Willkommen zur Vorlesung

Methodische Grundlagen des Software-Engineering

im Sommersemester 2012

Prof. Dr. Jan Jürjens

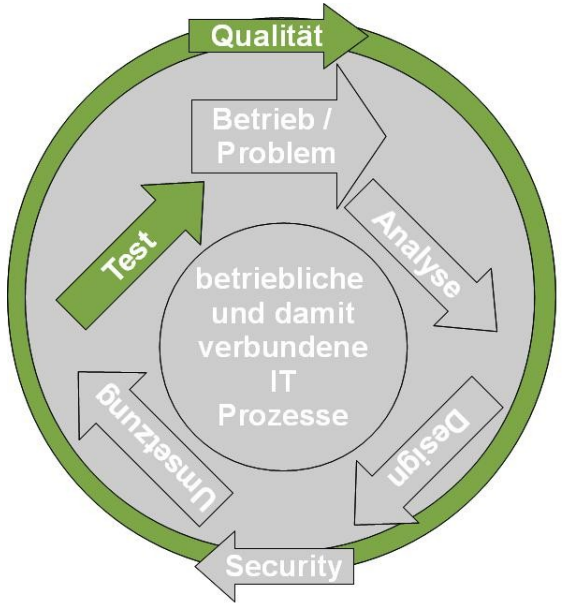
TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

4.7 Testwerkzeuge

Basierend auf dem Foliensatz
„Basiswissen Softwaretest - Certified Tester“
des „German Testing Board“
(nach Certified Tester Foundation Level Syllabus,
deutschsprachige Ausgabe, Version 2011)
(mit freundlicher Genehmigung)

Einordnung
Testwerkzeuge

- Geschäfts-Prozesse
- Qualitätsmanagement
- Testen
  - Einführung
  - Grundlagen Softwaretesten
  - Testen im Softwarelebenszyklus
  - Statischer Test
  - Black-Box-Test
  - White-Box-Test
  - Test-Management
  - Testwerkzeuge
  - Fuzzing
- Sicheres Software Design



4.7 Testwerkzeuge

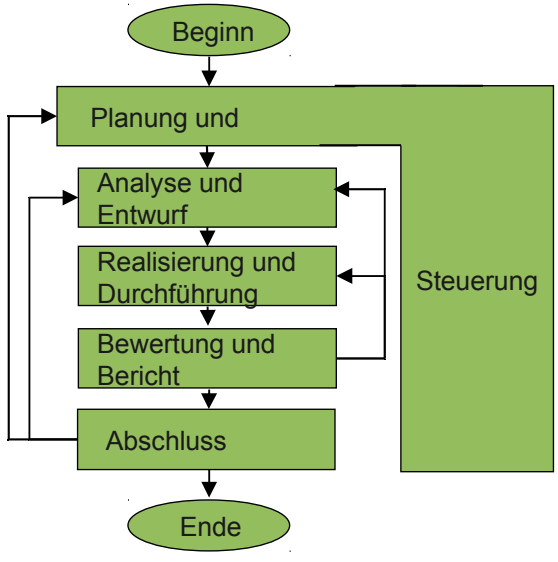
4.7 Test-werkzeuge

- Typen von Testwerkzeugen
- Effektive Anwendung von Werkzeugen: Potenzieller Nutzen und Risiken
- Auswahl und Einführung von Testwerkzeugen in eine Organisation

Typen von Testwerkzeugen (1)

In Anlehnung an den Begriff CASE-Tools werden Testwerkzeuge auch CAST-Tools (Computer Aided Software Testing) genannt. Unterschiedliche **Klassifikationen** möglich:

- Nach der unterstützen Aktivität bzw. Phase des Testprozesses
- Nach der unterstützen Teststufe
- Nach der unterstützen Testart
- Nach der unterstützen Testrolle
- Nach der unterstützten Technologie
- Nach kommerziell / Open-Source / Shareware



Klassifikation nach den unterstützten Aktivitäten:
Testwerkzeuge abhängig von der unterstützten Aktivität bzw. Phase des Testprozesses typisiert.

- Einzelne Testwerkzeuge unterstützen meist nur einen kleinen Ausschnitt des Testprozesses.
- Testwerkzeug-Familien oder Testwerkzeug-Suiten decken viele bzw. sämtliche Aktivitäten des Testprozesses ab und können als Einheit betrachtet werden.
- Innerhalb eines Testprojekts wird selten die ganze Bandbreite an Werkzeugen eingesetzt.

Typen von Testwerkzeugen (2)

Klassifikation nach Beeinflussung des Testobjekts: Intrusive und nicht intrusive Werkzeuge.

- **Intrusive Testwerkzeuge:** verändern oder beeinflussen das Verhalten des Testobjekts (auch als „Untersuchungseffekt“ bezeichnet).
  - Beispiel: Durch Instrumentierung eingefügter Code kann das Performance-Verhalten oder den Speicherbedarf beeinflussen.
- **Nicht-intrusive Testwerkzeuge:** verändern das Testobjekt nicht und beeinflussen dessen Verhalten nicht.

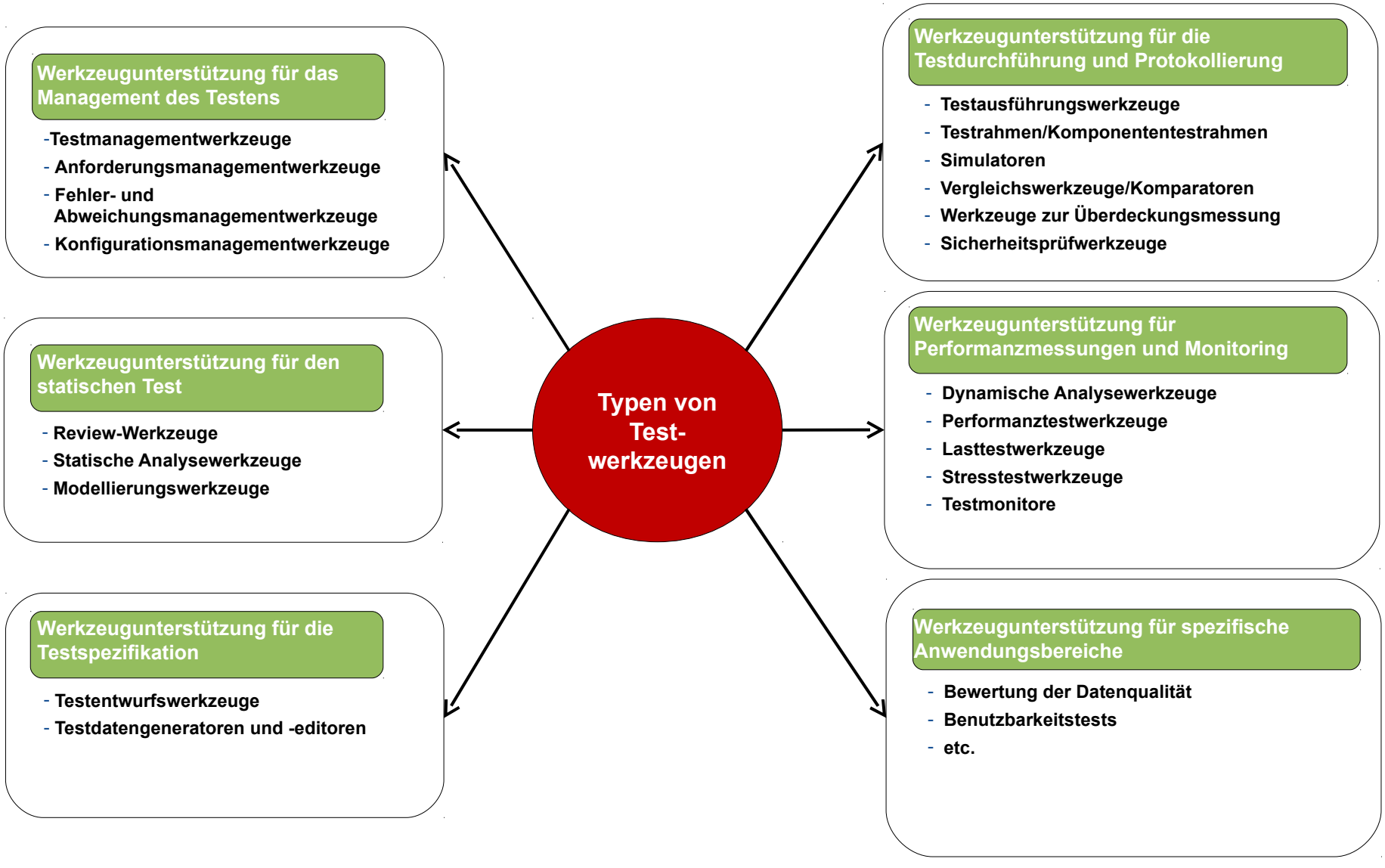
Klassifikation nach Hauptzielgruppe bzw. Hauptanwender des Werkzeugs:

- Wer soll das Werkzeug benutzen ? Tester ? Testleiter ? Entwickler zur Unterstützung des Komponenten- / Integrationstests ?
- Zahlreiche Werkzeuge unterstützen den Softwareentwicklungsprozess:
  - Werkzeuge speziell für die Unterstützung von Testern.
  - Werkzeuge, die nicht nur von Testern eingesetzt werden. Diese Werkzeuge können auch andere Aktivitäten des Softwareentwicklungsprozesses, z. B. das Anforderungsmanagement unterstützen.

