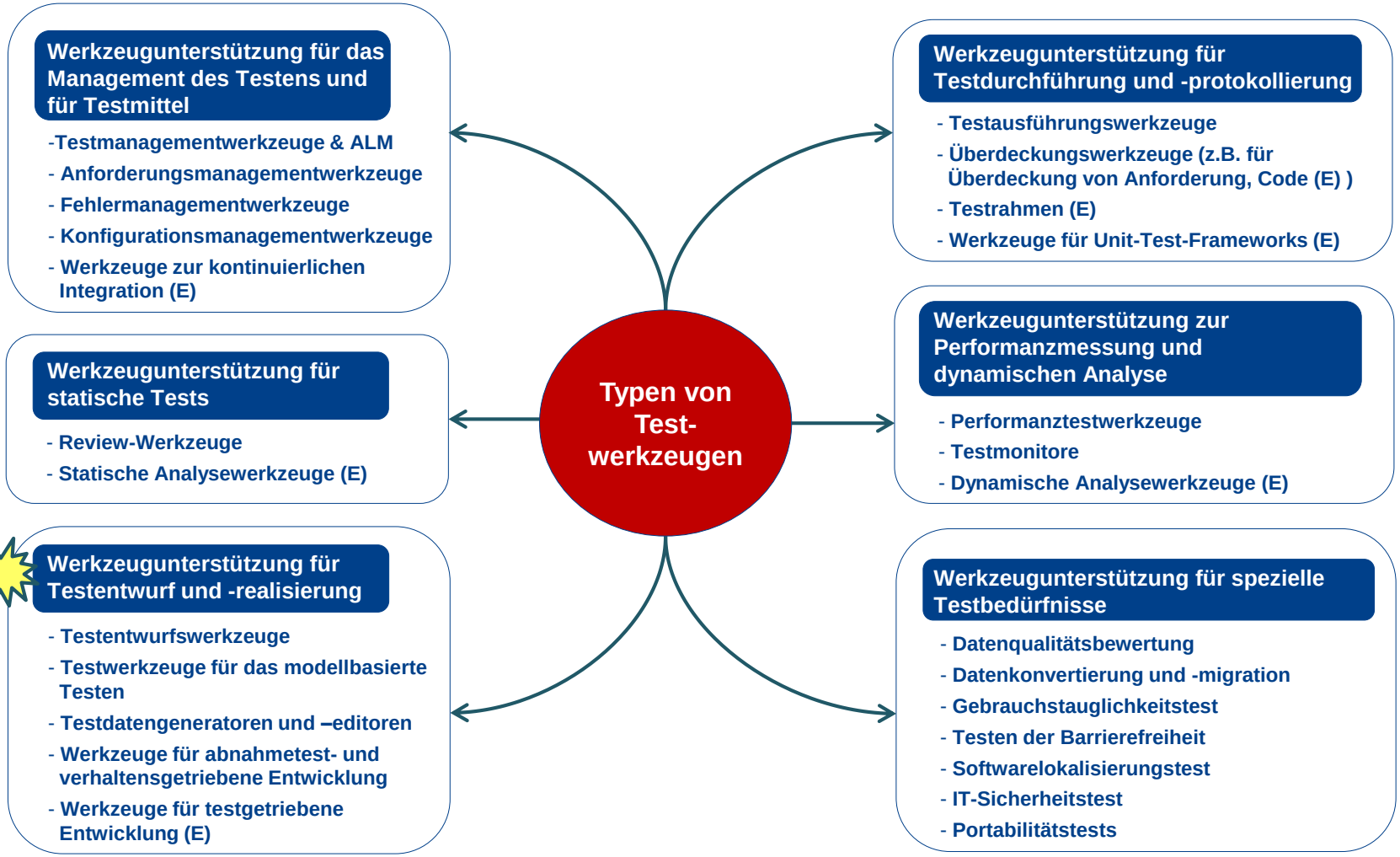
Review-Werkzeuge

- Verwaltung der Dokumente (data handling)
  - Verwaltung von Reviews und Checklisten (Erstellen, Bearbeiten, Löschen)
  - Verteilen von Reviewanmerkungen und von Reviewergebnissen
  - Verwaltung und Versionskontrolle der zu überprüfenden Dokumente
  - Workflow-Funktionalität für den Review- und Fehlerbehebungsprozess (z.B. Zuordnung von Aufgaben zu bestimmten Personen und/oder Personengruppen)
- Unterstützung der Gutachter (individual preparation)
  - Integration von weiteren Werkzeugen, z.B. zur Überprüfung der Einhaltung von Programmierrichtlinien
- Unterstützung der Reviewsitzung (meeting support)
  - Synchron vs. asynchron
  - Funktionalität zur verteilten Kommunikation (z.B. Nachrichtendienste, Chat, Videokonferenzen, Kalender, usw.)
  - Verteilte Reviews (Online-Reviews)
- Auswertung (data collection)
  - Sammlung von Reviewergebnissen
  - Auswertung von Reviewergebnissen durch Metriken (z.B. gefundene Abweichungen oder geleisteter Aufwand)

Statische Analysewerkzeuge (E)

- Analysieren verschiedener Eigenschaften des Programmcodes
  - Strukturelle Eigenschaften (z.B. Zyklomatische Zahl, Vererbungstiefe)
  - Datenflussanomalien (z.B. Zugriff auf nicht initialisierte Variablen)
  - Einhaltung von Programmierkonventionen (z.B. Einhaltung der maximalen Schachtelungstiefe)
  - Einhaltung von Konventionen zur sicheren Programmierung (secure code)
- Fortgeschrittene Funktionalität
  - Architekturprüfung und -visualisierung
  - Visualisierung von Metriken und Metrikenkorrelationen
  - Klonerkennung (duplizierter Code)
  - Zykluserkennung (zyklische Abhängigkeiten zwischen Elementen im Quellcode)

Typen von Testwerkzeugen (E – auch von Entwicklern genutzte Werkzeuge)



Testentwurfswerkzeuge

- Generieren Testfälle oder Testeingabedaten (Testdatengeneratoren) aus unterschiedlichen Modellen und Quellen, z.B. aus
  - Anforderungen
  - Graphischer Benutzungsschnittstelle (GUI)
  - Entwurfsmodellen (Zustands-, Daten- oder Objektmodell)
  - Code
- Generieren das Testorakel
  - Erwartetes Verhalten (Sollwerte/Sollreaktionen)
  - Erwarteter Nachzustand
  - aber: nicht immer möglich, oft müssen erwartete Reaktionen manuell ergänzt werden
- Keine Garantie für »gute« Testfälle
- Generierung strukturierter Vorlagen (Templates, Testrahmen)
- Testentwurfswerkzeuge decken meistens nur Teilaspekte der zu testenden Software ab

Testwerkzeuge für das modellbasierte Testen

- Unterstützen die Spezifikation sowie die Validierung und Verifikation von Modellen der zu erstellenden Software
- Ausgangspunkt für die Generierung von Testdaten und Testfällen aus dem Modell
- Ermöglichen frühzeitiges Aufdecken von Fehlern im Entwicklungsprozess
- Fortgeschrittene Funktionalität
  - Codegenerierung, Generierung von ausführbarem Testcode
  - Generierung von Testdaten und Testfällen
- Hauptnutzen von statischen Analyse- und Modellierungswerkzeugen:
  - Effektive Aufdeckung von Fehlerzuständen im Entwicklungsprozess
  - Später weniger Aufwand für Überarbeitung bzw. Nacharbeit

Testdatengeneratoren und -editoren (1 von 2)

- Arten von Testdatengeneratoren:
  - Datenbankbasiert
  - Codebasiert
  - Schnittstellenbasiert
  - Spezifikationsbasiert
- Datenbankbasierte Testdatengeneratoren
  - Analysieren Datenbankschemata zur Generierung von Testdaten
  - Analysieren Datenbankinhalte und filtern Testdaten heraus
  - Auch für Dateien (oder Datenströme) unterschiedlichster Formate verfügbar
  - Vorteil:
    Erzeugung von „künstlichen“ Testdaten, bei deren Verwendung es keinen Konflikt mit Datenschutzbestimmungen gibt (im Gegensatz zu realen Daten aus Datenbanken)

Testdatengeneratoren und -editoren (2 von 2)

- Codebasierte Testdatengeneratoren
  - Analysieren Code (Parametertypen, Kontroll-/Datenfluss, ...) des Testobjekts
  - Keine Generierung von erwarteten Werten (bzgl. Spezifikation)
  - Kein Erkennen von vergessenem Code
- Schnittstellenbasierte Testdatengeneratoren
  - Analysieren Testobjektschnittstelle (z.B. API oder GUI)
  - Analysieren der Parametertypen und Anwendung von Äquivalenzklassen- und Grenzwertanalyse
  - Keine Generierung von Sollwerten
  - Gut geeignet für Negativtests
- Spezifikationsbasierte (Anforderungsbasierte) Testdatengeneratoren
  - Ableitung von Testdaten aus der Spezifikation, die in einer formalen Notation vorliegen muss

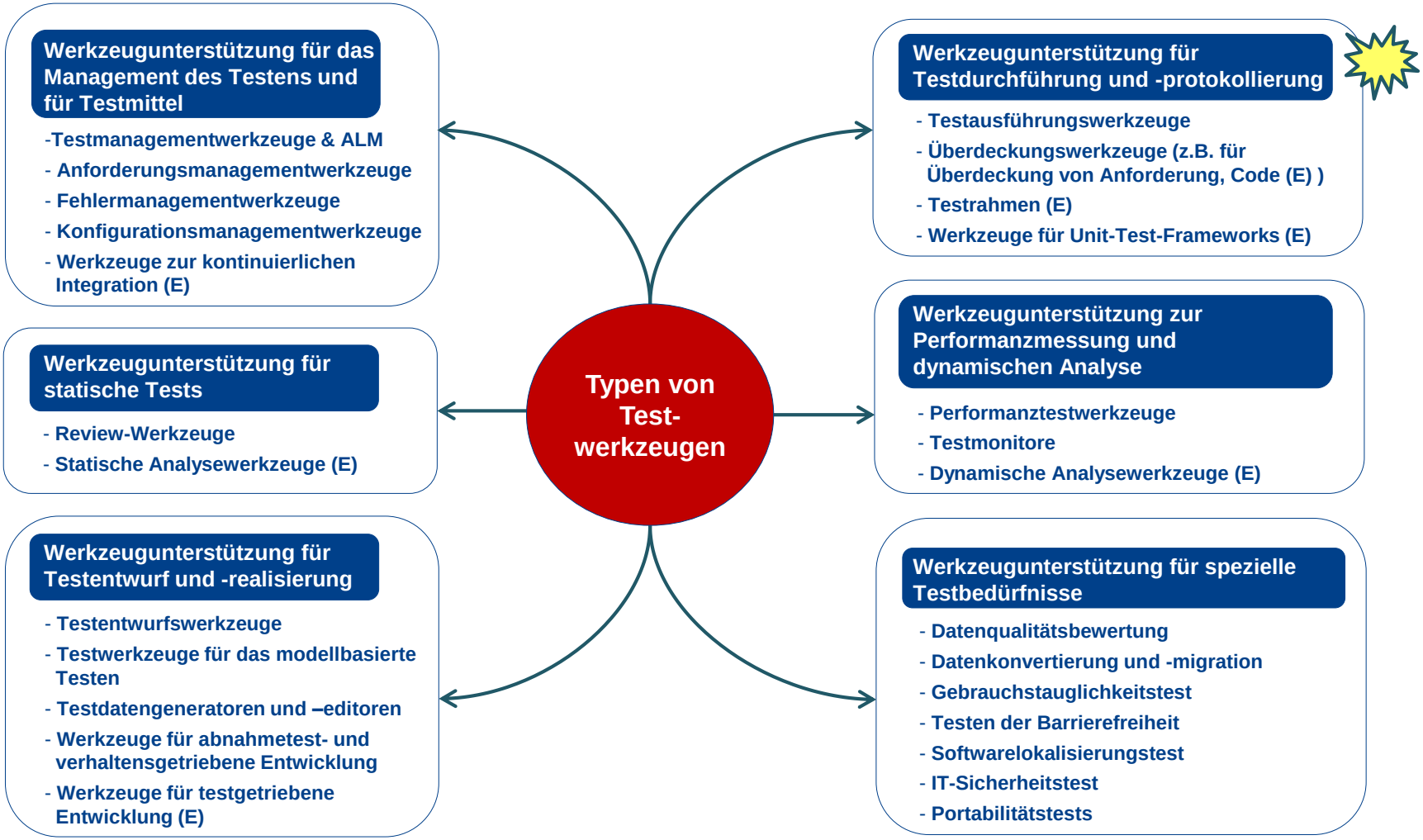
Werkzeuge für abnahmetest- und verhaltensgetriebene Entwicklung

