

Softwaretest

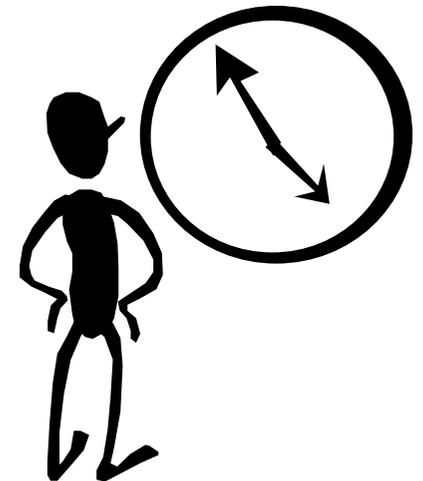
Software Engineering für große Informationssysteme

TU-Wien, Sommersemester 2004

Jürgen Lutz

Agenda

- Wozu überhaupt Softwaretest
- Das Problem (Fehler, error, bug)
- Testmethodik, Testarten
- Testfälle
- Testautomation

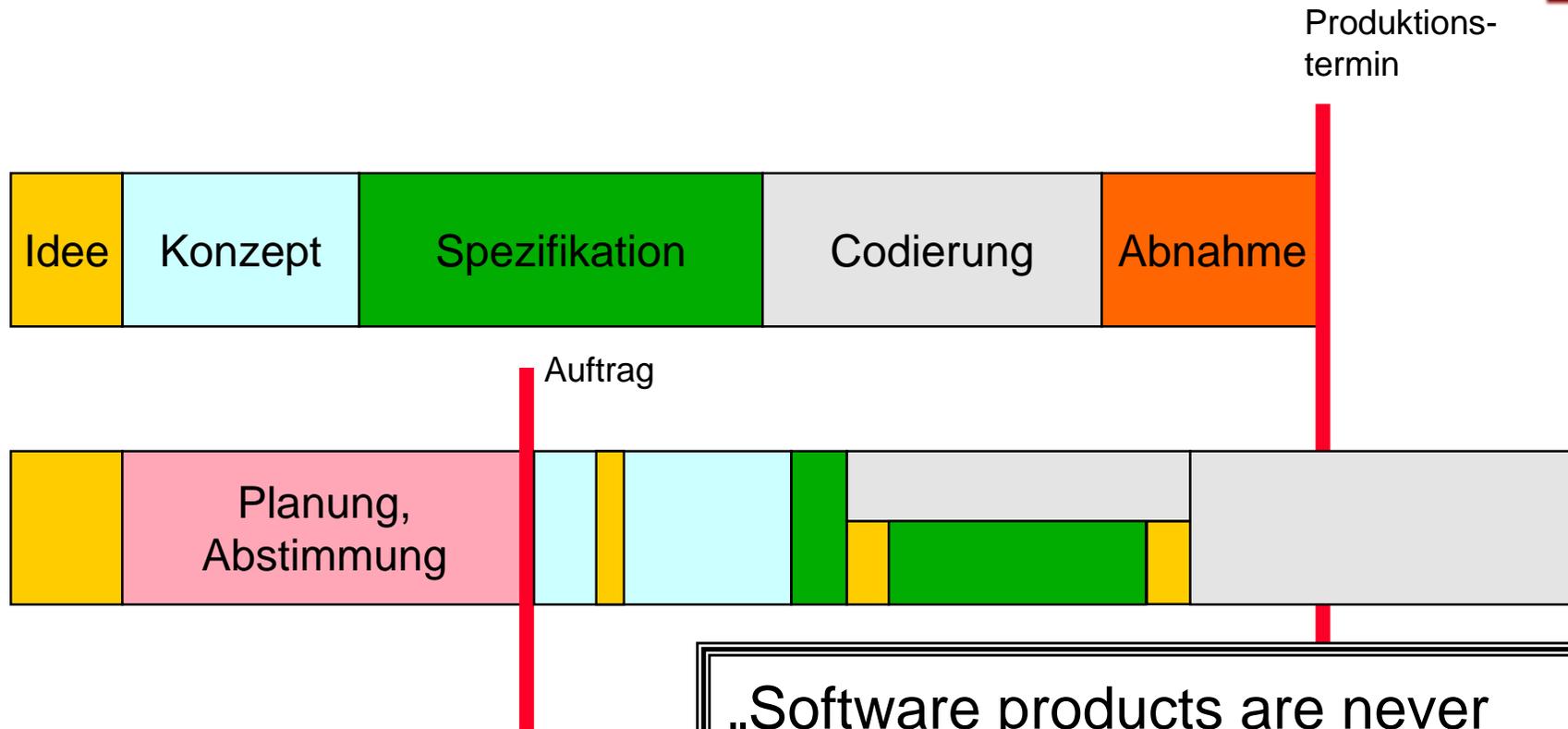


Wozu überhaupt Softwaretest?

- Wäre es nicht viel einfacher und billiger, bei der Softwareentwicklung Fehler zu vermeiden?
 - Es muss eine gründliche **Analyse- und Designphase** geben
 - Man muss **sauber und nach Spezifikationen entwickeln**, dann macht man kaum Fehler
 - Und wenn es doch Fehler gibt, **findet die der Entwickler** doch selber viel schneller



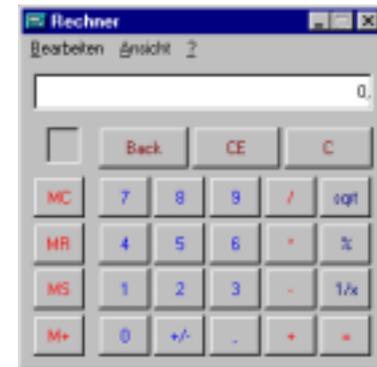
Ein Softwareprojekt



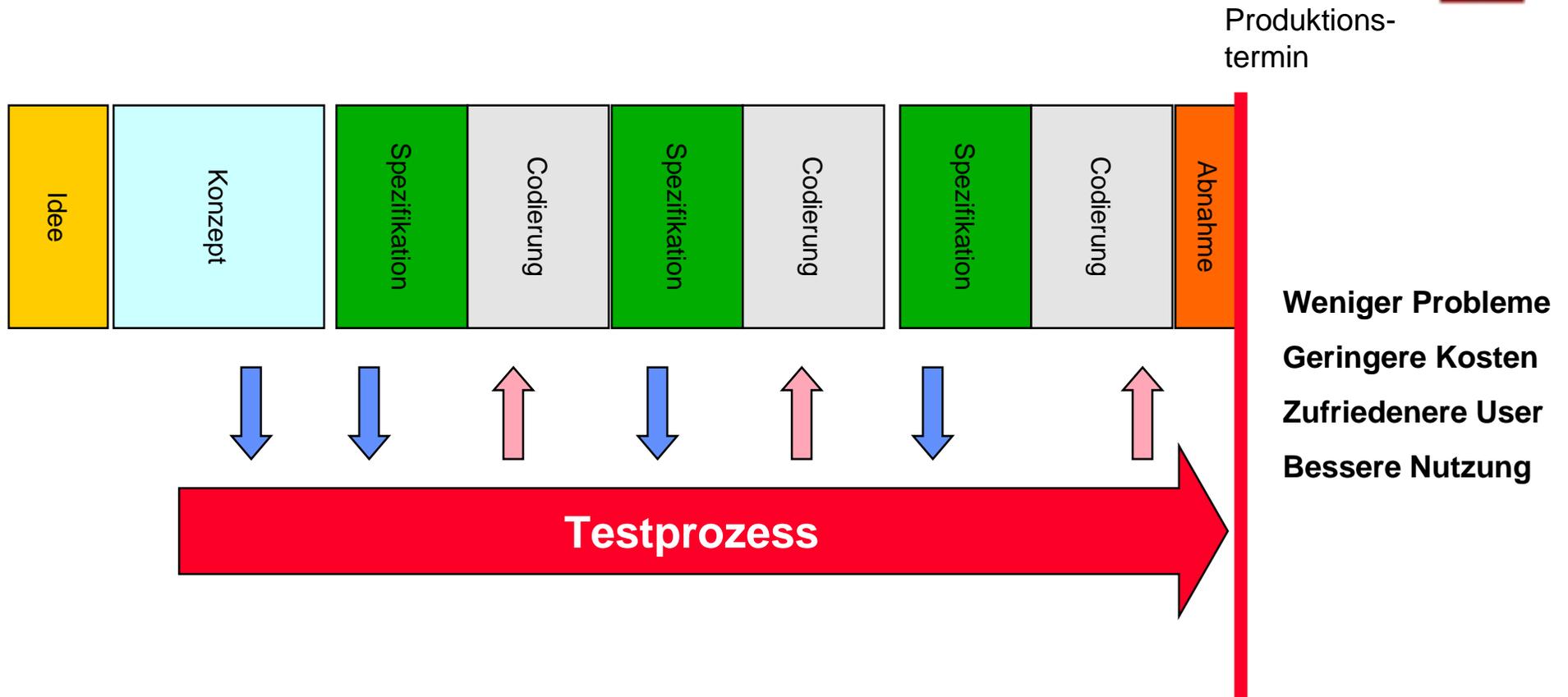
„Software products are never released – they escape!“ (Kaner, 1999)