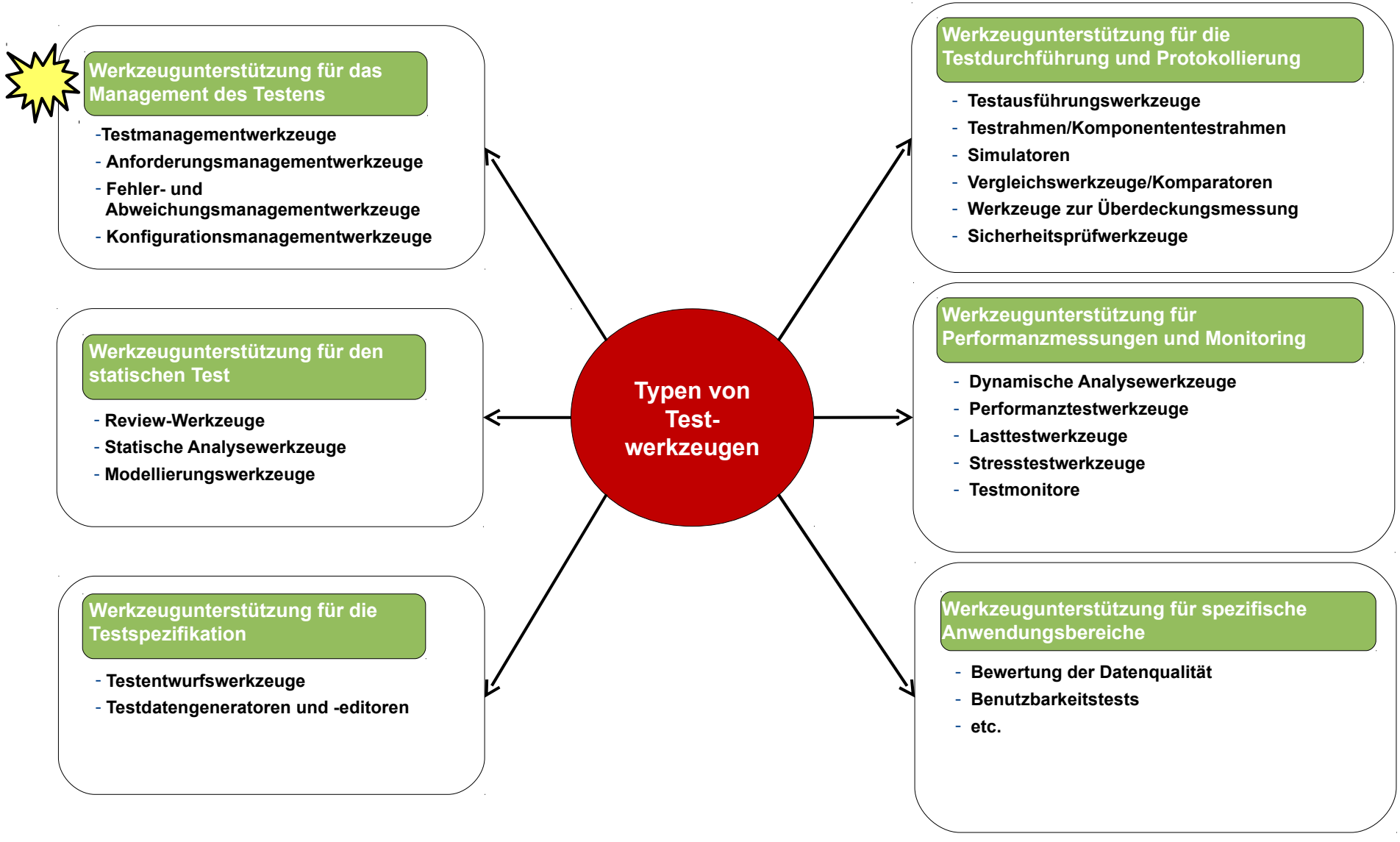
Der Begriff „**Testframework**“ wird mit unterschiedlichen Bedeutungen verwendet:

- wiederverwendbare Testbibliotheken (auch als Testrahmen bezeichnet)
- die Art des Entwurfs der Testautomatisierung (z.B. schlüsselwortgetrieben, datengetrieben)
- den gesamten Prozess der Testdurchführung

Typen von Testwerkzeugen (3)



Werkzeugunterstützung für das Management des Testens



Anzahl von Testfällen oft drei- bis vierstellig, daher Testplanungswerkzeuge für:

- **Erfassung, Katalogisierung, Verwaltung und Priorisierung von Testfällen**.

Fortgeschrittene Funktionalität:

- Erfassung / Import von Systemanforderungen (Requirements)
- Traceability (Rückverfolgbarkeit) von Tests, Testergebnissen und Vorfällen zu Anforderungen
- Testfallstatus (wie oft durchgeführt, mit welchem Resultat)
- Aufzeichnung von Testergebnissen und Erstellung von Fortschrittsberichten
- Zeit- und Ressourcenplanung
- Testfortschrittsüberwachung durch quantitative Analyse (Metriken)
- Abdeckungsprüfung (z.B. mindestens ein Test pro Anforderung)
- Eigenständige Versionskontrolle oder Schnittstelle zu einem externen Konfigurationsmanagementwerkzeug
- Weitere Schnittstellen zu Testausführungswerkzeugen sowie zu Fehler- und Abweichungsmanagementwerkzeugen

Anforderungsmanagement-werkzeuge

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

Fehler- und Abweichungs-managementwerkzeuge

Hunderte bis Tausende von Fehlermeldungen, daher Fehler- und Abweichungsmanagement-werkzeuge (engl. defect tracking) für:

- **Erfassung, Verwaltung, Verteilung von Abweichungen** (z.B. Fehlerzustände, Änderungsanforderungen, Fehlerwirkungen oder Anomalien)

Ermöglichen die Verfolgung der Abweichungen über die Zeit.

Unterstützen statistische Analysen und liefern Berichte über Abweichungen.

Fortgeschrittene Funktionalität:

- Parametrisierbare Fehler- bzw. Abweichungsstatusmodelle (Bearbeitungsstatus, Priorität etc.)
- Parametrisierbare Testfallstatusmodelle (z.B. geplant, spezifiziert, automatisiert, durchgeführt, blockiert)
- Workflow-Funktionalität für den Fehlerbehebungsprozess (z.B. Zuordnung von Aufgaben zu bestimmten Personen und/oder Personengruppen für Fehlerbehebung oder Nachtest)