- Abnahmetest durch den Kunden in Produktionsumgebung
  - Grundlage für Entscheidung zur Abnahme durch den Kunden
  - auch mit Kopie der Produktionsdaten möglich
- Häufige Varianten:
  - Alpha-Test: in-house, durch Tester durchgeführt
  - Beta-Tests: echte Umgebung, durch echte Kunden durchgeführt
- Abnahmetest- oder verhaltensgetriebene Entwicklung: Fokus auf bestimmte Formulierung der Anforderungen, um Test zu erleichtern
  - Wenn, dann …
  - Gherkin: GIVEN…, WHEN…, THEN…
- Tools:
  - Jbehave, Cucumber, Nbehave, …
  - oder Eigenentwicklung, z.B. auf Basis von xText

Werkzeuge für testgetriebene Entwicklung (E)

- „test first“-Ansatz
  - Test entwerfen, der eine Fehlerwirkung aufdeckt
  - Testobjekt so entwickeln, dass der Test keine Fehlerwirkung aufdeckt
  - Im Code des Testobjekts aufräumen (Refactoring)
- Werkzeuge
  - Build-Automatisierung: Jenkins, Hudson, CruiseControl
  - Build & Test: Maven (Ant, Gradle, …) & JUnit (NUnit, PHPUnit, …)
  - Mock-Objekte
  - Variante für Akzeptanz- und Systemtest: Fitnesse

Typen von Testwerkzeugen (E – auch von Entwicklern genutzte Werkzeuge)



Testausführungswerkzeuge (1 von 3)

- Automatische oder halbautomatische Ausführung von Testfällen:
  - Versorgung des Testobjekts mit Testdaten
  - Aufzeichnung der Reaktionen des Testobjekts
  - Protokollierung des Testlaufs
- Müssen Testschnittstelle des Testobjekts ansprechen können, in Abhängigkeit von der Teststufe (Komponenten-, Integrations-, Systemtest)
- Verschiedene Typen
  - Testausführungswerkzeuge
  - Testrahmen/Komponententestrahmen
  - Simulatoren
  - Vergleichswerkzeuge/Komparatoren
  - Werkzeuge zur Überdeckungsmessung
  - Sicherheitsprüfwerkzeuge

Testausführungswerkzeuge (2 von 3)

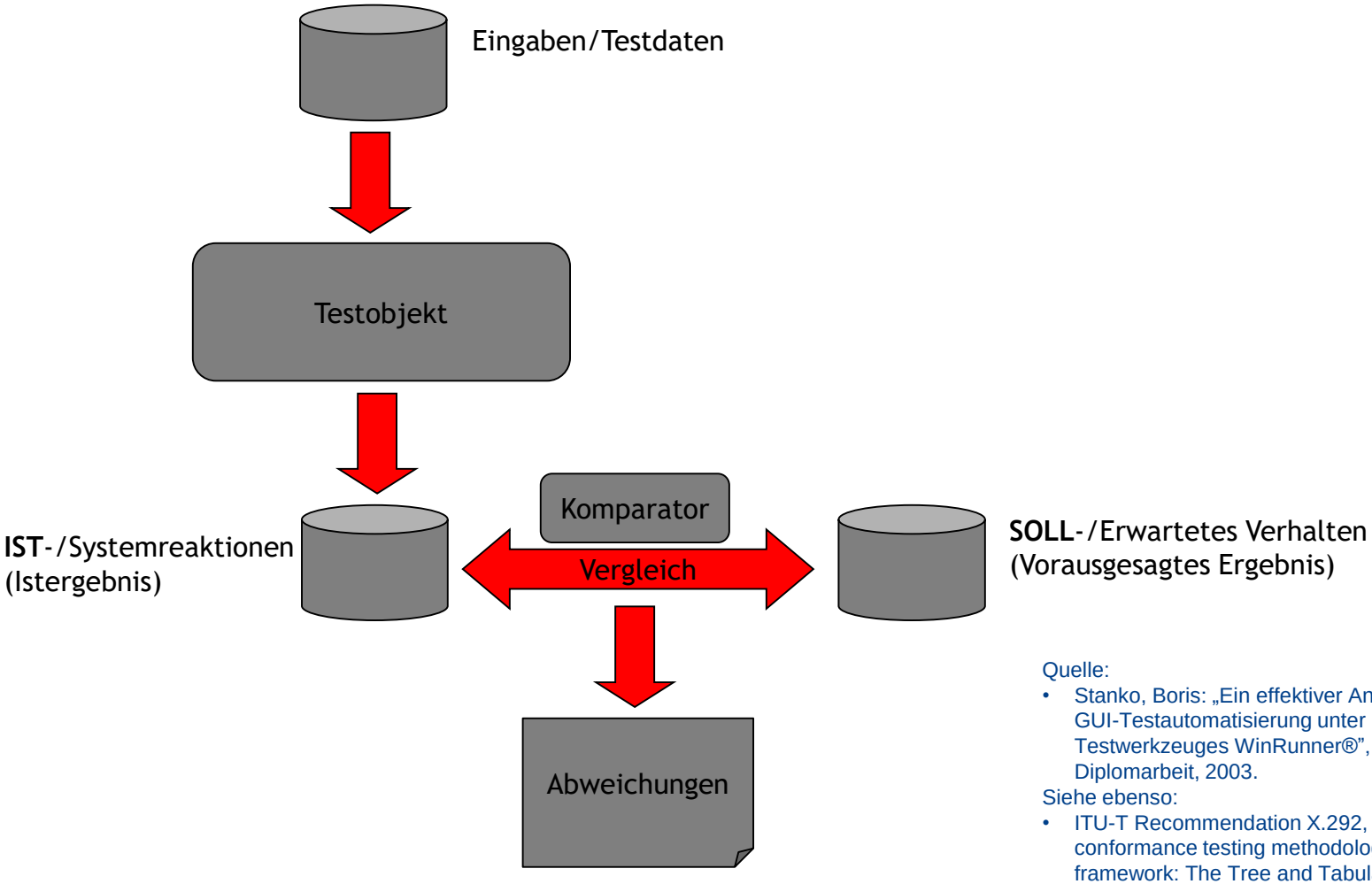
- Automatisieren funktionale Tests
- Testschnittstelle ist die äußere Schnittstelle des Testobjekts, beispielsweise die Bedienoberfläche des Testobjekts
- Unterschiedliche Ansätze zur Automatisierung der Testdurchführung
  - Capture & Replay
  - Skriptbasiertes Testen
  - Datengetriebenes Testen
  - Schlüsselwortgetriebenes Testen
  - Testrahmen/Komponententestrahmen

Testausführungswerkzeuge (3 von 3)

Vergleichswerkzeuge/Komparatoren

- dienen dem Vergleich zwischen erwartetem und beobachtetem Ergebnis
- verarbeiten marktgängige Datei- und Datenbankformate
- filtern relevante von irrelevanten Daten durch Filtermechanismen
- Testausführungswerkzeuge verfügen über Komparatoren für
  - GUI-Objekte
  - Bildschirminhalte
  - Konsoleninhalte
  - Komplexe Datentypen
  - …

Ausführung und Protokollierung von Tests



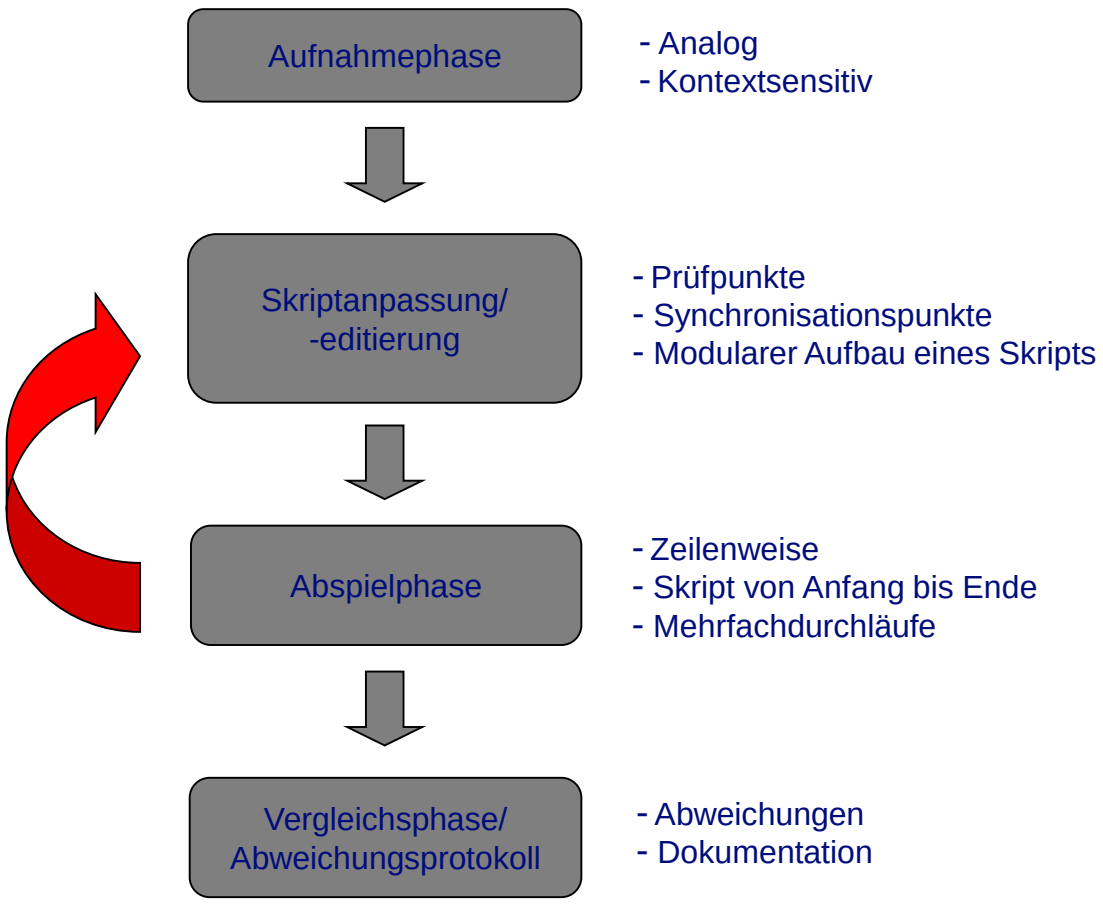
Quelle:
- Stanko, Boris: „Ein effektiver Ansatz zur GUI-Testautomatisierung unter Einsatz des Testwerkzeuges WinRunner®", Diplomarbeit, 2003.