Es wird teuer ohne Test

- In jedem Fachkonzept können Fehler gefunden werden
- Keine Spezifikation ist vollständig
- Jeder Entwickler produziert auch Fehler
- Es gibt keine fehlerfreie Software

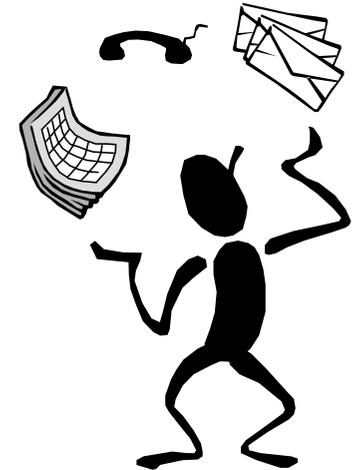


Ein Softwareprojekt



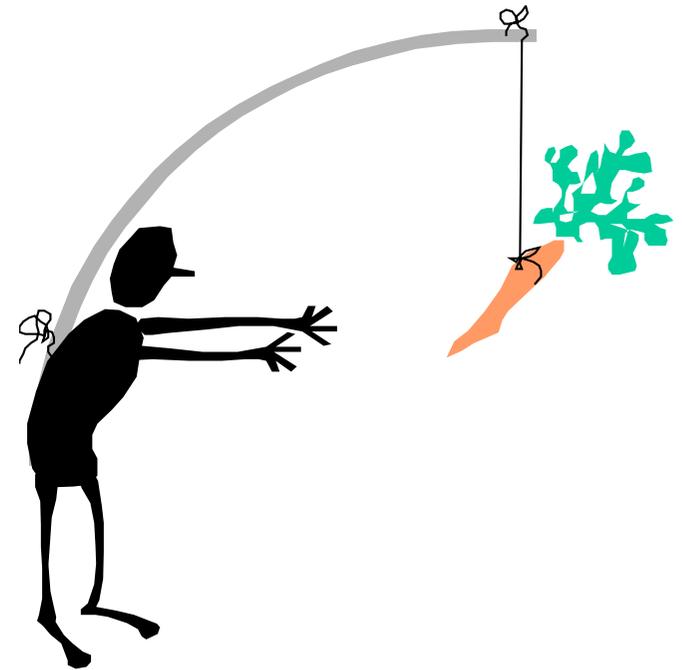
Test ist eine Dienstleistung

- Testplanung
- Testmanagement
- Testfallerstellung
- Testfalldurchführung
 - Manuell
 - Automatisiert
- Risikoanalyse
- Unterstützung des Projektmanagements



Was ist das Ziel von Softwaretest?

- Alle Fehler vor Produktiveinsatz finden und korrigieren
- Verifikation der geforderten Funktionalitäten
- ???



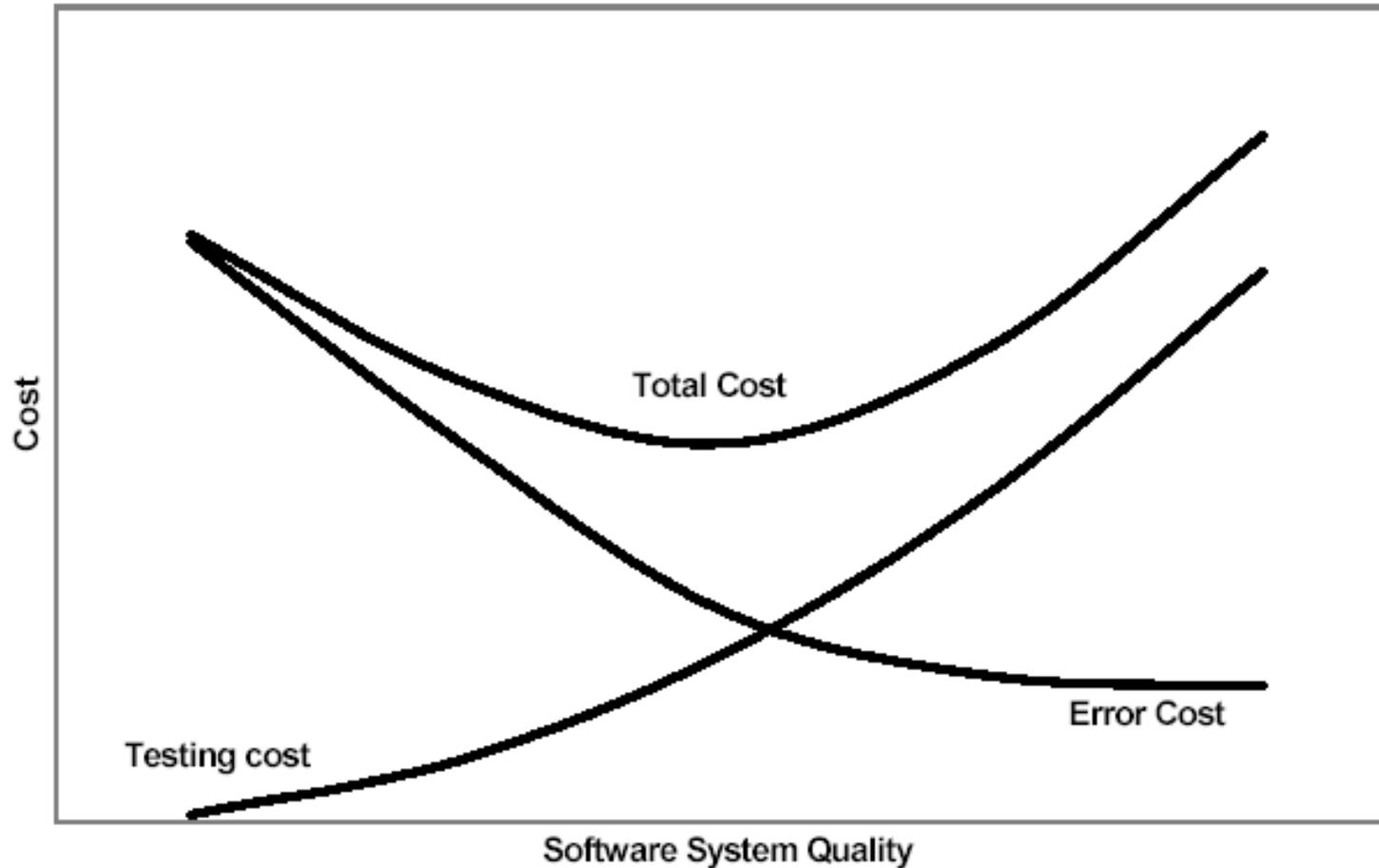
ZIEL: „Alle Fehler vor Produktiveinsatz finden...“

- Wie viele Fehler können theoretisch gefunden werden?
- Was ist überhaupt ein Fehler?
- Wie erkennt man einen Fehler?
- Ist es effizient jeden Fehler zu finden?