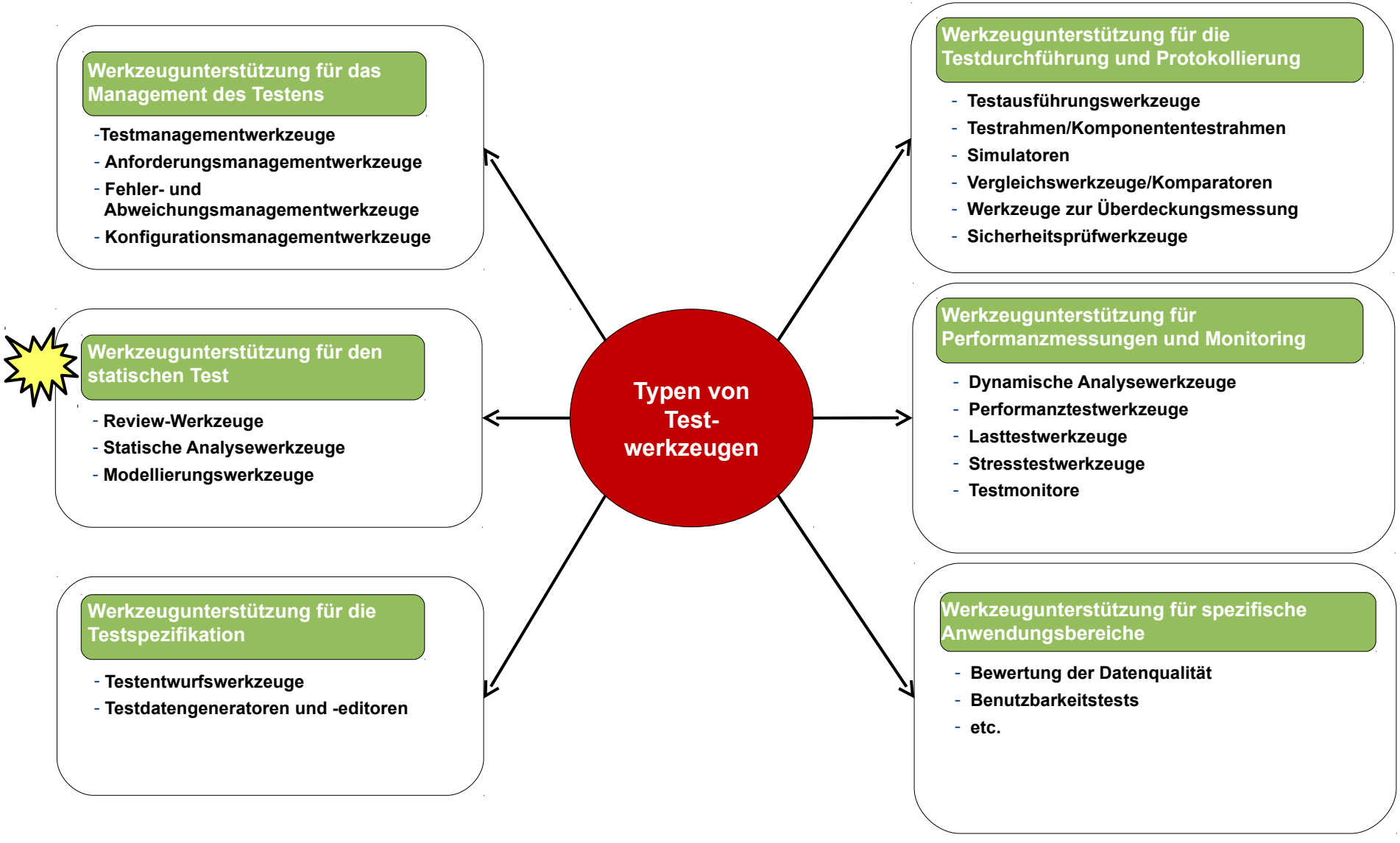
Häufig gekoppelt mit oder integraler Bestandteil von Testmanagementwerkzeugen

- Planung für Fehlernachtest, Fehlerstatistik und -analyse
- Generierung von Testdokumentation

Konfigurationsmanagement-werkzeuge

- Keine Testwerkzeuge im engeren Sinn, können als Testunterstützungswerkzeuge 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 Compilern, Browsern und Rechnern).

Werkzeugunterstützung für den statischen Test



Werkzeuge für den Reviewprozess

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

Auswertung (data collection):

- Unterstützung der Sammlung und Auswertung von Reviewergebnissen durch Metriken (z.B. gefundene Abweichungen oder geleisteter Aufwand)

Unterstützung der Reviewsitzung (meeting support):

- Synchron vs. Asynchron
- Funktionalität zur verteilten Kommunikation (z.B. Nachrichtendienste, Chat, Videokonferenzen, Kalender, usw.)
- Verteilte Reviews (online Reviews)

Werkzeuge zur statischen Analyse

Analysieren verschiedene **Charakteristika des Programmcodes**:

- Strukturelle Eigenschaften (z.B. Zyklomatische Zahl, Vererbungstiefe etc.)
- 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 den Elementen im Quellcode)

Modellierungswerkzeuge

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 Entwicklungs-prozess.

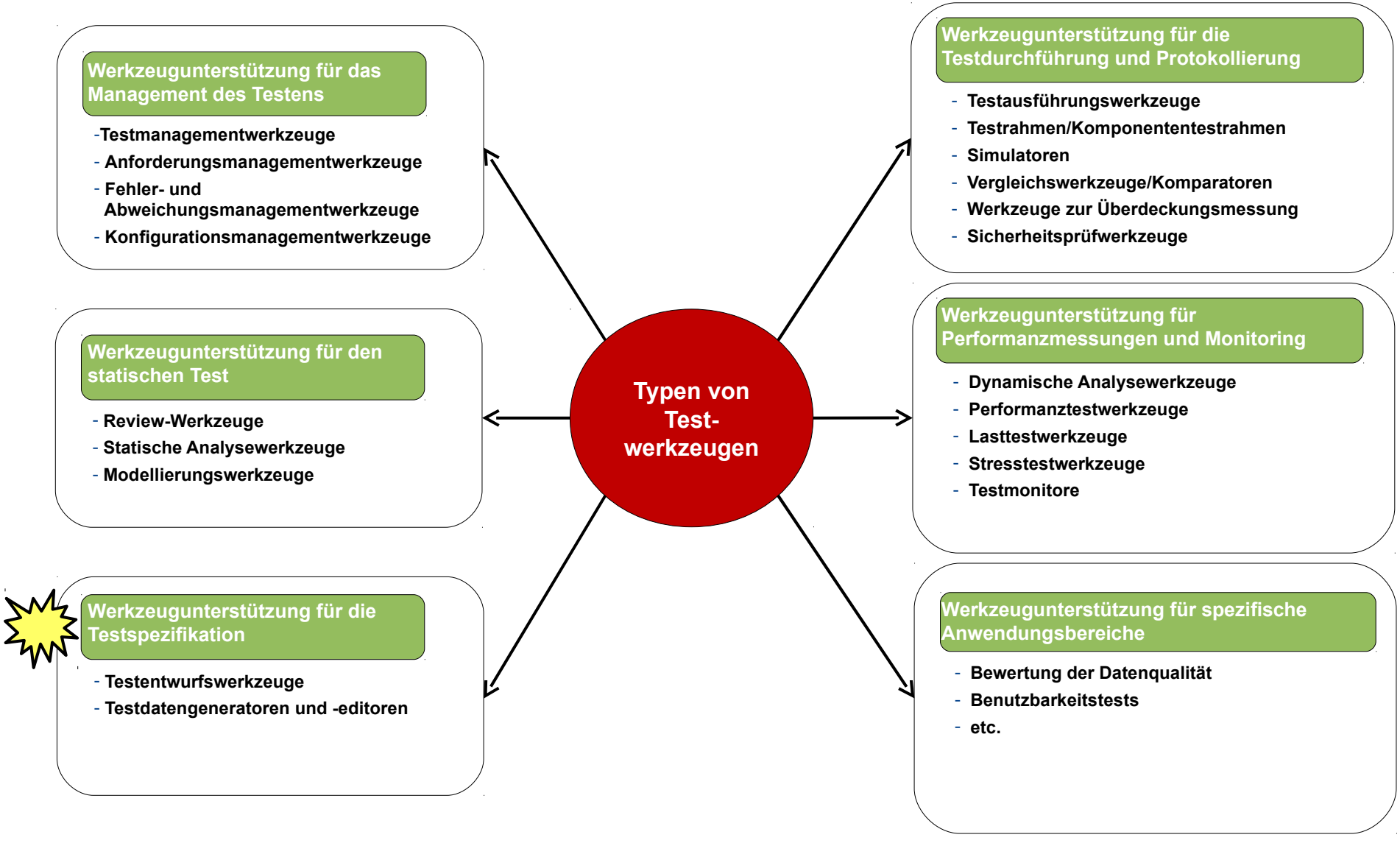
Fortgeschrittene Funktionalität:

- 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

Werkzeugunterstützung für die Testspezifikation



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 Sollreaktionen manuell ergänzt werden
- Keine Garantie für »gute« Testfälle
- Generierung strukturierter Vorlagen (Templates, Testrahmen)
- Testentwurfswerkzeuge decken meistens nur Teilaspekte der zu testenden Software ab

Testdatengeneratoren können wie folgt unterschieden werden: 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)

Codebasiert:

- Analysieren Code (Parametertypen, Kontroll-/Datenfluss, ...) des Testobjekts
- Keine Generierung von Sollwerten (bzgl. Spezifikation)
- Kein Erkennen von vergessenem Code