Siehe ebenso:
- ITU-T Recommendation X.292, OSI conformance testing methodology and framework: The Tree and Tabular Combined Notation (TTCN), 1998.

Ansätze zur Automatisierung der Testdurchführung (1 von 3)

- Capture/Replay-Ansatz durch Mitschnittwerkzeuge
  - Synonym: Capture & Playback-Testwerkzeug
  - Aufzeichnung (Capture) von Benutzeraktionen wie Mausbewegungen, Mausklicks und Tastatureingaben und Speicherung dieser Aktionen in Skripten
  - Definieren und Setzen von Checkpunkten
  - Abspielen (Replay) der Skripte inklusive Prüfung der Checkpunkte
  - Testdaten können oft aus externen Quellen eingelesen werden
  - Tester müssen die Skriptsprache kennen, um die Testskripte zu bearbeiten
  - Mittels der den Skripten zugrunde liegenden Programmiersprache ist es möglich (und oft nötig), die Skripte anzupassen
  - Aufzeichnung auch im Rahmen von explorativen Tests nützlich, um Tests reproduzieren und/oder dokumentieren zu können, falls eine Fehlerwirkung auftritt

Capture/Replay-Werkzeuge



Quelle: Stanko, Boris: „Ein effektiver Ansatz zur GUI-Testautomatisierung unter Einsatz des Testwerkzeuges WinRunner®", Diplomarbeit, 2003.

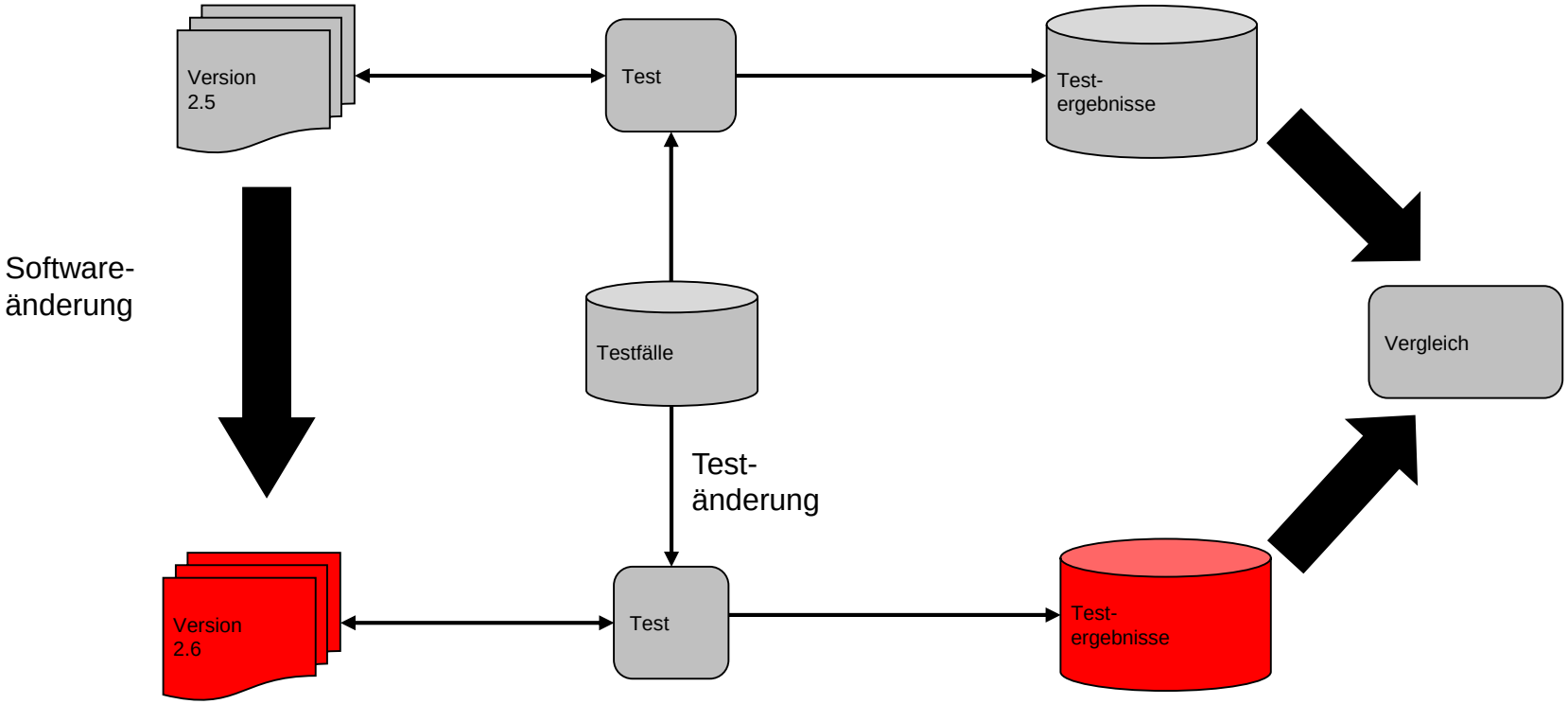
Ansätze zur Automatisierung der Testdurchführung (2 von 3)

- Datengetriebenes Testen
  - Testeingaben in Tabellenblatt abgelegt
  - Generisches Skript liest Daten „Zeile für Zeile“ ein
  - Testfälle mit unterschiedlichen Daten parametrisierbar
  - Trennung von Testdaten und Testvorgehensspezifikation
  - Tester können Testdaten ohne Kenntnis der Skriptsprache spezifizieren

Ansätze zur Automatisierung der Testdurchführung (3 von 3)

- Schlüsselwortgetriebenes Testen
  - Tabellenblatt enthält neben Testeingaben auch Schlüsselwörter (Aktionswörter, engl. „action words“)
  - Aktionswörter sind domänenspezifische Begriff und bilden die „Testsprache“
  - Keine Programmiererfahrung zur Spezifizierung „fachlicher“ Testfälle notwendig
  - Zuordnung zu entsprechenden Testskripten
  - Trennung zwischen „fachlicher“ und der „technischer“ Testfallspezifikation
  - Trennung von Testentwurf und Testrealisierung

Werkzeugunterstützung für Regressionstests



Angepasste Quelle:

- Stanko, Boris: „Ein effektiver Ansatz zur GUI- Testautomatisierung unter Einsatz des Testwerkzeuges WinRunner®", Diplomarbeit, 2003.

Überdeckungswerkzeuge (z.B. Anforderung, Code) (E)

- Ermitteln die verschiedenen Codeüberdeckungsmaße (z.B. Anweisungsüberdeckung, Entscheidungsüberdeckung, Funktionsaufrufe)
- Erstellen Statistiken oder graphische Darstellungen (z.B. Einfärben des Source-Codes) der erreichten Abdeckungen
- Werkzeuge
  - Cobertura
  - Code Cover
  - Coverage.py
  - EMMA (EclEMMA)
  - JaCoCo
  - …

Testrahmen (Komponententestrahmen) (E)

- alle Programme (u.a. Platzhalter, Testtreiber), die notwendig sind, um Testfälle auszuführen, auszuwerten und Testprotokolle aufzuzeichnen
  - wenn Komponenten noch nicht zur Verfügung stehen
  - wenn kontrollierte Umgebung für die Fehlerlokalisierung benötigt wird
- Platzhalter (Stubs)
  - benötigt, um nicht implementierte Komponenten zu simulieren
  - Einsatzgebiet: Komponenten- und Integrationstest
- Testtreiber
  - sprechen Testobjekte über deren Programmierschnittstelle (API) an
  - Einsatzgebiet eher Komponenten- und Integrationstest als Systemtest
  - Generische Testtreiber oder Testrahmengeneratoren:
    - Spezialisiert auf Programmiersprachen oder Entwicklungsumgebungen
    - Erleichtern die Programmierung einer Testumgebung erheblich
    - Automatische Generierung von Testrahmen, evtl. Generierung von Stubs
    - Funktionalität zur Abfrage von Reaktionen des Testobjekts und zur Protokollierung des Testablaufs