Fehlerfolgekosten

Testing and Error Costs Over Quality



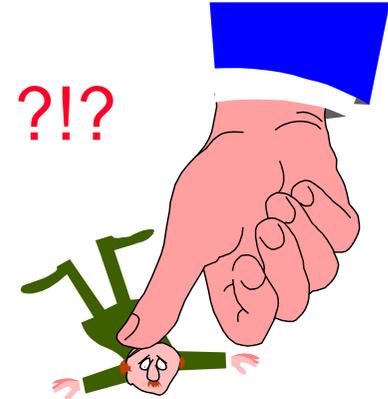
ZIEL: "... und korrigieren?"

- Kann jeder Fehler korrigiert werden?
 - Der Fehler muss nachvollziehbar sein
 - Der Fehler muss rechtzeitig gefunden werden
 - Die Behebung des Fehlers muss sich rentieren
- Organisatorische Probleme
 - Projektplan, Termindruck
 - Zuständigkeit muss geklärt sein
 - Berücksichtigung des Faktors „Mensch“



Fehler, Errors, Bugs...

Der Entwickler als ewiger Sündenbock !?!



- Es geht um die Behebung von Problemen, die den User bei der Anwendung der Software behindert.
- Diese Probleme müssen gefunden, berichtet und behoben werden.

... PTARs

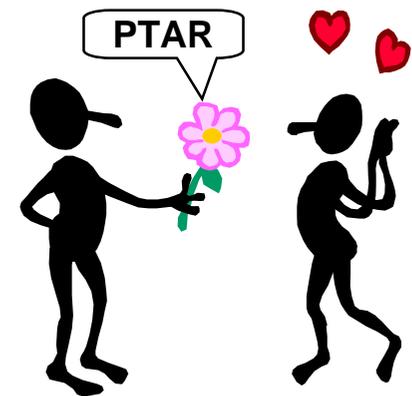
„**P**roblem **T**racking **A**nd **R**eporting“

=> Synonym für „dokumentiertes Problem“

„Ich habe dir einen PTAR geschickt“

vs.

„Du hast einen Fehler gemacht“



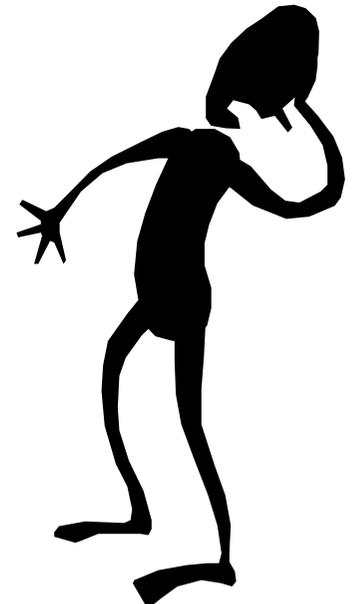
Was ist ein PTAR

- Technisches Problem
 - Eine Teilkomponente funktioniert nicht
 - Ein Programmteil reagiert falsch
 - Eine Schnittstelle liefert falsche Daten
- Fachliches Problem
 - Falscher / unvollständiger Workflow
 - Falsche Dateninterpretation
 - Unrichtige Berechnungen



Was kann noch ein PTAR sein

- Usability Probleme
- Schlechte Performance
- Bedienung nicht intuitiv
- Benutzerführung im Problemfall
- Unrichtiges / unvollständiges Hilfesystem
- Layout / Formatierung
- Rechtschreibfehler

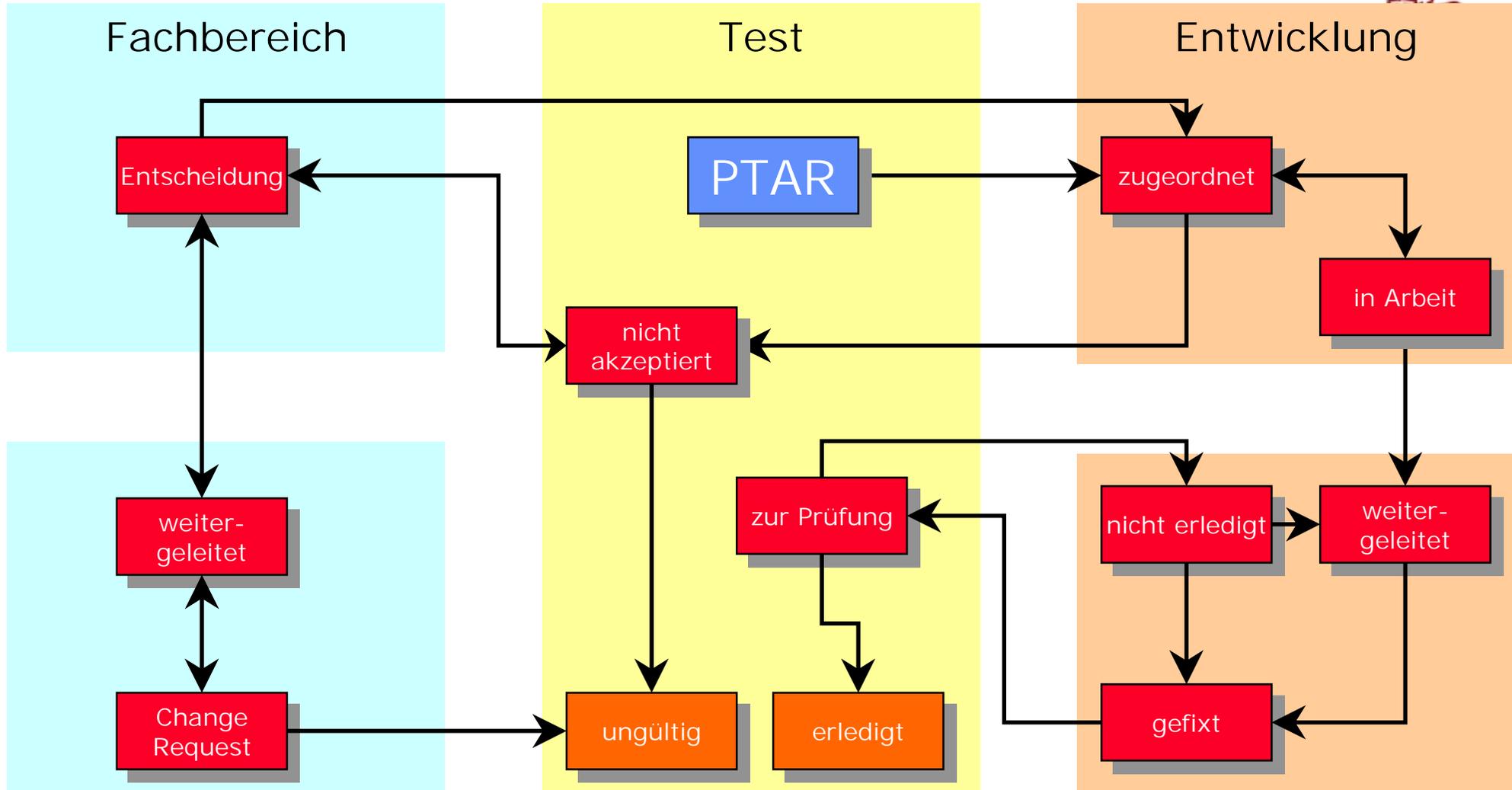


PTAR Behandlung

- Finden eines Problems
- Erkennen des Fehlers
- Soll/Ist Beschreibung des PTAR
- Kommunikation an Entwicklung
- Fachliche Entscheidung (change request ?)
- Zuordnung an zuständige Gruppe
- Kontrolle nach Korrektur
- Ablage



PTAR Workflow



ZIEL: „Die geforderte Funktionalität verifizieren?“

- Was wird überhaupt gefordert?
 - Spezifikationen, Fachkonzepte
 - Erwartbares Verhalten in vergleichbaren Applikationen
 - Vom User als offensichtlich erwartbar eingestuft
- Was kann überprüft werden?
- Ist es effizient, alles zu überprüfen?
 - Äquivalenzklassen
 - risk-based Vorgangsweise