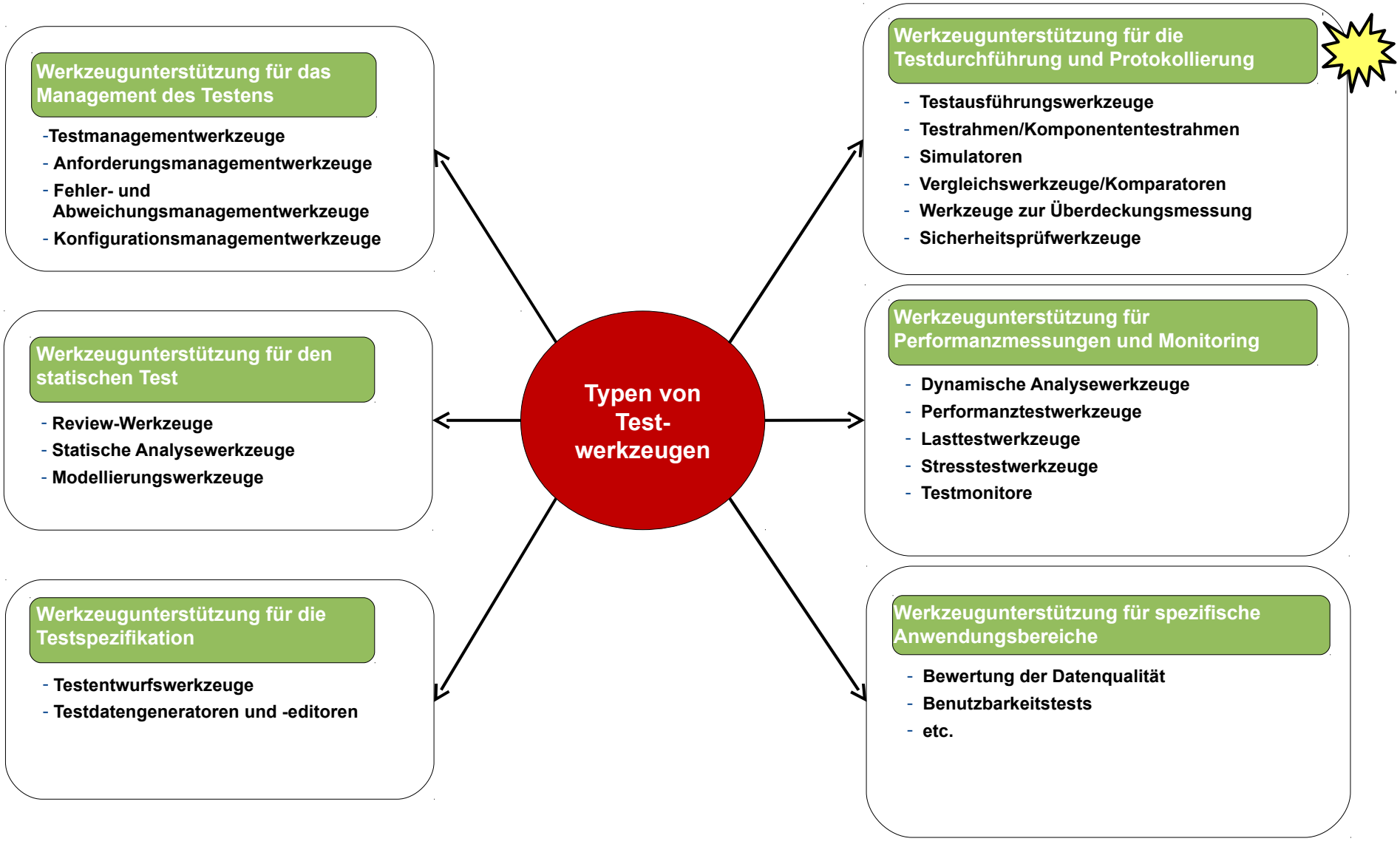
Schnittstellenbasiert:

- Analysieren Testobjektschnittstelle (z.B. API oder GUI)
- Analysieren der Parametertypen und Anwendung von Äquivalenzklassen- und Grenzwertanalyse
- Keine Generierung von Sollwerten
- Gute Eignung für Negativtests

Spezifikationsbasiert (Anforderungsbasiert):

- Ableitung von Testdaten aus der Spezifikation, die hierfür in einer formalen Notation vorliegen muss

Werkzeugunterstützung für die Testdurchführung und Protokollierung



Automatische oder halbautomatische Ausführung von Testfällen:

- Versorgung des Testobjekts mit Testdaten
- Aufzeichnung der Reaktionen des Testobjekts
- Protokollierung des Testlaufs

Müssen in der Lage sein, die Testschnittstelle des Testobjekts anzusprechen, die abhängig von der Teststufe ist (Komponenten-, Integrations-, Systemtest).

Verschiedene Typen:

- Testausführungswerkzeuge
- Testrahmen/Komponententestrahmen
- Simulatoren
- Vergleichswerkzeuge/Komparatoren
- Werkzeuge zur Überdeckungsmessung
- Sicherheitsprüfwerkzeuge

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

Ansätze zur Automatisierung der Testausführung (1)

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.

Testskript:

```
Maske »Vertragsdaten« aufrufen;
Daten für Kunde »Müller« erfassen;
Checkpunkt setzen;
Vertrag »Müller« in Vertragsdatenbank speichern;
Maske »Vertragsdaten« löschen;
Vertrag »Müller« aus Vertragsdatenbank lesen;
Maskeninhalt mit Checkpunkt vergleichen;
```

Ansätze zur Automatisierung der Testausführung (2)

Datengetriebener Ansatz:

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

Schlüsselwortgetriebener Ansatz:

- 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.

Testrahmen / Komponententestrahmen

Sammlung aller Programme (u. a. Platzhalter und Testtreiber), die notwendig sind, um Testfälle auszuführen, auszuwerten und Testprotokolle aufzuzeichnen.

Wird benötigt,

- wenn einige Komponenten noch nicht zur Verfügung stehen
- wenn eine kontrollierte Umgebung für die Fehlerlokalisierung benötigt wird

Platzhalter (Stubs) werden beim Komponenten- und Integrationstest benötigt, um noch nicht implementierte Komponenten für die Testdurchführung zu ersetzen bzw. zu simulieren.

Testtreiber:

- Sprechen Testobjekte über deren Programmierschnittstelle an
- Einsatzgebiet eher Komponenten- und Integrationstest als Systemtest
- Generische Testtreiber oder Testrahmengeneratoren:
  - Spezialisiert auf Programmiersprachen oder Entwicklungsumgebungen
  - Automatische Generierung von Testrahmen
  - Evtl. Generierung von Stubs
  - Funktionalität zur Abfrage von Reaktionen des Testobjekts und zur Protokollierung des Testablaufs
  - Erleichtern die Programmierung einer Testumgebung erheblich

Simulatoren (Scheinobjekte)

Möglichst umfassende und realitätsnahe **Nachbildung der Produktivumgebung**, 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 Handhelds)

Vergleichswerkzeuge / Komparatoren

Dienen dem Vergleich zwischen **erwartetem und aktuellem Ergebnis**.

Verarbeiten marktgängige Datei- und Datenbankformate.

Filtern relevante von irrelevanten Daten durch Filtermechanismen.

Testausführungswerkzeuge verfügen über Komparatoren für:

- GUI-Objekte
- Bildschirminhalte
- Konsoleninhalte

Werkzeuge zur Überdeckungsmessung

- Ermitteln die verschiedenen **Codeüberdeckungsmaße** (Anweisungsüberdeckung, Entscheidungsüberdeckung, Funktionsaufrufe etc.)
- Erstellen Statistiken oder graphische Darstellungen (z.B. Einfärben des Source-Codes) der erreichten Abdeckungen

Sicherheitsprüfwerkzeuge

Bewerten inwieweit die Software Sicherheitsaspekte wie

- die Vertraulichkeit,
- die Integrität der Daten,
- die Bestätigung der Echtheit,
- die Autorisierung,
- die Verfügbarkeit und
- die Nichtabstreitbarkeit