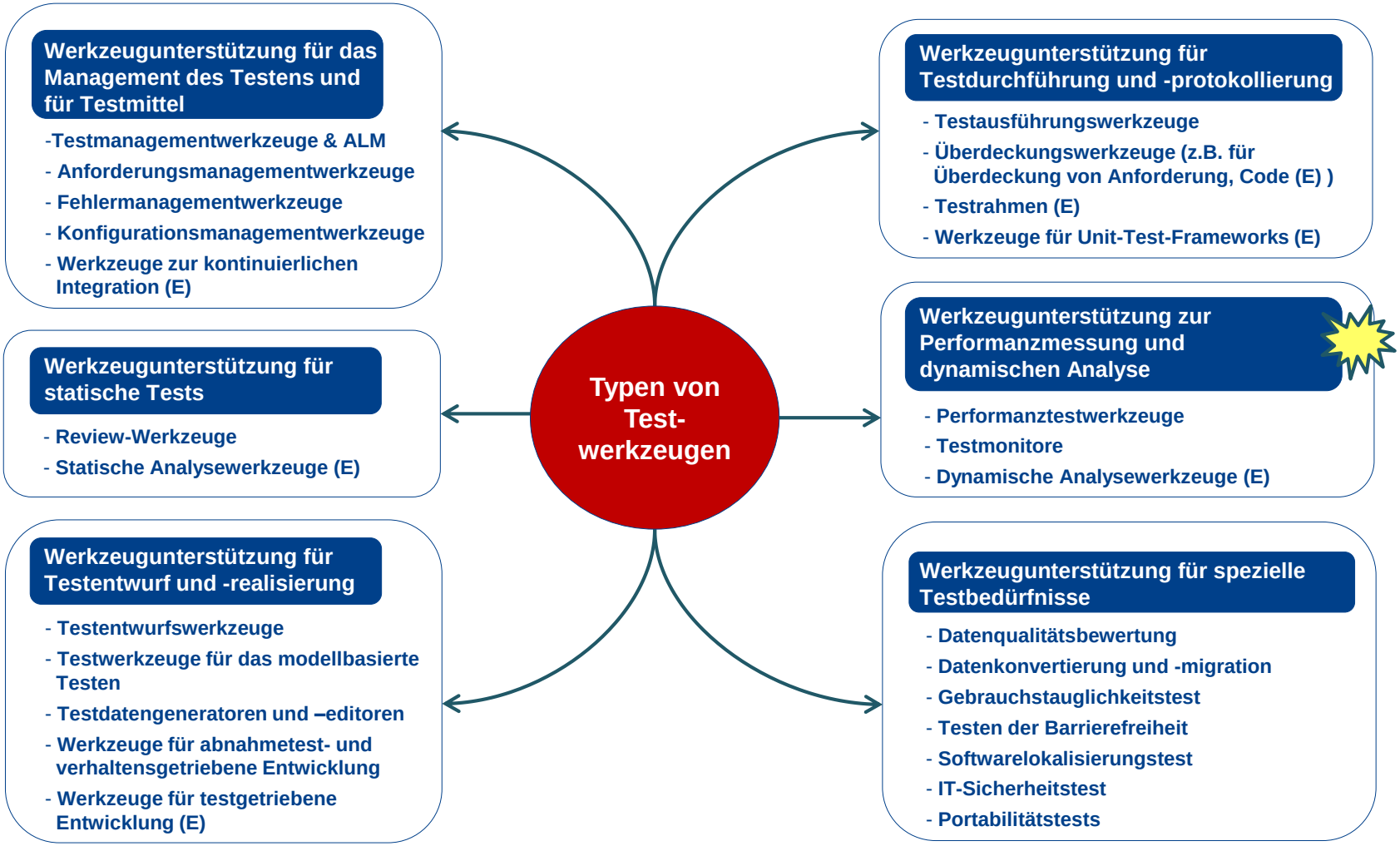
Werkzeuge für Unit-Test-Frameworks (E)

- meint Software-Frameworks zur Durchführung von Komponententests
- Nach kleineren Code-Änderungen (z.B. während des Refactoring)
  - Komponententests automatisch durchführen
  - Erhalt der ursprünglichen Funktionalität sicherstellen
- Unit-Test-Frameworks für die meisten Sprachen vorhanden
  - JUnit
  - CUnit
  - Qt
  - NUnit
  - …

Simulatoren (Scheinobjekte, Mocks)

- Möglichst umfassende und realitätsnahe Nachbildung der Produktiv-umgebung, wenn diese für den Test (noch) nicht zur Verfügung steht
- Beispiele:
  - Simulation von Hardware, die noch nicht gebaut ist (neue Handygeneration, neuer Prozessor etc.)
  - Simulation von externen Programmen (DBMS, Application Server etc.)
  - Entwicklung für andere Plattformen (z.B. für Smartphone, Tablet)

Typen von Testwerkzeugen (E – auch von Entwicklern genutzte Werkzeuge)



Performanz-/Last-/Stresstestwerkzeuge (1 von 2)

- Testen nicht-funktionale Anforderungen
- Performanztestwerkzeuge messen Antwortverhalten eines Systems unter verschiedenen simulierten Nutzungsbedingungen hinsichtlich
  - der Anzahl konkurrierender Nutzer
  - Hochlauf/Anlaufverhalten (ramp-up)
  - Häufigkeit/relativer Anteil von Transaktionen

- Werkzeuge für Lasttest generieren synthetische Last
- »Last« kann zum Beispiel sein:
  - Datenbankanfragen
  - Benutzertransaktionen
  - Netzwerkverkehr

Performanz-/Last-/Stresstestwerkzeuge (2 von 2)

- Stresstestwerkzeuge prüfen das Systemverhalten bei Überlastung (z.B. durch Betrieb mit zu hohem Datenvolumen oder auch gezielte Fehlbedienung)
- Durch Last-/Stress- und Performanztests aufgedeckte Mängel können durch Ausbau der Hardware und/oder Optimierung performanzkritischer Softwarekomponenten beseitigt werden

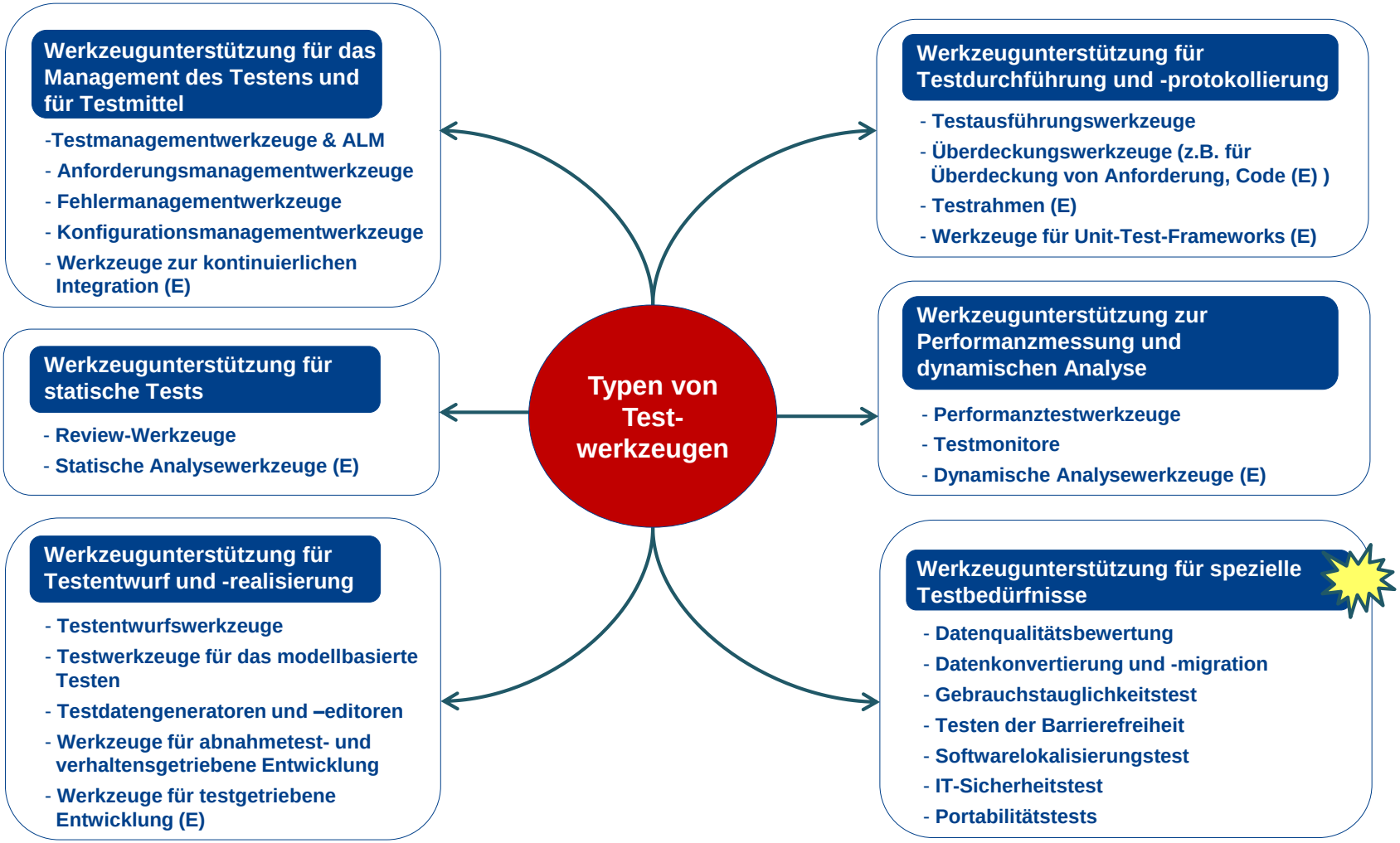
Testmonitore

- Testbegleitende Aufzeichnung, Analyse und Überprüfung von Daten und Messwerten, z.B. Netzwerkverkehr, Datenbankbelastung
- Warnen, falls Probleme in der Verwendung von Diensten auftreten
- Versionierung der verwendeten Software und Testmittel ermöglicht Rückverfolgbarkeit
- Keine Testwerkzeuge im engeren Sinn

Dynamische Analysewerkzeuge (E)

- Stellen zusätzliche Informationen zur Verfügung, die erst zur Laufzeit eines Programms erfasst werden können
  - Speicherbelegung
  - Zeigerzuordnung
  - Zeigerarithmetik
  - Memory Leaks
- Anwendung: Komponenten- und Integrationstest sowie für Tests der Middleware

Typen von Testwerkzeugen (E – auch von Entwicklern genutzte Werkzeuge)



Werkzeuge zur Datenqualitätsbewertung

- Datenqualität ist wesentlich für Erfolg der Software
- Allgemein:
  - Daten(qualität) -> Informationen -> Prognosen -> Entscheidungen
- Speziell:
  - Daten steuern Software
  - Qualität von Daten beeinflusst Qualität von Software
- Prüfung der Datenqualität
  - Syntaktisch: rudimentäre Eingangsprüfung der Daten
  - Semantisch: drücken die Daten die gewünschten Sachverhalte aus?
  - Trade-off:
    - Je spezifischer die Datenprüfung, desto teurer
    - Je abstrakter die Datenprüfung, desto weniger Fehler werden gefunden

Werkzeuge für Datenkonvertierung und -migration

- Daten müssen aktuell gehalten werden.
  - Umwandlung in neue Formate (Konvertierung)
  - Umzug in neues Datenverwaltungssystem / Plattform
    - In vielen kleinen Schritten (Migration)
    - In einem großen Schritt (Portierung)
- Datenmigration (ETL-Prozess)
  - Extraktion (und Auswahl der zu extrahierenden Daten)
  - Transformation
  - Laden
- Verifikation & Test
  - Definition von Testfällen für ETL-Prozess
  - Statistiken über transformierte Echtdaten (Anzahl der Elemente, …)

Werkzeuge für Gebrauchstauglichkeitstests

- Gebrauchstauglichkeit (Usability): Ausmaß, in dem ein Produkt in einem bestimmten Kontext effizient zur Erreichung von Zielen einsetzbar ist
- Gebrauchstauglichkeitstest / Usability-Test
  - Empirische Evaluation mit echten Benutzern
  - Benutzer sollen Produkt zu einem Zweck benutzen
    - Sie werden dabei beobachtet
    - Ihre Schwierigkeiten werden festgestellt
    - Sie sollen dabei laut denken
- Arten der Werkzeugunterstützung
  - Video- & Bildschirmaufzeichnung
  - Tracking-Software
  - Verfolgen der Augenbewegungen