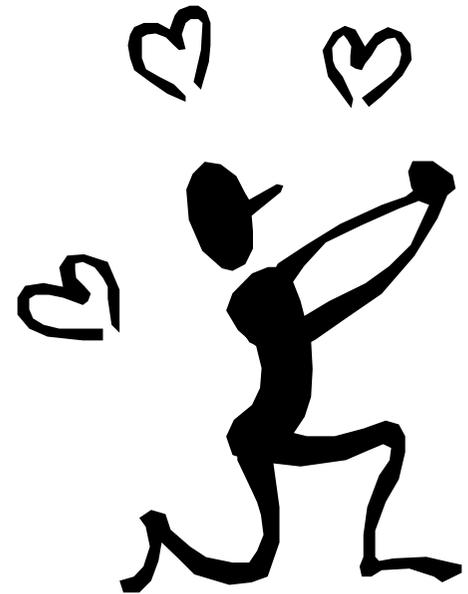
Ziele des Softwaretests

- Kürzere Entwicklungszeiten
- Bessere Planbarkeit (Termin & Kosten)
- Bessere Aufgabenverteilung
- Steigerung der Qualität
- Reduktion der Fehlerfolgekosten



Ziele des Softwaretests

- Dem Anwender einer IT-Lösung die erwarteten Funktionalitäten zum geplanten Zeitpunkt in ausreichendem Maße sicherstellen

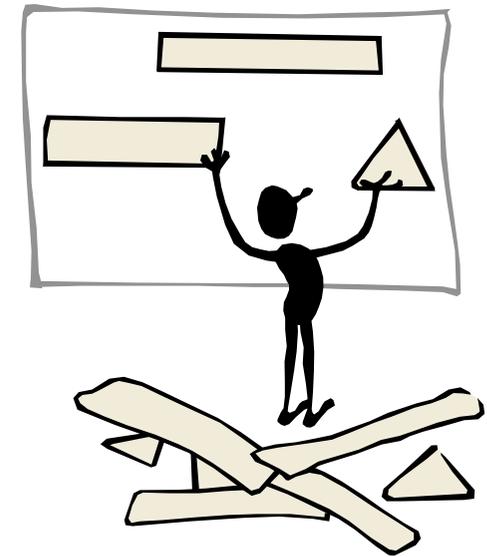


- Abweichungsmanagement
 - PTAR-System
- Testprojekt
 - Testkonzept
 - Messbarkeit von Testerfolg
- Testphasen, Testarten
- Testdurchführung, Testautomation



Projektabhängiges Testkonzept

- Arbeitsgrundlage für die Testabteilung
- Inhalte
 - Testinhalt, Basis der Testplanung
 - Testansatz
 - Testfallfindung
 - Testarten, Testautomation
 - Testbett, Testumgebungen
 - Ansprechpartner, Verantwortliche
 - Vorgehensmodell



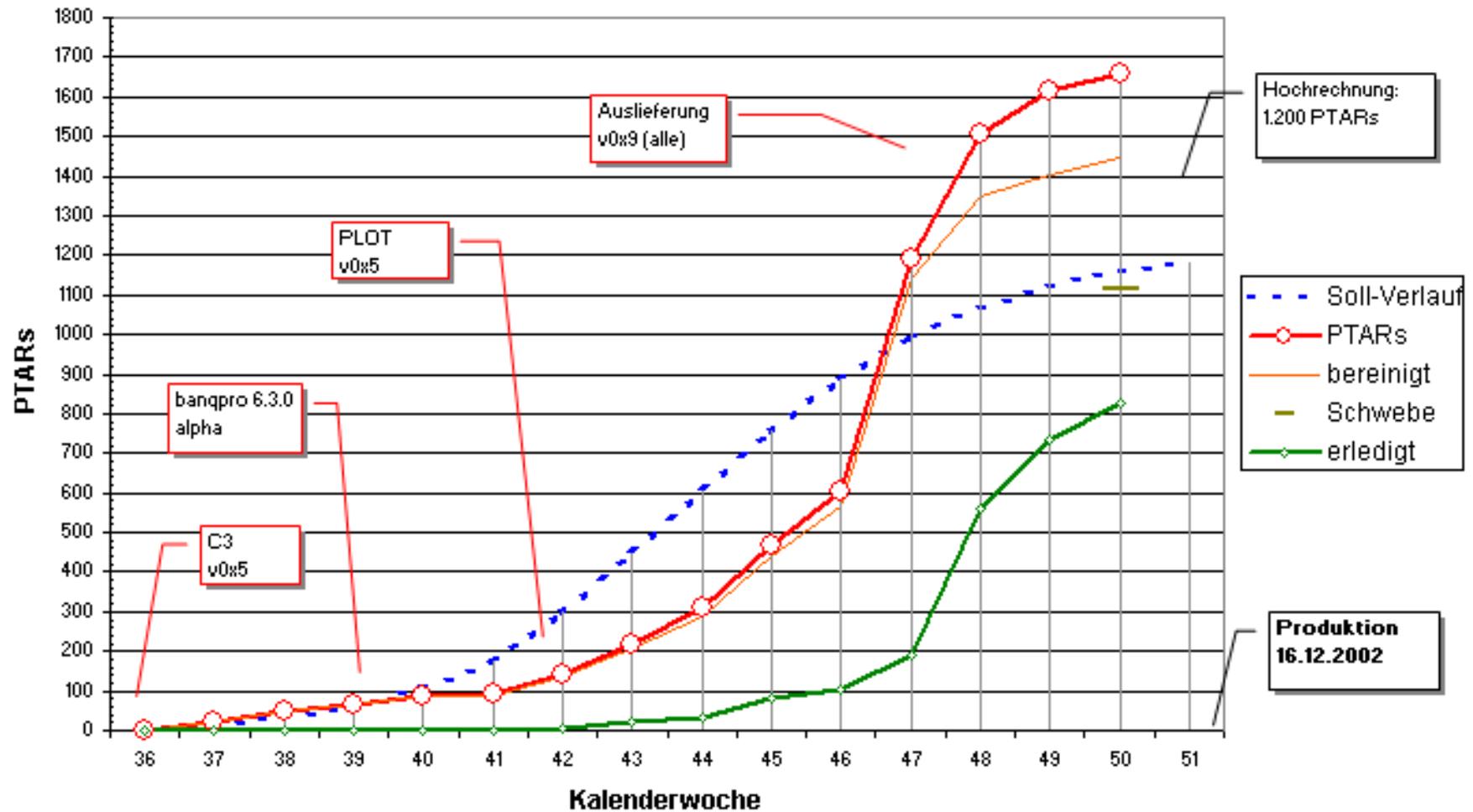
Fehlerhochrechnungen

- Wie kann man Testerfolg messen?
- Hochrechnung der erwarteten Fehler
 - function points
 - lines of code
 - ca. 50 bis 80 Fehler pro kLOC
 - ca. 350 bis 500 LOCs pro PM
 - Development PM
 - ca. 20 bis 40 Fehler pro PM



Fehleranalyse

PTAR-Entwicklung Gesamt



Fehlerverteilung

- Wo werden die Fehler gefunden
 - 42% Analyse & Design
 - 31% Code & Unit Test
 - 23% Funktionstest / Betreibbarkeitstest
 - 4% Produktion



Testphasen

Entwicklungsseitiger Test einzelner Funktionen und Module

Strukturierte Betestung der gewünschten Funktionalität (IST / SOLL Abgleich)