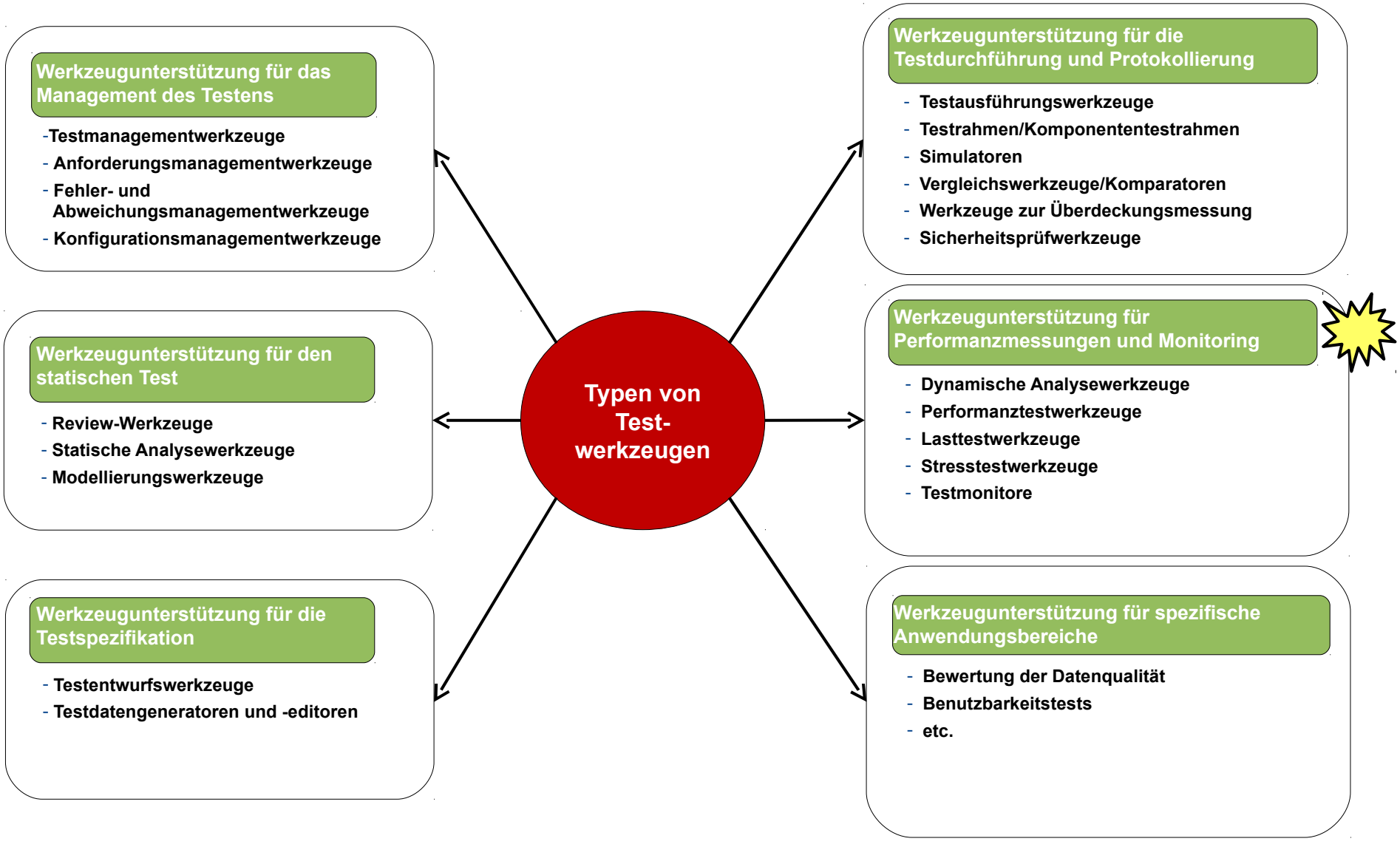
Schützt und analysieren Systeme im Hinblick auf evtl. Sicherheitslücken.

Testen die Abwehr eines Rechners oder eines Netzwerks z.B. gegen Angriffe von Computerviren oder Denial of Service Angriffe (Hackerattacken).

Viele Sicherheitsprüfwerkzeuge sind keine Testwerkzeuge im engeren Sinne, z.B. Firewalls, die bei den Sicherheitsprüfungen eingesetzt werden können.

Meistens technologie- und plattformabhängig.

Werkzeugunterstützung für Performanzmessungen und Monitoring



Dynamische Analysewerkzeuge

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.

Performanz- / Last- / Stresstestwerkzeuge

Testen nicht-funktionale Anforderungen.

Werkzeuge für **Performanztest** 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, zum Beispiel:

- Datenbankanfragen
- Benutzertransaktionen
- Netzwerkverkehr

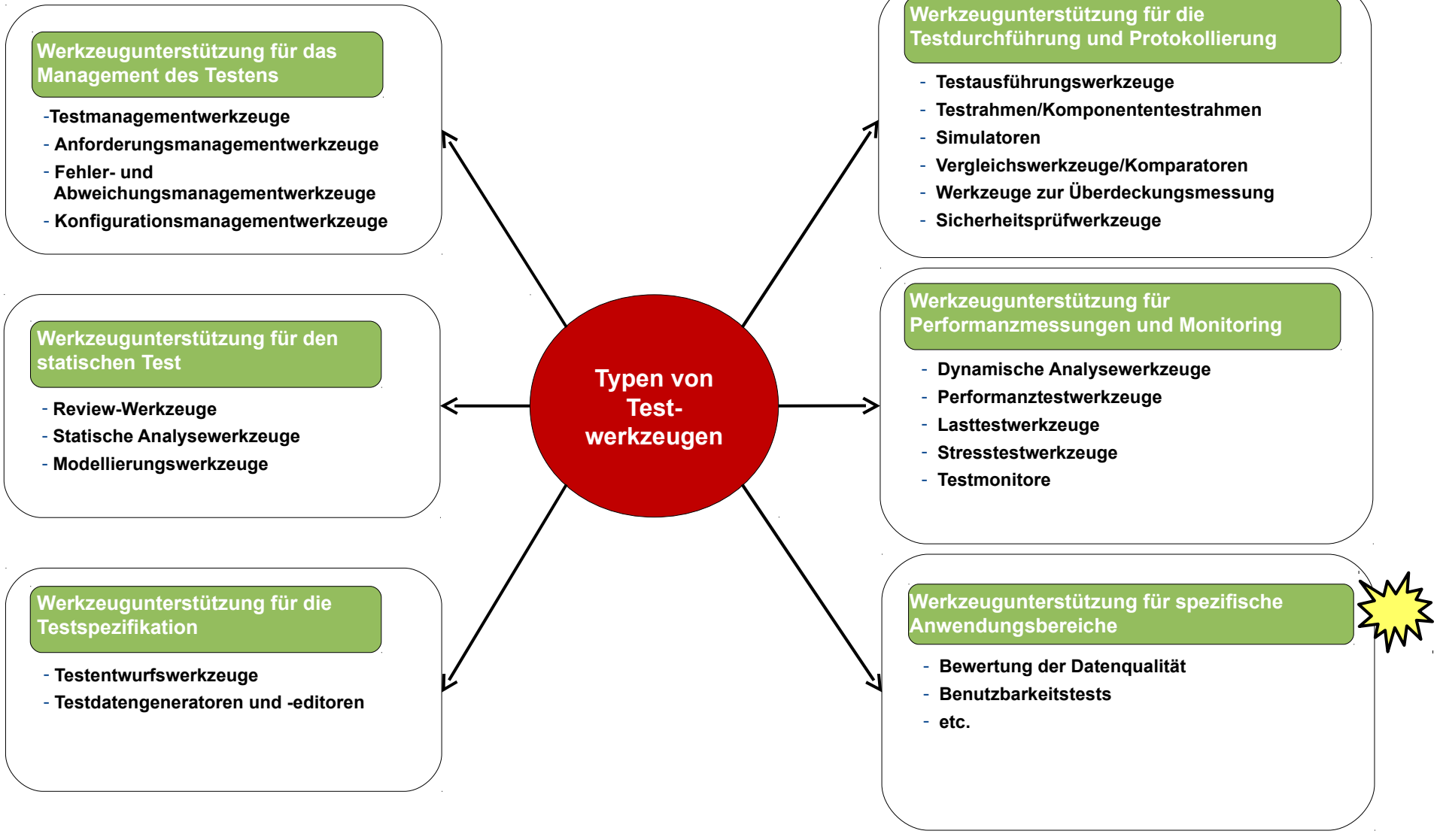
Stresstestwerkzeuge prüfen das Systemverhalten bei Überlastung (z.B. durch Betrieb mit zu hohem Datenvolumen oder auch gezielte Fehlbedienung).

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.

Werkzeugunterstützung für spezifische Anwendungsbereiche



Werkzeugunterstützung für spezifische Anwendungsbereiche

Unterstützen bestimmte Typen von Applikationen, z.B.:

- Performanzwerkzeuge speziell für web-basierte Applikationen
- Werkzeuge zur Unterstützung von GUI-Tests
- Testwerkzeuge für Protokolltests
- Statische Analysewerkzeuge für bestimmte Entwicklungsplattformen
- Dynamische Analysewerkzeuge für die Prüfung von spezifischen Sicherheitsaspekten
- Kommerzielle Werkzeug-Suiten für eingebettete oder sicherheitskritische Systeme
- Werkzeuge zur Bewertung der Datenqualität
  - Bei Datenkonvertierungs- und Migrationsprojekten oder bei Anwendungen wie Data Warehouses stehen die Daten im Mittelpunkt
  - Werkzeuge werden beispielsweise eingesetzt um die Daten zu validieren oder um Datenkonvertierungs- und Migrationsvorschriften zu prüfen
- Werkzeuge für Benutzbarkeitstests
  - Werkzeuge, die Mausbewegungen aufzeichnen
  - Werkzeuge, die Webseiten auf Barrierefreiheit prüfen

Weitere Werkzeuge (keine speziellen Testwerkzeuge)

Zahlreiche Werkzeuge, die in unterschiedlichen Kontexten, aber auch für das Testen verwendet werden können.