Werkzeuge für das Testen der Barrierefreiheit

- Barrierefreiheit ist eine Spezialisierung der Gebrauchstauglichkeit
- Fokussiert auf Nutzer mit fehlenden / eingeschränkten Fähigkeiten
  - und entsprechend eingeschränkten Interaktionsmöglichkeiten
- Mängel in Barrierefreiheit können auf Gebrauchstauglichkeit durchschlagen
- Gesetzgeber: Behindertengleichstellungsgesetz (BGG)
- Testmöglichkeiten
  - Manuell
    - Langsam, aber alternativlos
  - Automatisch
    - Schnell und günstig, aber sehr unvollständig
    - Z.B. Textalternativen zu Bildern vorhanden?
  - Unterstützend
    - Anzeige von Alternativtexten, Einbinden des W3C-Validator, …

Werkzeuge für Softwarelokalisierungstests

- Lokalisierung = Ortsbestimmung (L10n oder i18n)
- Anpassen von Inhalten an kulturelle oder sprachliche Gegebenheiten im geographischen oder ethnischen Nutzungsgebiet
  - Sprache, Maßeinheiten (Datum, Zeit, Temperatur, Länge, …)
  - Musik, Videos, Zeichensatz, Bilder, Hilfetexte, …
- Softwarelokalisierung von Anfang an einplanen
- Lokalisierung…
  - Selbst umsetzen
  - Durch Dienstleister einkaufen (Angebote sehr unterschiedlich)
- Tools
  - Übersetzung
  - Verbesserung der Übersichtlichkeit
  - …

Werkzeuge für IT-Sicherheitstests

- IT-Sicherheit (Security) bzgl. Angriffen von außen
- IT-Sicherheitstest ermittelt Schwäche gegenüber solchen Angriffen
  - bilden bekannte Angriffe auf ähnliche Systeme ab
- Ziele
  - Technische Schwachstellen
    - Angriffe von außen: z.B. Speicherüberlauf erzeugen & Schadcode einschleusen
    - Fehlbedienungen von innen: z.B. Zugriffsrechte zu freizügig gewährt
  - Social Engineering-Schwachstellen
    - Ausspionieren von Mitarbeitern und deren Beziehung: z.B. vermeintliche Mail vom Chef über Millionenüberweisung
- Tools
  - Portscanner, Sniffer, Passwort-Cracker, …

Werkzeuge für Portabilitätstests

- Portabilität = Übertragung (lat.)
  - Meint die Plattformunabhängigkeit/Übertragbarkeit eines Programms
  - (Laufzeitumgebung, Betriebssystem, Hardware, …)
- Verschiedene Stufen möglich…
  - Webanwendungen (nur vom Browser abhängig)
  - Apps (Hybrid, Multi-Channel)
  - Bytecode (Java)
  - Laufzeit-Interpretation (Perl, Python, …)
  - Quellcode-Portabilität
- Portabilitätstest: Test der Portabilität einer Software
  - Software auf anderer Plattform installierbar?
  - Fehler während der Installation korrekt behandelt?
  - Auch Deinstallation oder downgrade funktioniert?
  - …?

Werkzeugunterstützung für spezifische Anwendungsbereiche (Beispiele)

- Performanzwerkzeuge speziell für web-basierte Applikationen
- Testwerkzeuge für Protokolltests
- Statische Analysewerkzeuge für bestimmte Entwicklungsplattformen
- Dynamische Analysewerkzeuge für die Prüfung von spezifischen Sicherheitsaspekten
- Werkzeug-Suiten für eingebettete oder sicherheitskritische Systeme
- Werkzeuge zur Unterstützung von GUI-Tests

Weitere Werkzeuge (keine speziellen Testwerkzeuge)

- Werkzeuge, die in unterschiedlichen Kontexten, aber auch für das Testen verwendet werden können
  - z.B. Tabellenkalkulation, SQL (Datenbanken), Debugger

Debugger

- Funktionalität:
  - Zeilenweise Abarbeitung des Programms
  - Anhalten des Programms in jeder Zeile möglich
  - Auslesen und Setzen von Variablen
- Verwendung im Rahmen des Testens:
  - Zum Erzwingen von »exotischen« bzw. schwer herstellbaren Testsituationen (oft bei der Ausnahmebehandlung)
  - Können als Testschnittstelle für Tests auf Komponentenebene dienen
- Eher Entwickler- als Testerwerkzeug

Werkzeugunterstützung für das Testen

Nutzen von Testwerkzeugen

- Einsparung von Ressourcen durch effiziente Bearbeitung von Aufgaben
- Standardisierung der Testdokumentation
- Transparenz des Testprozesses
- Durch Werkzeugeinsatz objektive Messungen
  (z.B. Überdeckungsmessungen, Messung des Systemverhaltens)
- Einfache Auswertung (Statistiken und graphische Darstellungen) über durchgeführte Tests (z.B. über Testfortschritt, Fehlerrate und Performanz)
- Höherer Automatisierungsgrad
  - Konsistenz und Wiederholbarkeit von Tests
  - Nachweisbarkeit von Tests
  - Testtiefe
  - Geschwindigkeit: Ausführung einer großen Menge von Tests möglich
  - Entlastung des Testteams von sich wiederholenden Tätigkeiten
  - Ausführbarkeit von Tests, die manuell nicht bzw. nur schwer ausgeführt werden können
  - Höhere Zuverlässigkeit der Tests beispielsweise durch automatisierten Vergleich großer Datenmengen

Risiken von Testwerkzeugen (1 von 2)

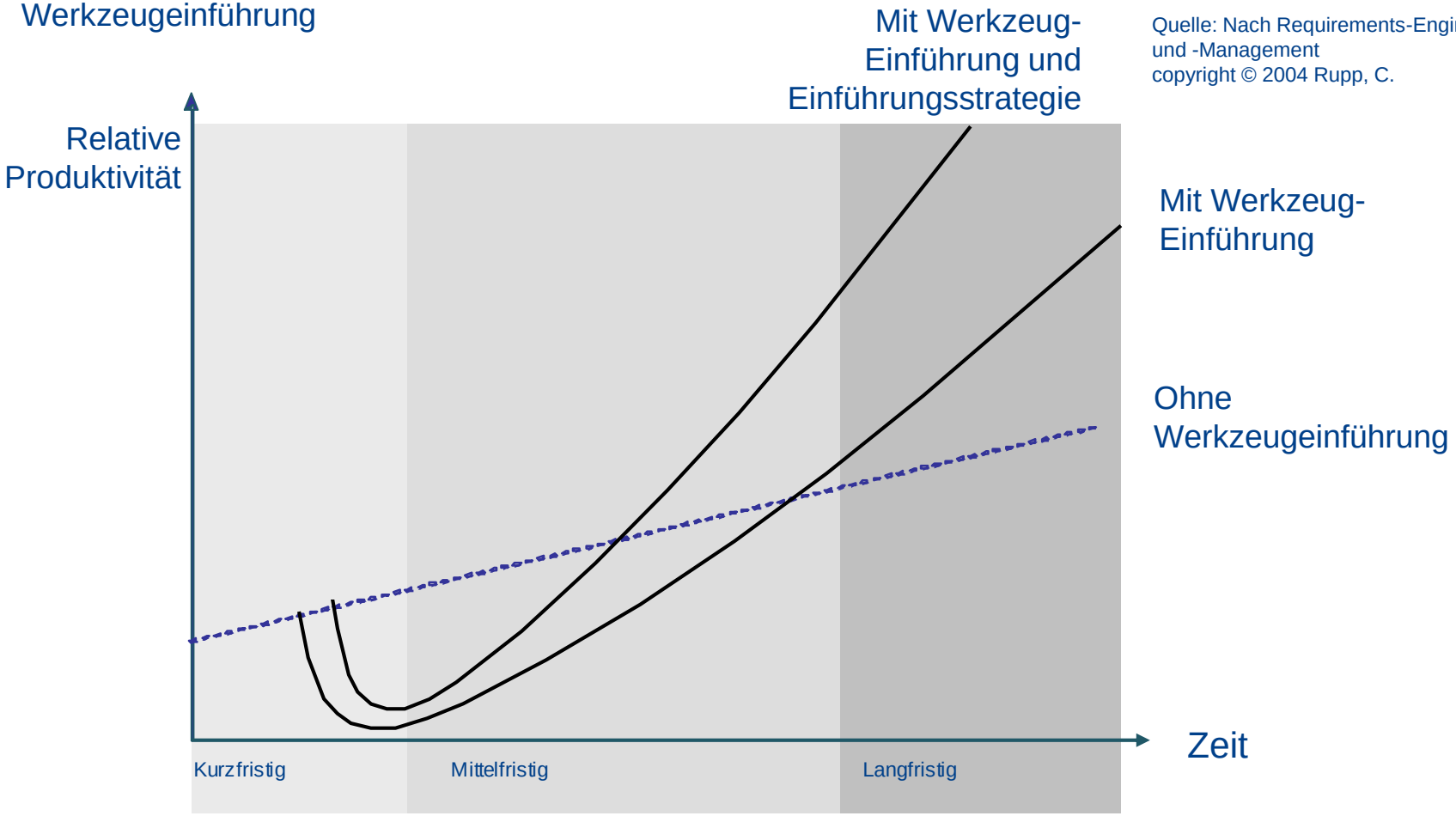
- Unrealistische Erwartungen an Automatisierung und Werkzeug
- Unterschätzung der Zeit, der Kosten und des Aufwands
  - von initialer Einführung eines Werkzeugs (einschließlich Schulung und Beratung)
  - bis zur Effizienzsteigerung (Lernkurveneffekt: Erst nach einer Einarbeitungszeit steigt mittelfristig die Produktivität an und übersteigt die Produktivität, die ohne Werkzeugeinführung erreicht worden wäre)
  - um bisherige Prozesse anzupassen
  - für die Wartung der durch das Werkzeug erzeugten Artefakte
- Blindes Vertrauen in das Werkzeug (Vernachlässigung oder sogar Ersatz für Testentwurf oder Ersatz für eigentlich besser geeignetes manuelles Testen)

Risiken von Testwerkzeugen (2 von 2)

- Vernachlässigung der Versionskontrolle der im Testwerkzeug verwalteten Entitäten (z.B. Anforderungen, Testfälle, Testergebnisse, etc.)
- Vernachlässigung der Komplexität der Schnittstellen zu anderen Werkzeugen (z.B. Schnittstellen zu Anforderungsmanagementwerkzeugen anderer Hersteller)
- Risiko, dass das Werkzeug nicht mehr weiter entwickelt wird
- Risiko, dass das Werkzeug den Hersteller wechselt
- Risiko eines unzureichenden Supports seitens Hersteller (z.B. lange Antwortzeiten bei Anfragen, zusätzliche Kosten für „Premium“-Support)
- Einführung von Testwerkzeugen ohne funktionierende Prozesse wird scheitern: „Automated chaos is just faster chaos!“
- Einführung von Testwerkzeugen häufig kurzfristig mit Produktivitätseinbußen verbunden, daher keine Lösung für kurzfristige Personalengpässe