Test der Interaktion einzelner Schnittstellen und Teilkomponenten

Wiederholte Durchführung von Funktionstests zur Grundqualitätssicherung

Test der Gesamtapplikation mit allen Teilkomponenten

Performance-, Last- und Katastrophentests in produktions-äquivalenter Umgebung

Code & Unit Test

Funktions Test

Integrations Test

Regressions Test

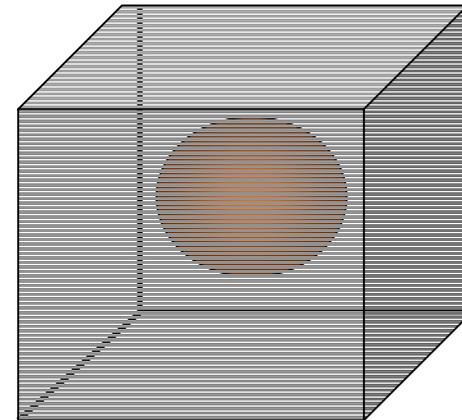
System Test

Betriebbar-keitstest



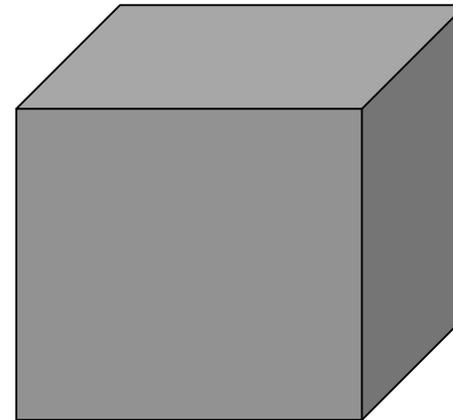
Code & Unit Test

- Wird von Entwicklung durchgeführt
- White Box Test
- Überprüfung von
 - einzelnen Funktionen
 - Schnittstellen
 - einzelnen Modulen



Funktionstest

- Durchführung vom Testteam
- Black Box Test
- Test gegen Spezifikation
- Schnittstelle GUI oder API



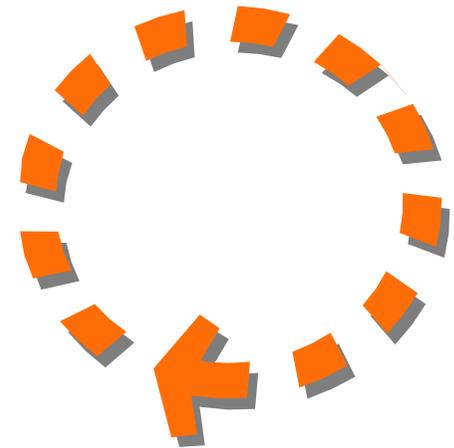
Integrationstest

- Integration mehrerer Module, Komponenten oder Applikationen
- Überprüfung der Schnittstellen
 - Datenstrukturen
 - Interaktion
- Sind zwei Komponenten parallel lauffähig?



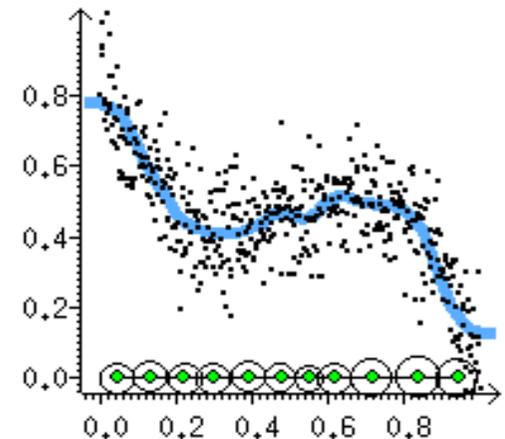
Regressionstest

- Grundqualitätssicherung
- Ausgewählte Testfälle
aus dem Funktionstestfallset
(30% der Testfälle)
- Automatische Durchführung
- Mit jeder neuen Version
- Fehler werden aufgenommen



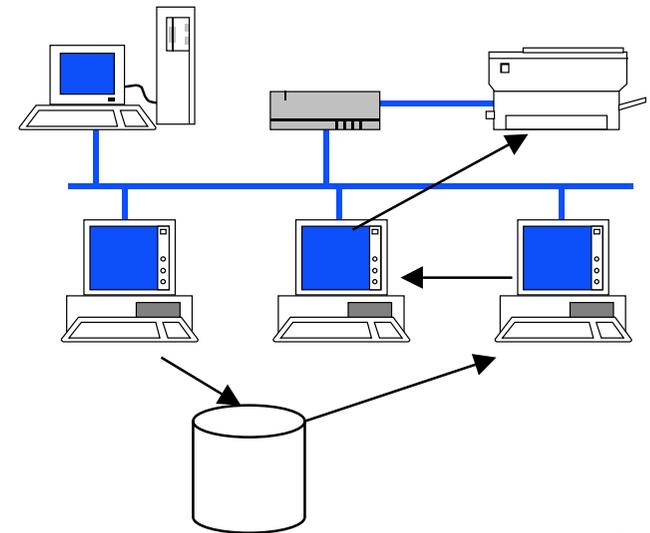
Acceptancetest

- Übergabemodus
- Ausgewählte Testfälle
aus dem Funktionstestfallset
- Die wichtigsten Funktionsbereiche
abgedeckt
- Mit jeder neuen Version
- Entscheidung ob Version testbar



Systemtest

- Test in produktionsäquivalenter Umgebung
- Anbindung aller in der Produktionsumgebung vorhandenen Applikationen
- In der richtigen Version !!
- Betestung von durchgängigen Geschäftsprozessen



Betriebbarkeitstest

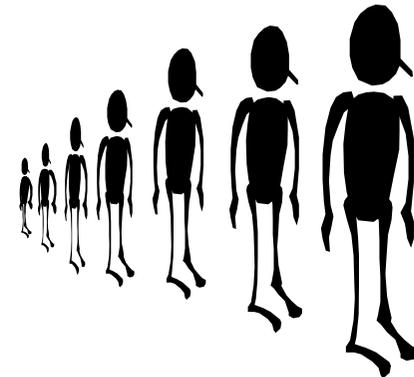
- Installationstest
- Performancetest
- Belastungstest
- Katastrophentest



Manuelles Testen

- Basis: Testfallbeschreibung
- Systematisches Vorgehen
- Guerilla Vorgehen

- Manuelle Durchführung immer notwendig
- Probleme
 - Nachvollziehbarkeit
 - Kosten



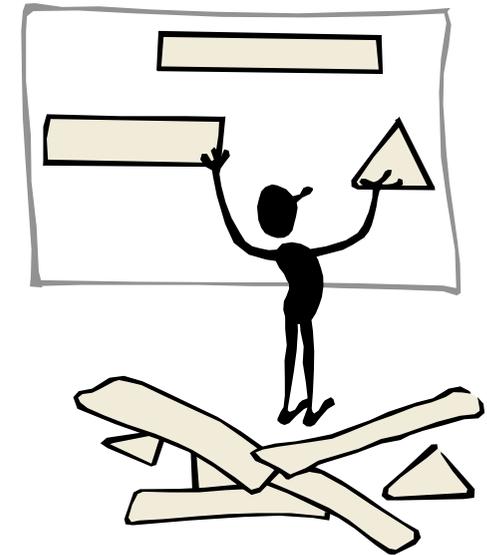
Testfallfindung

- Was ist ein Testfall
 - Beschreibung einer Applikationsbedienung mit einem erwarteten Ergebnis.
- Wie findet man ‚gute‘ Testfälle
 - Chance einen Fehler zu finden
 - Realistisch (risk-based)
 - Breite Abdeckung
- Woraus besteht ein Testfall



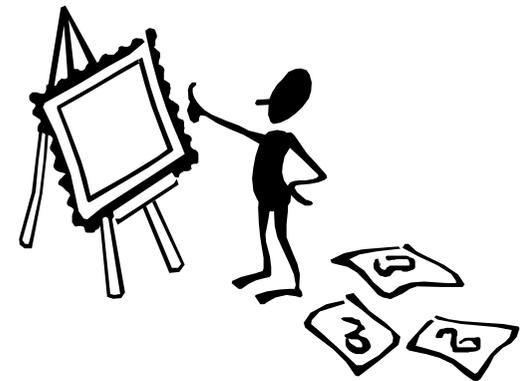
Testfallfindung

- Erstellung von Funktionsdiagrammen
 - Geschäftsprozesse, Usecases, Funktionen
 - Masken, Gruppen, Controls
- Bewertung der Kritikalität
- Skalierung der Testtiefe
- Planung der Testbereiche
- Definition der Testfälle