Zum Beispiel 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

4.7 Testwerkzeuge

4.7 Test-werkzeuge

- Typen von Testwerkzeugen
- **Effektive Anwendung von Werkzeugen: Potenzieller Nutzen und Risiken**
- Auswahl und Einführung von Testwerkzeugen in eine Organisation

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)

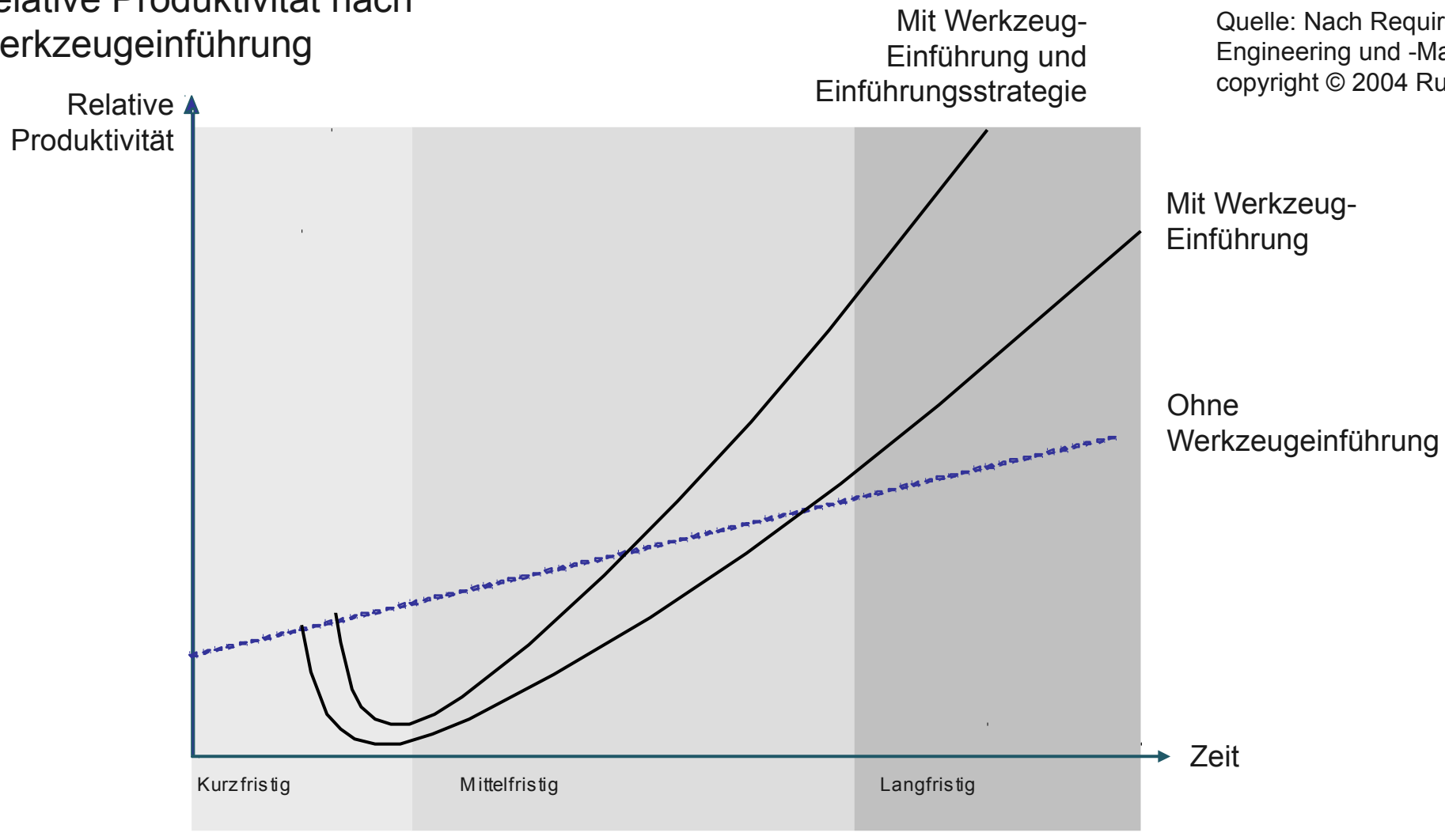
- Einführung von Testwerkzeugen nur bei vorhandenen, funktionierenden Prozessen möglich.
- Einführung von Testwerkzeugen häufig kurzfristig mit Produktivitäts-einbußen verbunden, daher keine Lösung für kurzfristige Personalengpässe.
- Unrealistische Erwartungen an Automatisierung und Werkzeug.
- Unterschätzung 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

Risiken von Testwerkzeugen (2)

- Blindes Vertrauen in das Werkzeug (Vernachlässigung oder sogar Ersatz für Testentwurf oder Ersatz für eigentlich besser geeignetes manuelles Testen).
- 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, etc.).

Lernkurveneffekt

Relative Produktivität nach Werkzeugeinführung

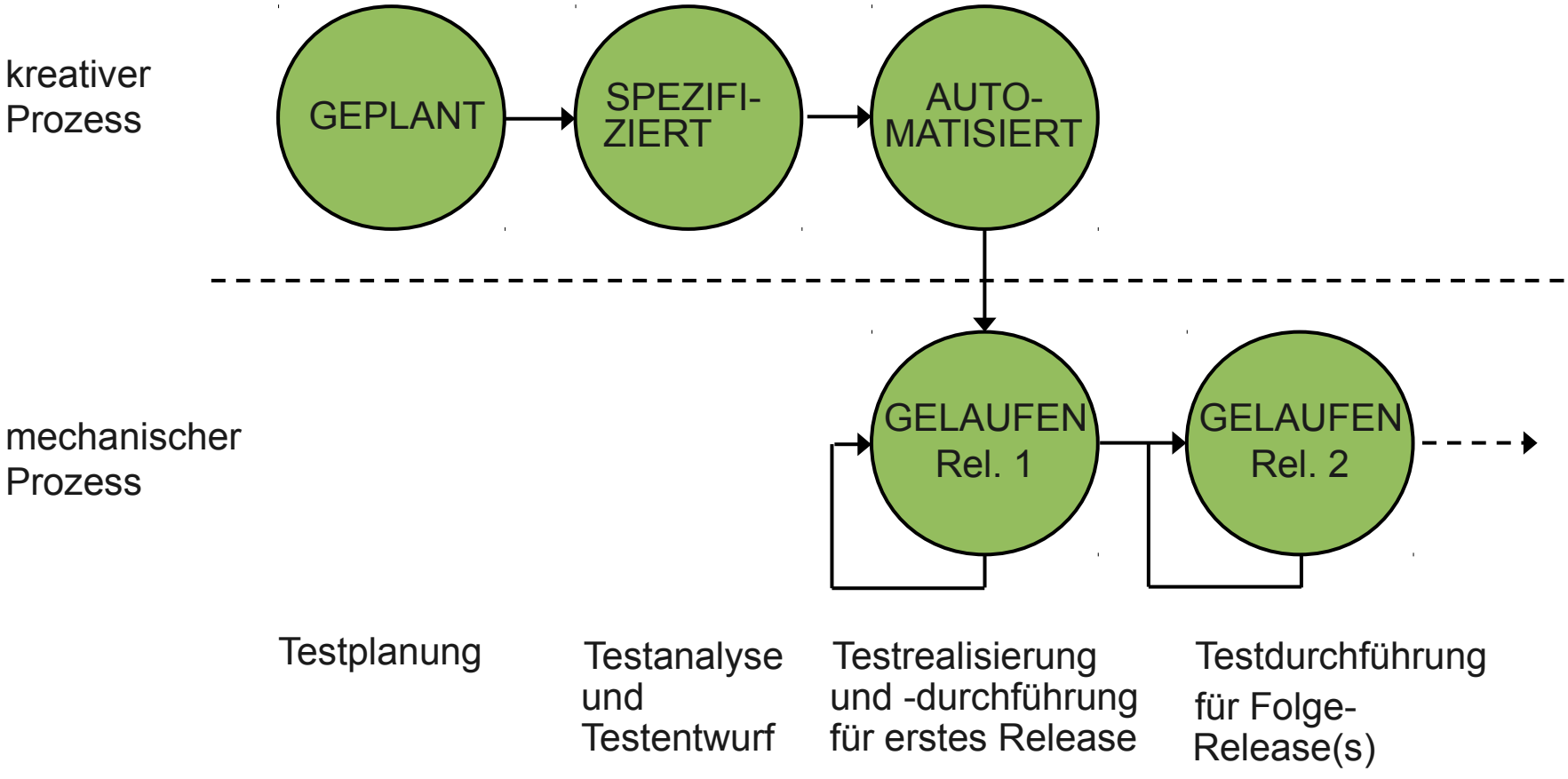


4.7 Testwerkzeuge

4.7 Test-werkzeuge

- Typen von Testwerkzeugen
- Effektive Anwendung von Werkzeugen: Potenzieller Nutzen und Risiken
- Auswahl und Einführung von Testwerkzeugen in eine Organisation