Lernkurveneffekt

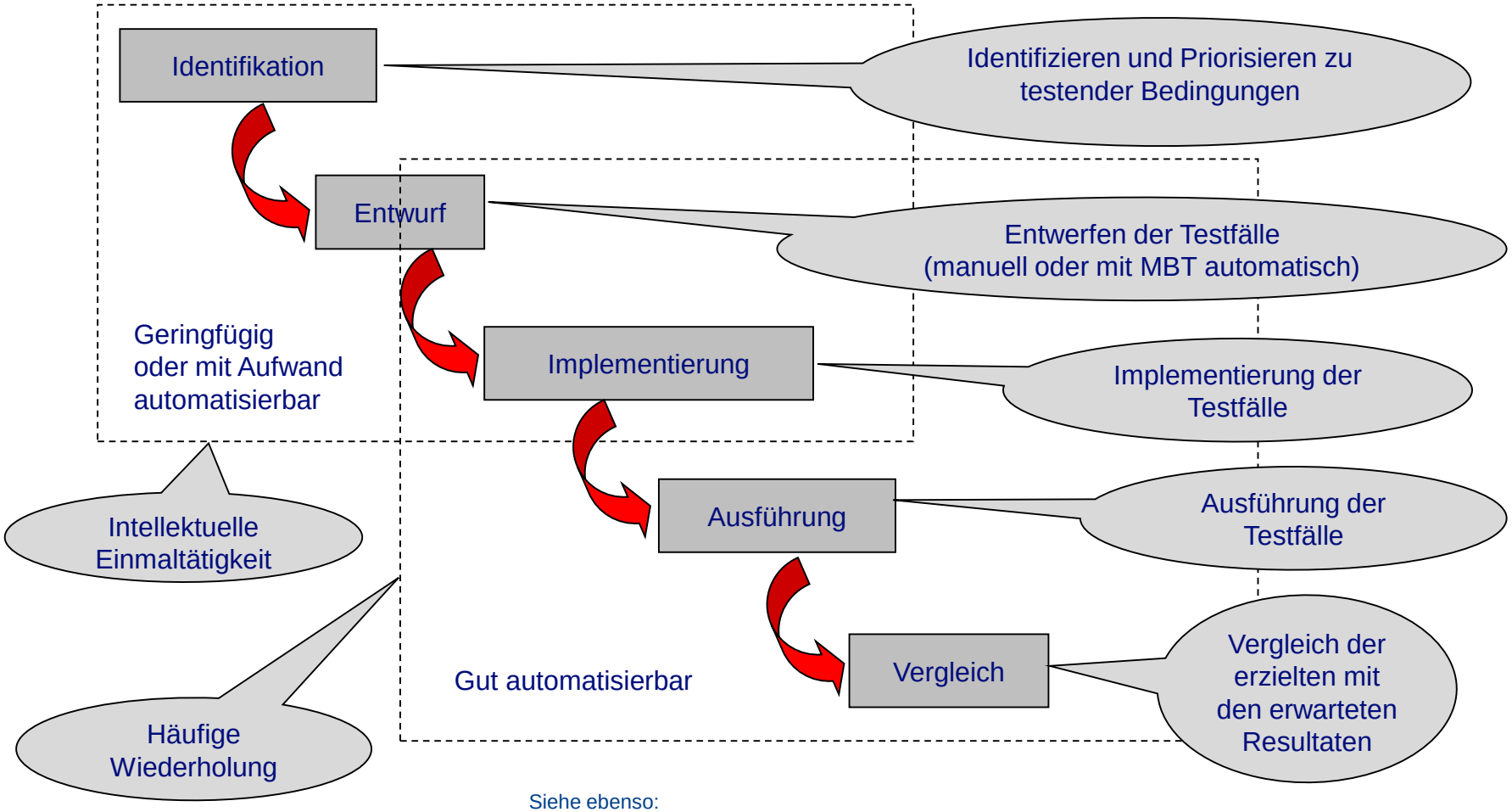
Relative Produktivität nach Werkzeugeinführung



Quelle: Nach Requirements-Engineering und -Management
copyright © 2004 Rupp, C.

Werkzeug-unterstützung für das Testen

Automatisierbare Testaktivitäten



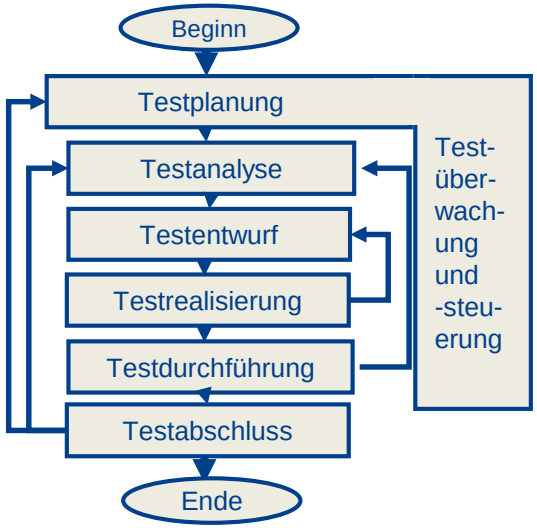
Siehe ebenso:

- ISTQB Test Automation Engineer Syllabus, 2016
- ISTQB Model-Based Testing Syllabus, 2015

Einführungsreihenfolge von Testwerkzeugen

Bei der Einführung der verschiedenen Arten von Testwerkzeugen ist folgende Einführungsreihenfolge empfehlenswert :

1. Fehlermanagement
2. Konfigurationsmanagement
3. Testplanung
4. Testdurchführung
5. Testdatengenerierung
6. Testanalyse und Testentwurf



Also: Im Vergleich zum Testprozess andere Reihenfolge der Automatisierung; intellektuell anspruchsvolle Aktivitäten immer zuletzt!

Kosten der Werkzeugeinführung

Bei einer Werkzeugeinführung entstehen (meist) Kosten für

- Auswahl
- Anschaffung
- Installation und Inbetriebnahme
- Hardware
- Mitarbeiterschulung
- Wartung

Wirtschaftlichkeit der Testautomatisierung

Beispiel Testdurchführung: Wann amortisiert sich die automatisierte Durchführung gegenüber der manuellen?

$$Ersparnis = n * (T_M - T_A) - A$$

n = Anzahl der Testläufe

T_M = Aufwand für eine manuelle Durchführung

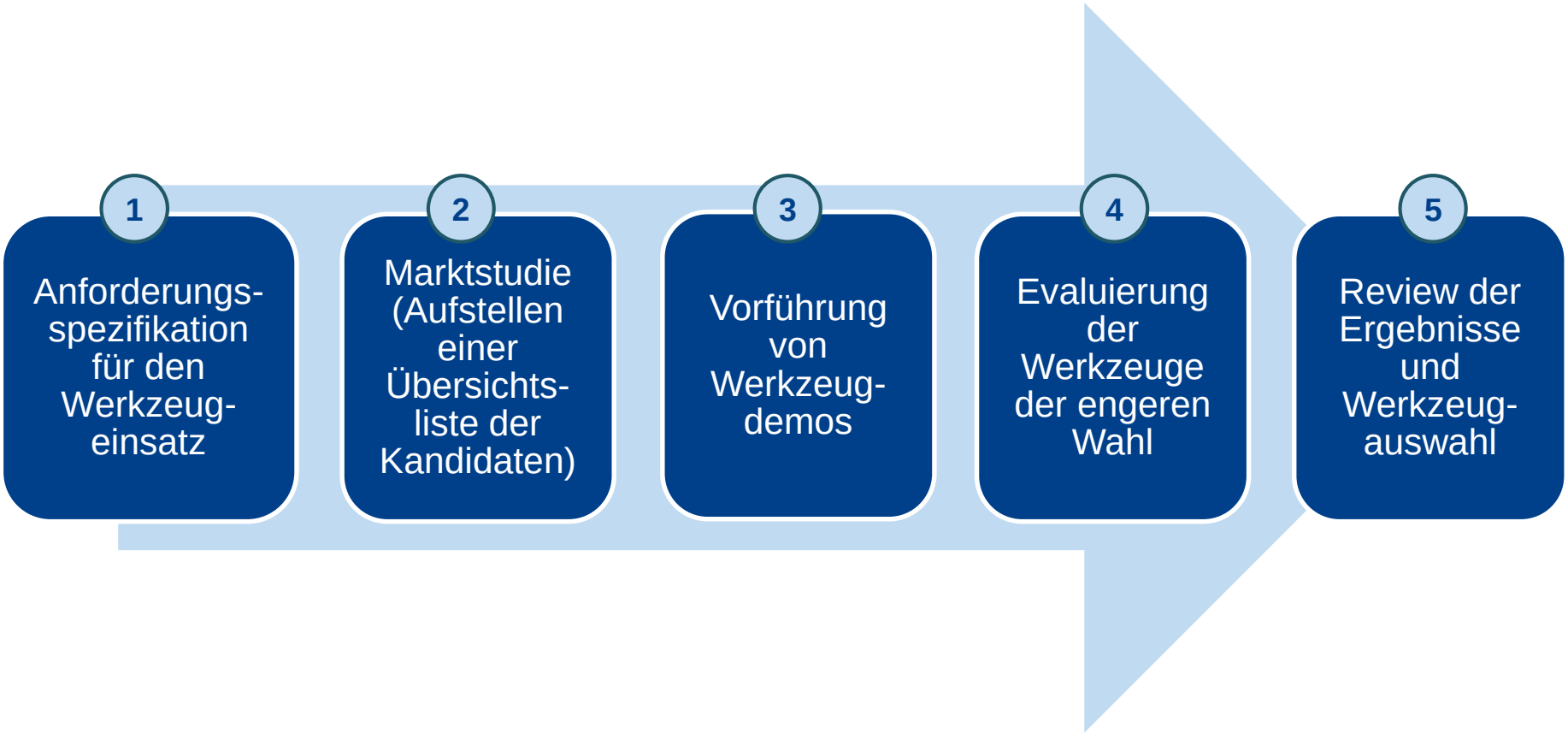
T_A = Aufwand für eine automatisierte Durchführung

A = Aufwand für die Automatisierung (meistens sehr groß!)