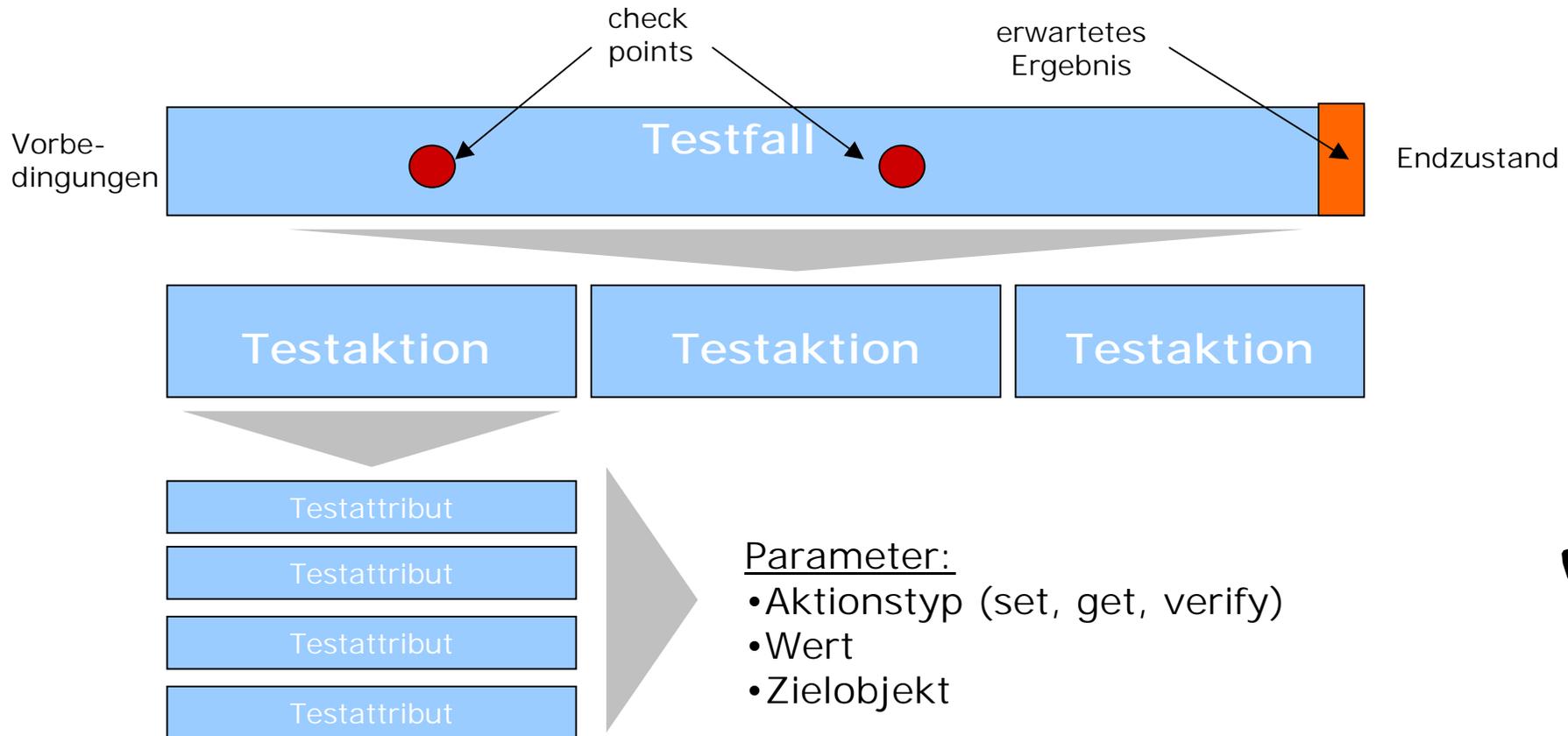


Inhalt eines Testfalles

- Detaillierte Beschreibung
- Zweck des Testfalls (Testfokus)
- Vorbedingungen
- Aufräumarbeiten
- Zugrunde liegende Testdaten
- Beschreibung der einzelnen Schritte
- Erwartetes Ergebnis
- Tatsächliches Ergebnis pro Durchführung

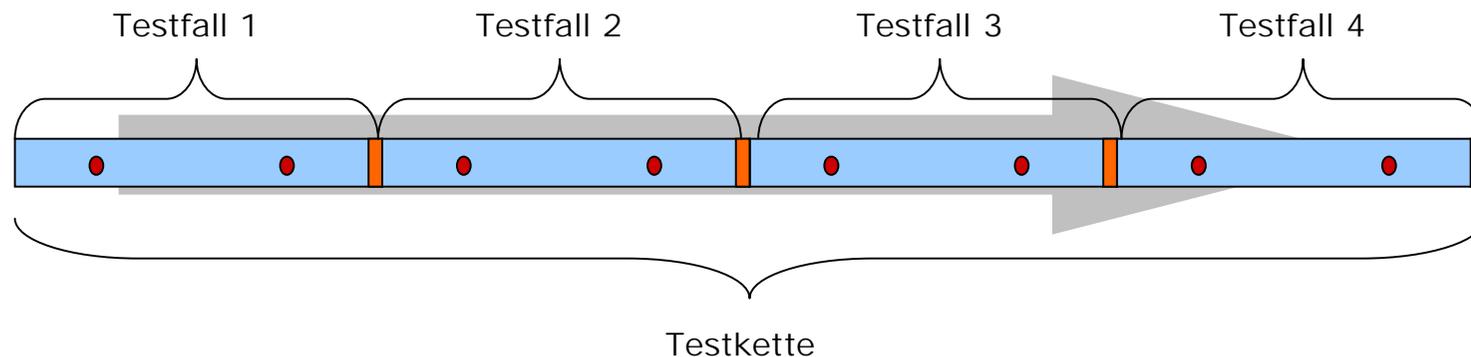


Struktur eines Testfalls



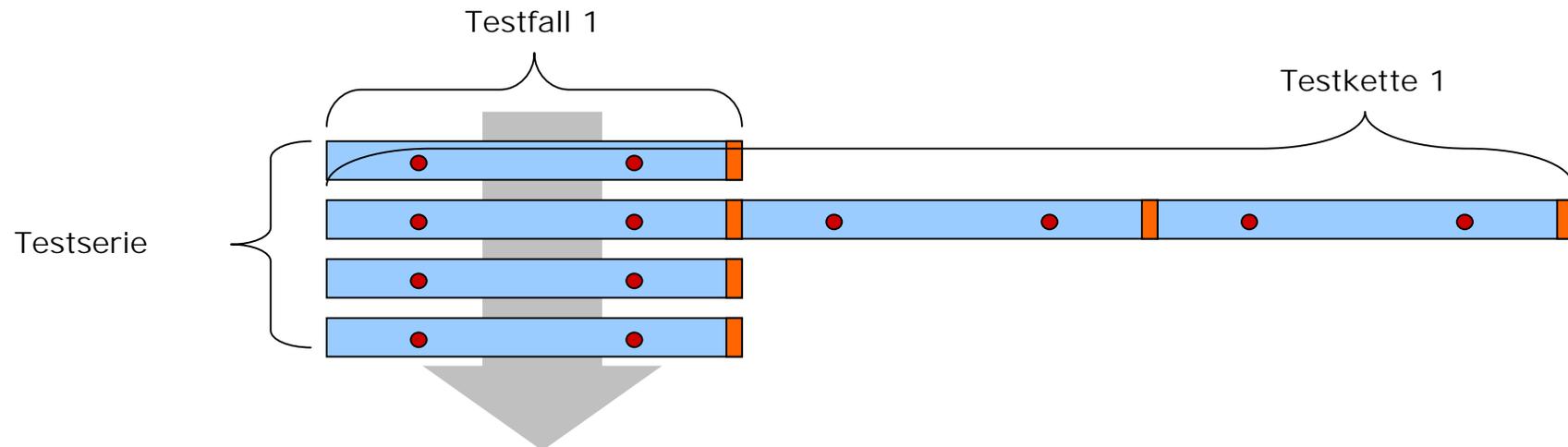
Testfälle

- Testkette
 - Aneinanderreihung von Testfällen
 - Funktionale Verbindung
 - Voneinander abhängig