Lebenszyklus eines Testfalls

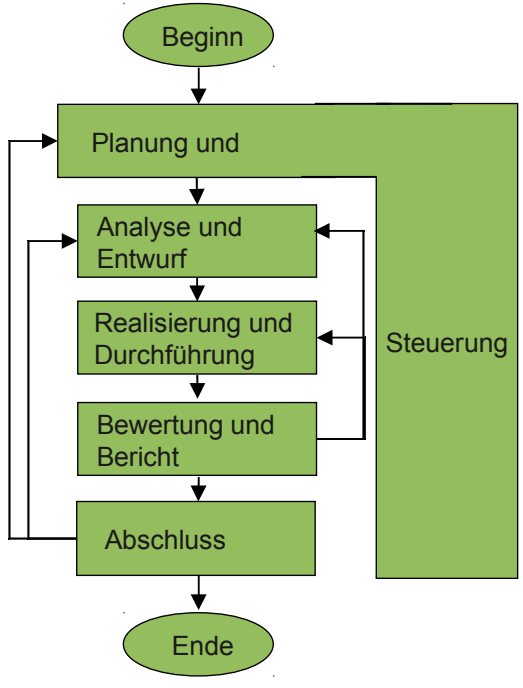


Einführungsreihenfolge von Testwerkzeugen

Bei der Einführung der verschiedenen Arten von Testwerkzeugen ist es empfehlenswert, sich an folgende Einführungsreihenfolge zu halten:

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

Im Vergleich zum Testprozess invertierte „Reihenfolge“ der Aktivitäten, intellektuell anspruchsvolle Aktivitäten immer zuletzt automatisieren!



Wirtschaftlichkeit der Werkzeugeinführung

Bei einer Werkzeugeinführung entstehen (meist) Kosten für Auswahl, Anschaffung, Wartung, Hardware, Mitarbeiterschulung. Wann amortisiert sich die Automatisierung des Testlaufs gegenüber der manuellen Durchführung?

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

n = Anzahl der Testläufe

T_M = Aufwand für eine manuelle Durchführung

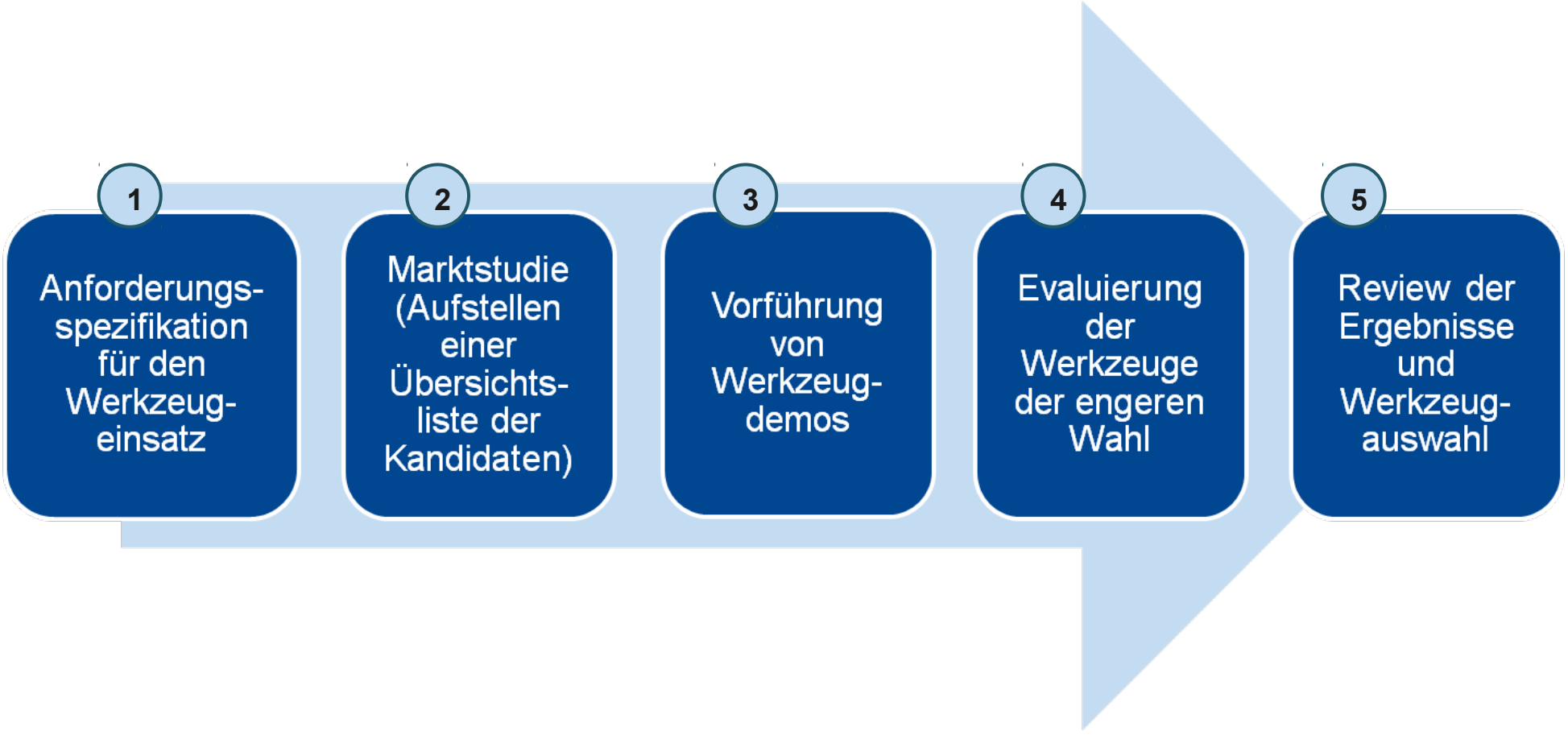
T_A = Aufwand für eine automatisierte Durchführung

A = Aufwand für die Automatisierung

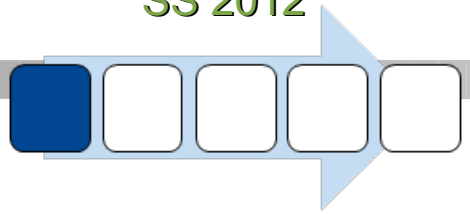
Aber: *Ersparnis* = $n * (T_M - T_A) - A$ ist in vielen Fällen zu einfach gedacht:

- Automatisierte Tests werden meist häufiger ausgeführt als ihre manuellen Gegenstücke.
- Testautomatisierung führt oft dazu, dass *mehr* Testfälle spezifiziert und getestet werden (was nicht automatisch eine Verbesserung bedeutet).
- Manche Tests lassen sich nicht manuell durchführen (z.B. Performanztest).
- Die Qualität der entstehenden Software wird durch die Automatisierung (in schwer quantifizierbaren Maße) beeinflusst, was aber in die Formel miteinbezogen werden müsste.

Prozess der Werkzeugauswahl

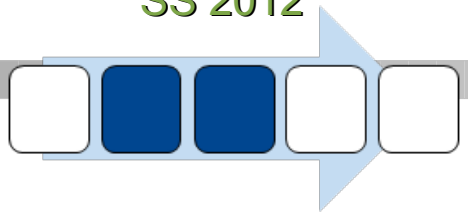


Kriterien der Anforderungsspezifikation



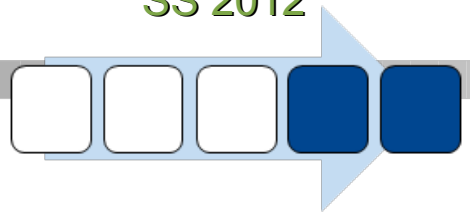
- Güte des Zusammenspiels mit den potentiellen Testobjekten
- Know-how der Tester über Werkzeug oder Methode
- Möglichkeit zur Integration in die vorhandene Entwicklungsumgebung
- Möglichkeit zur Integration mit anderen eingesetzten Testwerkzeugen
- Plattform, auf der das Werkzeug eingesetzt werden soll
- Integrationsmöglichkeiten mit Werkzeugen desselben Herstellers
- Service, Schulungsangebot, Verlässlichkeit und Marktstellung des Herstellers
- Lizenzbedingungen, Preis, Wartungskosten