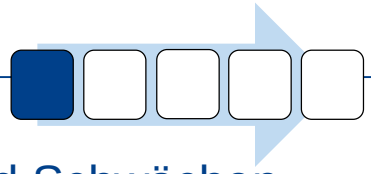
Aber: Das ist in vielen Fällen zu einfach gedacht:

- Automatisierte Tests meist häufiger ausgeführt als manuelle Tests
- Testautomatisierung kann dazu führen, dass *mehr* Testfälle spezifiziert und getestet werden (was nicht immer besser ist)
- Manche Tests lassen sich nicht manuell durchführen (z.B. Performanztest)
- Neben den quantitativen Kosten muss auch der Qualitätssprung berücksichtigt werden

Schritte der Werkzeugauswahl

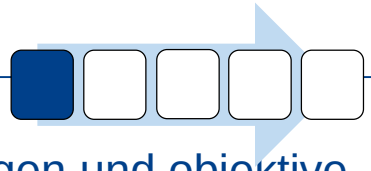


Grundsätzliche Überlegungen vor der Auswahl eines Werkzeugs



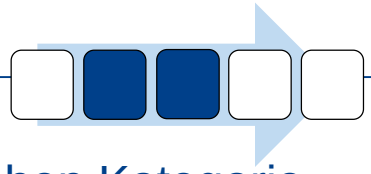
- Bewertung des Unternehmens, Analyse der Stärken und Schwächen
- Identifizierung von Möglichkeiten für die Verbesserung des Testprozesses, unterstützt durch Werkzeuge
- Verständnis der Technologien, die von den Testobjekten genutzt werden, um ein Werkzeug auszuwählen, das mit dieser Technologie kompatibel ist
- Werkzeuge für Build und kontinuierliche Integration, die im Unternehmen bereits im Einsatz sind, um Werkzeugkompatibilität und Integration zu gewährleisten
- Überlegung zu Vor- und Nachteilen verschiedener Lizenzmodelle (z.B. Standardsoftware oder Open Source)
- Schätzung des Kosten-Nutzen-Verhältnisses auf Basis eines konkreten Business Case (falls benötigt)

Anforderungen an konkrete Testwerkzeuge



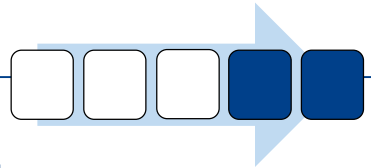
- Bewertung des Werkzeugs gegen spezifizierte Anforderungen und objektive Kriterien (für Nutzen und Anwendung)
  - Unterstützte Aspekte der Testautomatisierung
  - Unterstützte Methoden/Techniken
  - Plattform, auf der das Werkzeug eingesetzt werden soll
  - Güte des Zusammenspiels mit den potentiellen Testobjekten
  - Möglichkeit zur Integration in die vorhandene Entwicklungsumgebung
  - Integrationsmöglichkeiten mit Werkzeugen desselben Herstellers
  - Möglichkeit zur Integration mit anderen eingesetzten Testwerkzeugen
- Lizenzbedingungen, Preis, Wartungskosten
- Verfügbarkeit des Werkzeugs für eine kostenfreie Testperiode (ob und wie lange)
- Bewertung des Anbieters (einschließlich Trainings-, Unterstützungs- und kommerzieller Aspekte wie Marktstellung und Verlässlichkeit) oder der Unterstützung für nichtkommerzielle Werkzeuge (z.B. Open Source)
- Notwendigkeit und Umfang von Coaching und Anleitung zur Nutzung des Werkzeugs für spätere Anwender

Marktstudie und Werkzeugdemos



- Erstellung einer Liste verfügbarer Werkzeuge der fraglichen Kategorie
- Einschränken auf eine »engere Auswahl« anhand des Kriterienkatalogs und angeforderten Informationen über die Werkzeuge bzw. Internetrecherche
- Werkzeugdemonstration von den Herstellern der Produkte in der engeren Auswahl
- Einblick gewinnen in bzw. Hinterfragen der Servicephilosophie der jeweiligen Anbieter

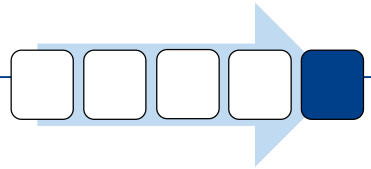
Evaluierung und Werkzeugauswahl



Nach erfolgten Werkzeugdemos sind in der abschließenden Evaluierungsrunde folgende Punkte zu verifizieren:

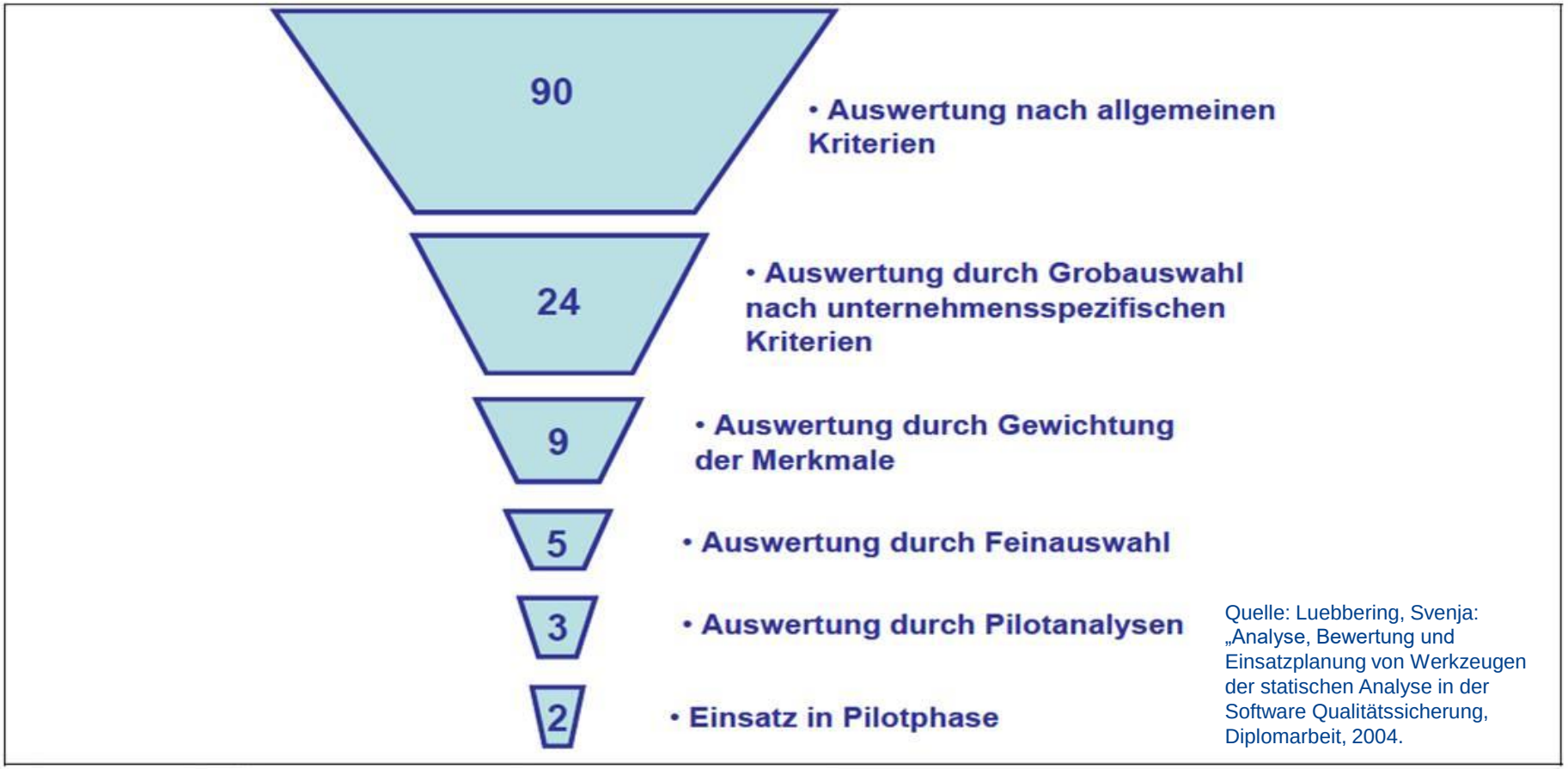
- Harmoniert das Werkzeug mit den Testobjekten und der Entwicklungsumgebung?
- Wurden die sonstigen Leistungsmerkmale, die das Werkzeug in die finale Auswahl gebracht haben, in der Praxis tatsächlich erreicht?
- Kann der Herstellersupport qualifizierte Auskunft und Hilfe auch bei Nichtstandardfragen liefern?

Prinzipien der Werkzeugauswahl



- Bewertung der Reife einer Organisation,
  - Erarbeitung eines Stärken-Schwächen-Profils
  - Identifikation von Verbesserungspotentialen, die durch Testwerkzeuge erreicht werden können
- Evaluation gegen klar spezifizierte Anforderungen und objektive Kriterien
  - Ermöglicht objektive Auswahl eines Testwerkzeugs
- Evaluation des Werkzeuges (proof-of-concept), um die geforderte Funktionalität zu prüfen und um zu ermitteln, ob das Produkt die gesetzten Ziele erfüllen kann
- Evaluation der Anbieter (einschließlich Trainingsunterstützung, Support und der kommerziellen bzw. vertraglichen Aspekte)
- Identifikation der internen Anforderungen für das Coaching und die Schulung der Anwendung des Werkzeugs

Werkzeugauswahl



Quelle: Luebbering, Svenja: „Analyse, Bewertung und Einsatzplanung von Werkzeugen der statischen Analyse in der Software Qualitätssicherung, Diplomarbeit, 2004.