Testfälle

- Testserien
 - Aneinanderreihung von Testketten und/oder Testfällen
 - Fachliche / Thematische Verbindung
 - Voneinander unabhängig



Testautomation

- Wozu Testautomation
 - Ermöglicht breite Abdeckung
 - Vorhandene Testfälle können in jeder neuen Version wiederholt werden
 - Kein manueller Eingriff notwendig
 - Manche Tests können manuell schwer oder überhaupt nicht durchgeführt werden



Testautomation

- Vorteile der Testautomation
 - Wiederholbarkeit
 - Durchführung in konstanter Geschwindigkeit
 - Wiederverwendbarkeit
 - Mehr und umfangreichere Testläufe
 - Besserer Einsatz von Ressourcen (staff morale)
 - Reduktion der Testzeiten



Testautomation

- Grenzen der Testautomation
 - Manuelle Tests werden dadurch nicht unnötig
 - Die erste Durchführung eines automatisierten Testfalls dauert deutlich länger



Testautomation

- Häufige Fehler / Missverständnisse
 - Unrealistische Erwartungen
 - Testautomation löst nicht alle Probleme
 - Wenig Testerfahrung
 - Unzureichende Testfall Dokumentationen
 - Ineffiziente Testfälle
 - Erwartung, viele neue Fehler zu finden

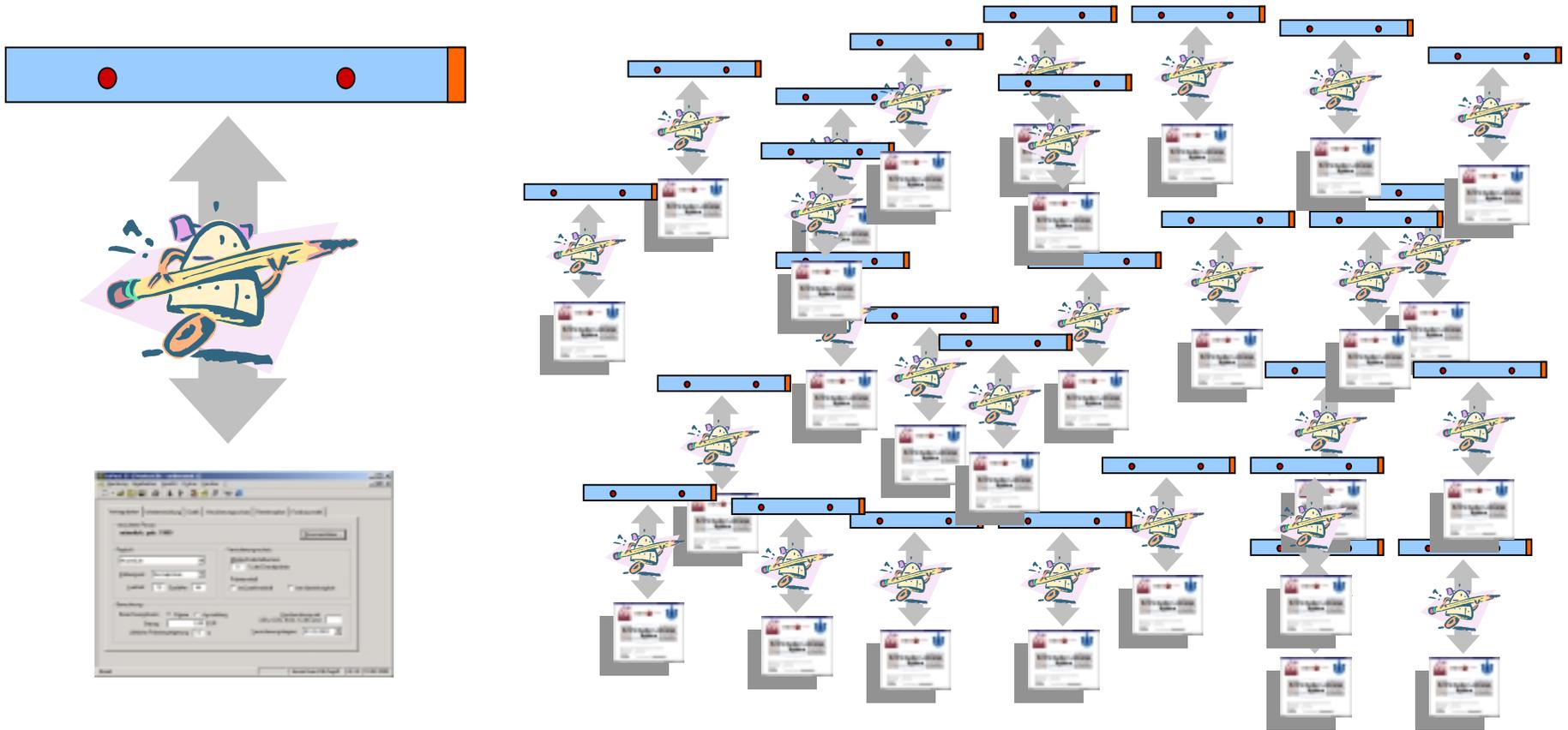


Testautomation

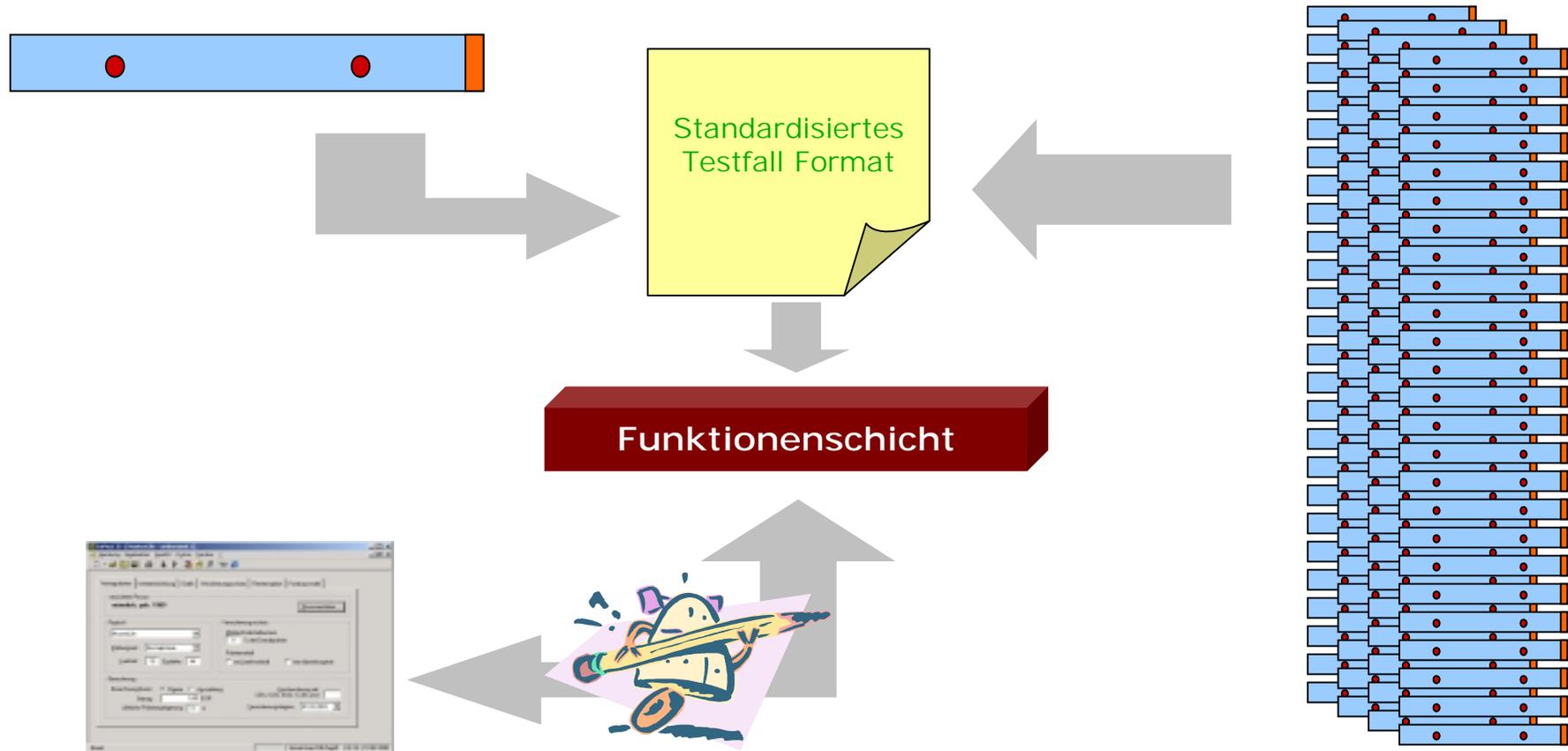
- Häufige Probleme
 - Wartung der automatisierten Testfälle
 - Technische Probleme - fehlerhafte Testtools
 - Organisatorische Dinge
 - Testautomation ist eine mittel- bis langfristige Investition