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

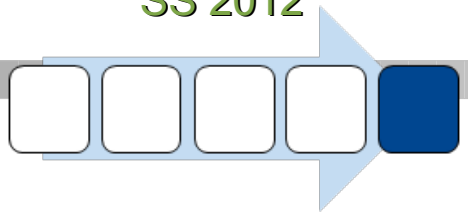
Evaluierung und Werkzeugauswahl



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

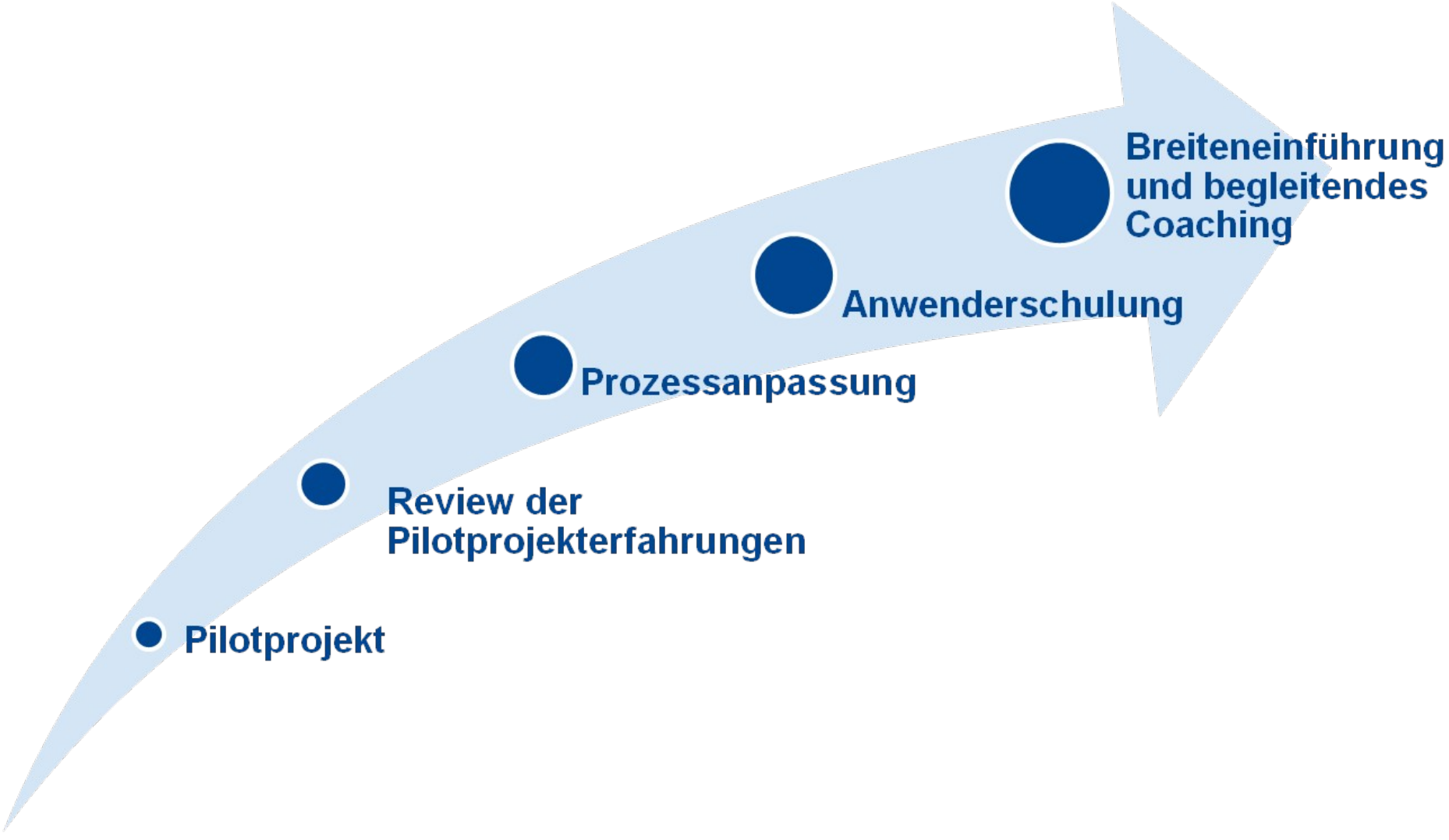
- Harmoniert das Werkzeug mit den Testobjekten und der Entwicklungsumgebung ?
- Wurden die sonstigen Leistungsmerkmale, die das Werkzeug in die finale Auswahl gebracht haben, in der Praxis tatsächlich erreicht ?
- Kann der Herstellersupport qualifizierte Auskunft und Hilfe auch bei Nichtstandardfragen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

Prozess der Werkzeugeinführung



Pilotprojekt

Durchführung eines Pilotprojekts zur Evaluierung des Testwerkzeugs in einem realen Projektumfeld. **Ziele:**

- Genaues Kennenlernen des Werkzeugs
- Analyse, inwieweit das Werkzeug mit den existierenden Werkzeugen und Prozessen zusammenpasst, Ermittlung des Anpassungsbedarfs
- Realistische Kosten-Nutzen-Bewertung
- Ermittlung des Schulungsbedarfs
- Entscheidung über die Standardisierung des Werkzeugeinsatzes (z.B. Namenskonventionen für Dateien und Tests, Neuanlage von Bibliotheken und die Festlegung von modularen Testsuiten)

Sollte nicht durch die Personen durchgeführt werden, welche die Werkzeugauswahl vorgenommen haben (Objektivität).

Erst nach der Evaluierung des Pilotprojekts wird die Breiteneinführung vorgenommen.

Ein unzureichend ausgewähltes oder eingeführtes Werkzeug endet im Schrank !

Tipps für eine erfolgreiche Breiteneinführung des Werkzeugs

- 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 und was nicht ?
- Adaptierung und Prozessverbesserung müssen mit dem Testwerkzeug harmonieren.
- Testwerkzeug soll schrittweise eingeführt werden.
- Schulungsbedarf muss richtig ermittelt und zugesichert werden.
- Richtlinien für die Werkzeugbenutzung müssen definiert werden.
- Sammeln und Auswerten von Feedback aus der Werkzeugbenutzung um aus Erfahrungen zu lernen.
- Werkzeugverwendung und der tatsächliche Nutzen sollen überwacht werden.

Zusammenfassung (1)

Für jede Phase im Testprozess sind Werkzeuge zur Qualitätsverbesserung und/oder zur Automatisierung der Testarbeiten verfügbar.

Testwerkzeuge können unterschiedlich klassifiziert werden:

- Je nach Aktivitäten des Testprozesses, die unterstützt bzw. automatisiert werden.
- Je nachdem, ob das Werkzeug das Verhalten des Testobjekts beeinflusst oder nicht.
- Je nachdem, wer das Werkzeug typischerweise benutzt.

Unterschiedliche Ansätze zur Automatisierung der Testdurchführung:

- Capture / Replay Ansatz: „Ad-hoc“ Automatisierung
- Datengetriebener Ansatz: Trennung von Testdaten und Testvorgehensspezifikation
- Schlüsselwortgetriebener Ansatz: Trennung von Testentwurf und Testrealisierung

Zusammenfassung (2)

- 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 sollte sorgfältig vorbereitet und gegen klar spezifizierte Anforderungen erfolgen, um objektive Auswahl des Testwerkzeugs zu gewährleisten.
- Das Vorhandensein eines Werkzeugs garantiert aber nicht, dass dieses von den Anwendern auch benutzt wird. Rechtzeitige Information und Schulung der Betroffenen erhöhen die spätere Akzeptanz des ausgewählten Testwerkzeugs.
- Bevor ein Testwerkzeug unternehmensweit eingeführt wird, sollte Erfahrung im Rahmen eines Pilotprojektes gesammelt werden.
- Bei der Einführung der verschiedenen Arten von Testwerkzeugen ist es empfehlenswert, intellektuell anspruchsvolle Aktivitäten zuletzt zu automatisieren.
- **Der Einsatz von Testwerkzeugen bedingt einen funktionierenden Testprozess !**

Folgende Fragen sollten Sie jetzt beantworten können

- Welche Typen von Testwerkzeugen werden unterschieden?
- Was ist der Zweck der Werkzeugunterstützung für den Test?
- Was bedeutet es, wenn ein Testwerkzeug intrusiv ist?
- Welche grundlegenden Funktionen werden den Testmanagementwerkzeugen zugeschrieben?
- Welche grundlegenden Funktionen werden den Fehler- und Abweichungsmanagementwerkzeugen zugeschrieben?
- Welche typische Funktionalität bieten Werkzeuge für die Unterstützung des Reviewprozesses an?
- Welche typischen Charakteristika des Programmcodes können mit Hilfe von Werkzeugen zur statischen Analyse analysiert werden?
- Welche Typen von Testdatengeneratoren werden unterschieden?
- Welche unterschiedlichen Ansätze zur Automatisierung der Testdurchführung werden unterschieden?
- 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 Pilotprojektes im Rahmen der Einführung eines Testwerkzeugs?

Nach diesem Abschnitt sollten Sie...

- Typen von Testwerkzeugen benennen und ihre grundlegende Funktionalität beschreiben können
- Den Begriff „Testwerkzeug“ und den Zweck der Werkzeugunterstützung für den Test erklären 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

Diese Begriffe sollten Sie kennen ...