Werkzeug-unterstützung für das Testen

Überlegungen zu Testwerkzeugen

Klassifizierung von Testwerkzeugen

Typen von Testwerkzeugen

Nutzen und Risiken der Testautomatisierung

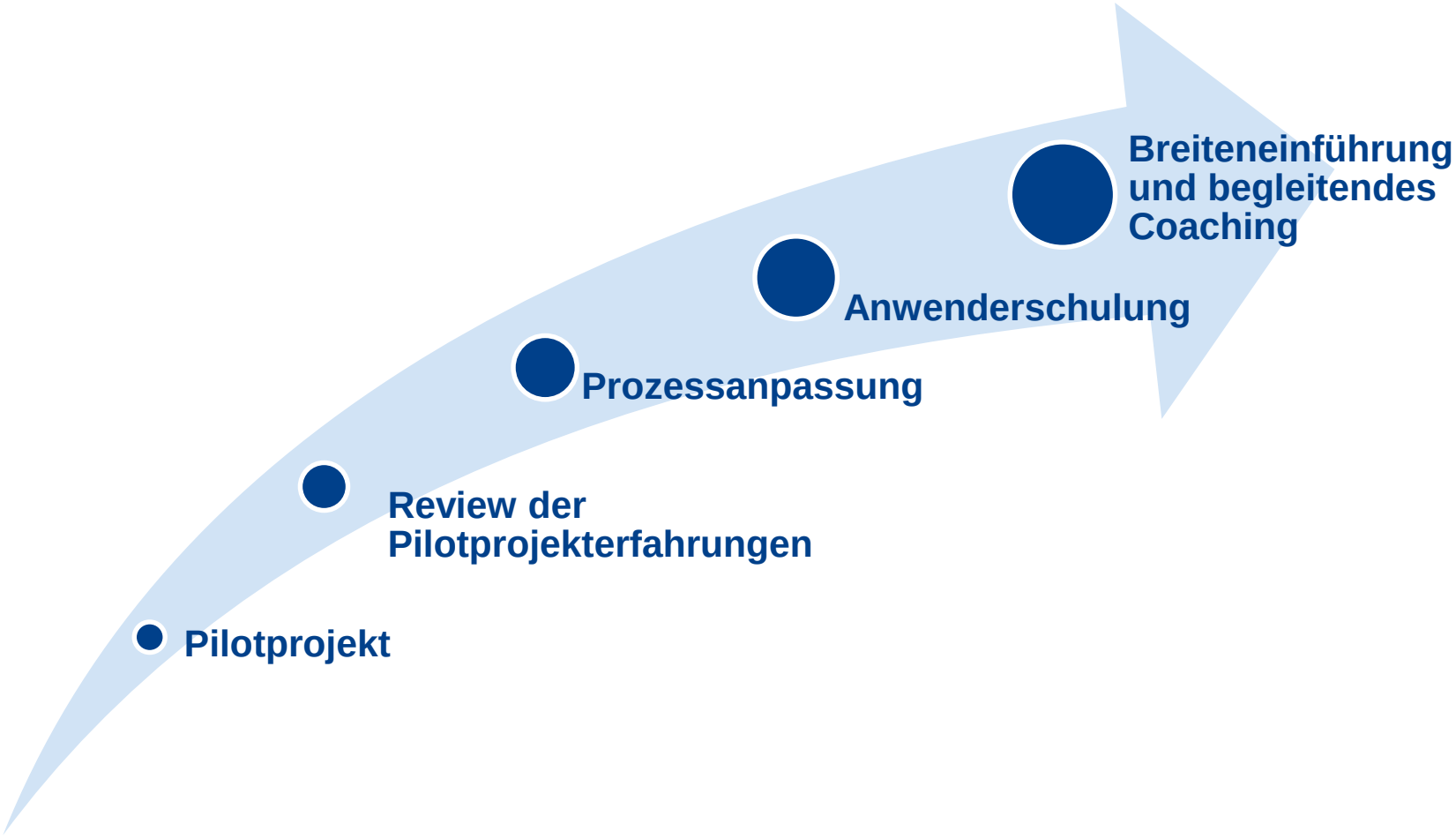
Effektive Nutzung von Werkzeugen

Auswahl von Werkzeugen

Pilotprojekte für die Einführung eines Werkzeugs

Erfolgsfaktoren für Werkzeuge

Schritte der Werkzeugeinführung



Pilotprojekt

- Durchführung Pilotprojekt zur Werkzeug-Evaluierung in realem Projektumfeld
- Ziele
  - Genaues Kennenlernen der Stärken und Schwächen des Werkzeugs
  - Passt das Werkzeug in existierende Werkzeug- und Prozesslandschaft?
  - Ermittlung des Anpassungsbedarfs
  - Entscheidung über die Standardisierung des Werkzeugeinsatzes (z.B. Namenskonventionen für Dateien und Tests, Neuanlage von Bibliotheken und die Festlegung von modularen Testsuiten)
  - Realistische Kosten-Nutzen-Bewertung
  - Ermittlung des Schulungsbedarfs
- Sollte nicht durch die Personen durchgeführt werden, welche die Werkzeugauswahl vorgenommen haben (Objektivität)
- Breiteneinführung (Roll-out) erst nach der Evaluierung des Pilotprojekts möglich
- unzureichend ausgewählte oder eingeführte Werkzeuge werden nicht genutzt!

Werkzeug-unterstützung für das Testen

Tipps für die erfolgreiche Einführung eines Werkzeugs

- Testwerkzeug schrittweise einführen!
- Realistische Erwartungshaltung: Alle von der Werkzeugeinführung Betroffenen rechtzeitig informieren
  - Welchen Einfluss hat das Testwerkzeug auf die eigene Arbeit?
  - Welche Gründe gibt es für die Einführung?
  - Was kann durch das neue Testwerkzeug erreicht werden, was nicht?
- Adaptierung und Prozessverbesserung müssen mit dem Testwerkzeug harmonieren
- Bedarf an Schulung, Coaching und Anleitung der Benutzer des Werkzeugs muss richtig ermittelt und umgesetzt werden
- Richtlinien für die Werkzeugbenutzung definieren
- Sammeln und Auswerten von Nutzungsdaten und Feedback aus der Werkzeugbenutzung standardisieren, um aus Erfahrungen zu lernen
- Werkzeugverwendung und den tatsächlichen Nutzen überwachen

Weitere Erfolgsfaktoren für Werkzeuge

- Sicherstellen, dass das Werkzeug technisch und organisatorisch in den Softwareentwicklungslebenszyklus integriert ist
  - Kann unterschiedliche Organisationen betreffen, die für den Betrieb und/oder Drittanbieter verantwortlich sind
- Eine Kombination mehrerer Werkzeugen verwenden, um ein Ziel zu erreichen, das kein einzelnes Werkzeug alleine erreichen kann
- Spezifische Werkzeuge für die zu testende Domäne (Finanzwesen, Automotive, …) sind oft einfacher und weniger fehleranfällig anwendbar
- Die Programmiersprache des Werkzeugs sollte den Entwicklern bereits bekannt (und beliebt) sein, damit diese beim Schreiben der Skripte etc. helfen können
- Die Testumgebung automatisiert einrichten, damit keine Tests fehl-schlagen, weil die Umgebung falsch eingerichtet wurde (falsch-negative)

Zusammenfassung (1 von 3)

- Für jede Phase im Testprozess sind Werkzeuge zur Qualitätsverbesserung und/oder zur Automatisierung der Testarbeiten verfügbar
- Testwerkzeuge können unterschiedlich klassifiziert werden
  - nach Aktivitäten des Testprozesses, die unterstützt bzw. automatisiert werden
  - ob das Werkzeug das Verhalten des Testobjekts beeinflusst oder nicht
  - wer das Werkzeug typischerweise benutzt
- Unterschiedliche Ansätze zur Automatisierung der Testdurchführung
  - Capture & Replay-Ansatz: „Ad-hoc“ Automatisierung
  - Datengetriebenes Testen: Trennung von Testdaten und Testvorgehensspezifikation
  - Schlüsselwortgetriebenes Testen: Trennung von Testentwurf und Testrealisierung

Zusammenfassung (2 von 3)

- Der Einsatz von Testwerkzeugen verspricht zahlreiche Vorteile, z.B. die Einsparung von Ressourcen oder die Erhöhung der Transparenz der Testdokumentation
- Dem potentiellen Nutzen stehen häufig hohe Investitionskosten gegenüber
- Die Auswahl eines Testwerkzeugs sollte sorgfältig vorbereitet und gegen klar spezifizierte Anforderungen erfolgen, um eine objektive Auswahl zu gewährleisten
- Das alleinige Vorhandensein eines Werkzeugs garantiert nicht, dass die Anwendern es auch nutzen. Beteiligung an der Auswahl sowie Information und Schulung der Betroffenen erhöhen die Akzeptanz des Testwerkzeugs

Zusammenfassung (3 von 3)

- Vor der unternehmensweiten Einführung eines Testwerkzeugs Erfahrung im Rahmen eines Pilotprojekts sammeln
- Bei der Einführung der verschiedenen Arten von Testwerkzeugen am besten intellektuell anspruchsvolle Aktivitäten zuletzt automatisieren

- **Der Einsatz von Testwerkzeugen setzt einen funktionierenden Testprozess voraus!**
- Die Werkzeuge folgen dem Prozess! Nicht anders herum.

Folgende Fragen sollten Sie jetzt beantworten können

- Welche Typen von Testwerkzeugen gibt es?
- Was ist der Zweck der Werkzeugunterstützung für den Test?
- Was bedeutet es, wenn ein Testwerkzeug intrusiv ist?
- Welche grundlegenden Funktionen haben Testmanagementwerkzeuge?
- Welche grundlegenden Funktionen haben Testausführungswerkzeuge?
- Welche grundlegenden Funktionen haben Fehlermanagementwerkzeuge?
- Welche typische Funktionalität bieten Werkzeuge für die Unterstützung des Reviewprozesses?
- Welche Eigenschaften des Programmcodes können mit Hilfe von Werkzeugen zur statischen Analyse typischerweise analysiert werden?
- Welche Typen von Testdatengeneratoren gibt es?

Folgende Fragen sollten Sie jetzt beantworten können

- Welche unterschiedlichen Ansätze zur Automatisierung der Testdurchführung gibt es?
- Welche Werkzeuge sind für die Unterstützung der Entwickler beim Komponenten- bzw. Integrationstest geeignet?
- Welche Vorteile und Risiken sind mit einer Werkzeugeinführung verbunden?
- Welche Tätigkeiten spielen bei der Einführung eines Testwerkzeugs eine Rolle?
- Welche sind die Ziele eines Pilotprojekts im Rahmen der Einführung eines Testwerkzeugs?

Muster-Prüfungsfragen

Testen Sie Ihr Wissen…

Frage 1

39. Welche der folgenden Aussagen beschreibt am EHESTEN einen Vorteil für die Nutzung eines Testausführungswerkzeugs? [K1]

a)	Es ist einfach, Regressionstests zu erstellen.	☐
b)	Es ist einfach, die Versionen von Testobjekten zu kontrollieren.	☐
c)	Es ist einfach, Testfälle für Zugriffssicherheitstests zu entwerfen.	☐
d)	Es ist einfach, Regressionstests durchzuführen.	☐

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019

Frage 1 - Lösung

39. Welche der folgenden Aussagen beschreibt am EHESTEN einen Vorteil für die Nutzung eines Testausführungswerkzeugs? [K1]

a)	Es ist einfach, Regressionstests zu erstellen.	☐
b)	Es ist einfach, die Versionen von Testobjekten zu kontrollieren.	☐
c)	Es ist einfach, Testfälle für Zugriffssicherheitstests zu entwerfen.	☐
d)	Es ist einfach, Regressionstests durchzuführen.	☒

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019

Frage 2

40. Welches Testwerkzeug (A-D) zeichnet sich durch die folgende Klassifizierung (1-4) aus? [K2]

1. **Werkzeugunterstützung zur Verwaltung von Tests und Testmitteln.**
2. **Werkzeugunterstützung für statische Tests.**
3. **Werkzeugunterstützung für die Testdurchführung und Protokollierung.**
4. **Werkzeugunterstützung zur Performanzmessung und dynamischen Analyse.**

A. **Überdeckungsanalysatoren**
B. **Konfigurationsmanagementwerkzeuge**
C. **Reviewwerkzeuge**
D. **Testmonitore**

a)	1A, 2B, 3D, 4C.	☐
b)	1B, 2C, 3D, 4A.	☐
c)	1A, 2C, 3D, 4B.	☐
d)	1B, 2C, 3A, 4D.	☐

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019

Frage 2 - Lösung

40. Welches Testwerkzeug (A-D) zeichnet sich durch die folgende Klassifizierung (1-4) aus? [K2]

1. **Werkzeugunterstützung zur Verwaltung von Tests und Testmitteln.**
2. **Werkzeugunterstützung für statische Tests.**
3. **Werkzeugunterstützung für die Testdurchführung und Protokollierung.**
4. **Werkzeugunterstützung zur Performanzmessung und dynamischen Analyse.**

A. **Überdeckungsanalysatoren**
B. **Konfigurationsmanagementwerkzeuge**
C. **Reviewwerkzeuge**
D. **Testmonitore**

a)	1A, 2B, 3D, 4C.	☐
b)	1B, 2C, 3D, 4A.	☐
c)	1A, 2C, 3D, 4B.	☐
d)	1B, 2C, 3A, 4D.	☒

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019