Testautomation – capture & replay



Testautomation – generischer Ansatz



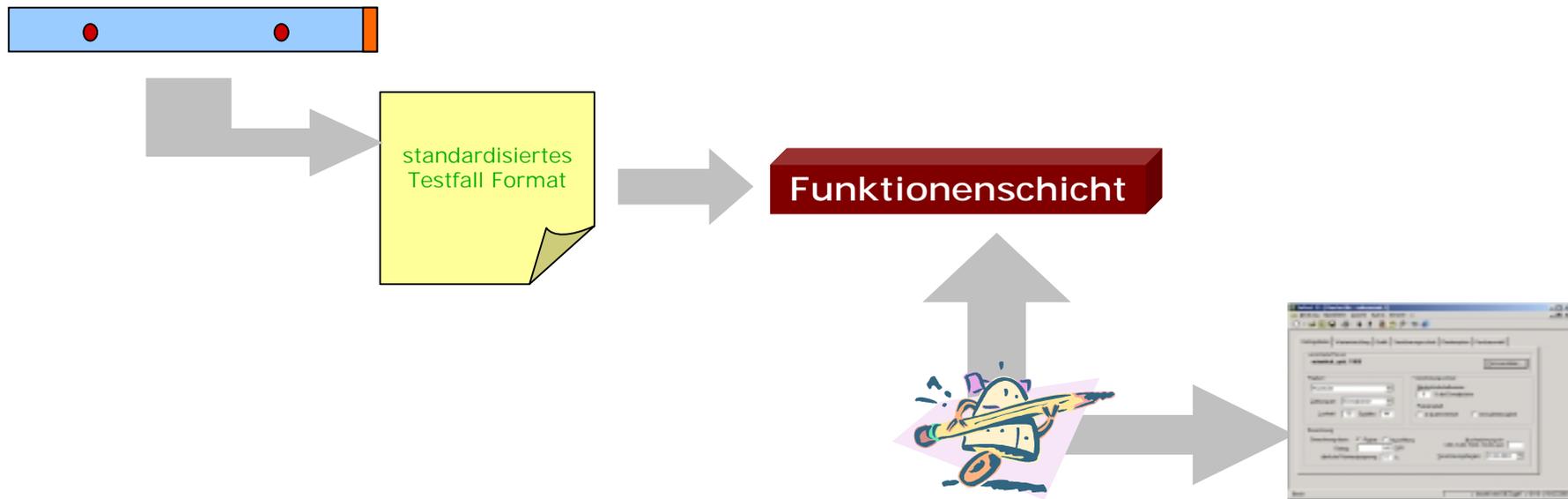
Testautomation

- Unser Konzept der Testautomation
 - NICHT capture & replay
 - Zu hoher Wartungsaufwand
 - Geringe Effizienz bei hoher Anzahl von Testfällen
 - Kein error handling möglich
 - Keine unbeaufsichtigten Testläufe möglich
 - Generischer Ansatz
 - Unterschiedliche Testfälle verwenden die selben Funktionen und Methoden
 - Hoher Grad an Wiederverwendbarkeit
 - Geringer Wartungsaufwand, effizienter ...

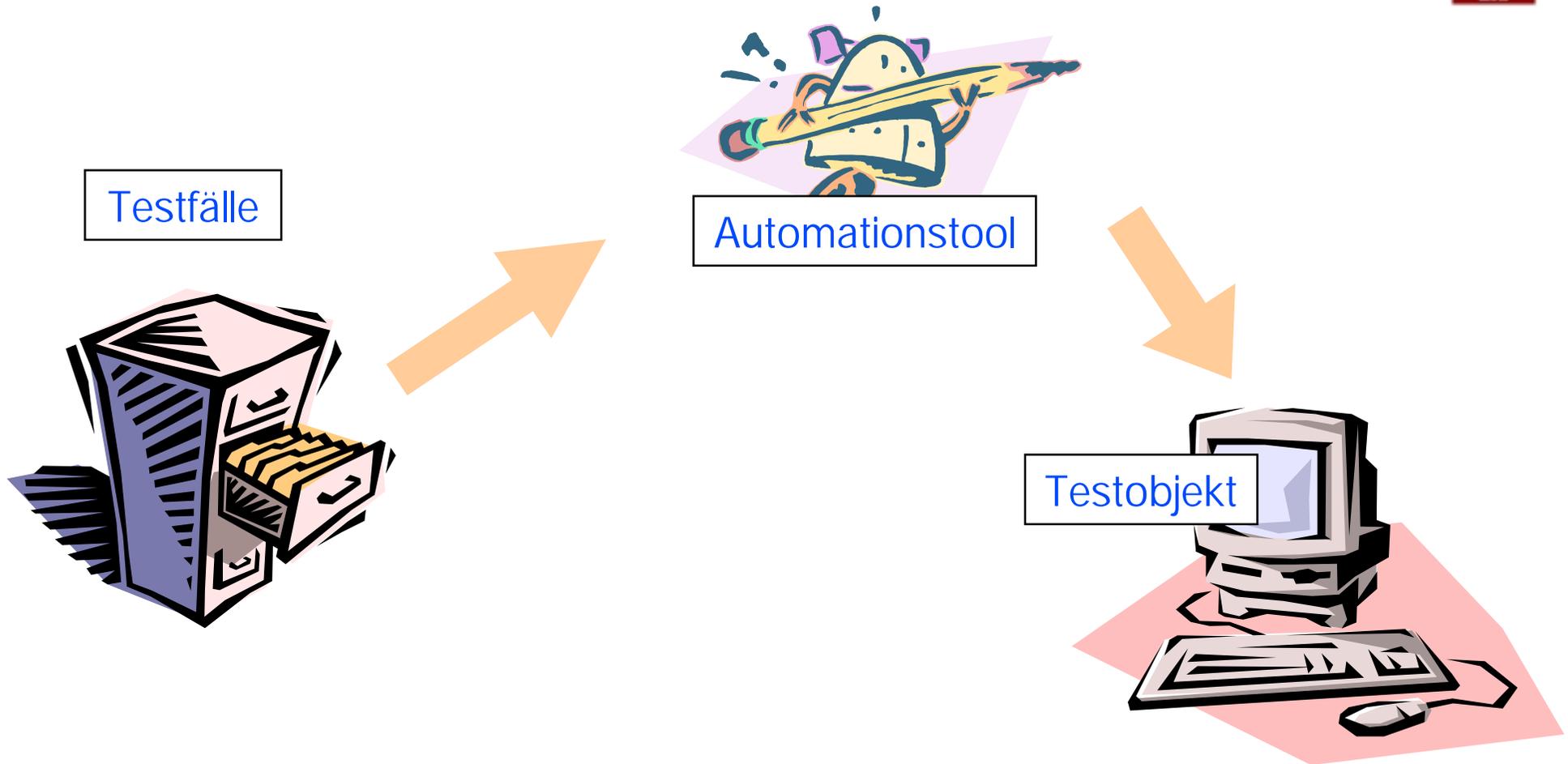


Testtools

- Der Grundsatz unserer Testtool-Sammlung
 - strikte Trennung von Testfallerstellung und Testdurchführung

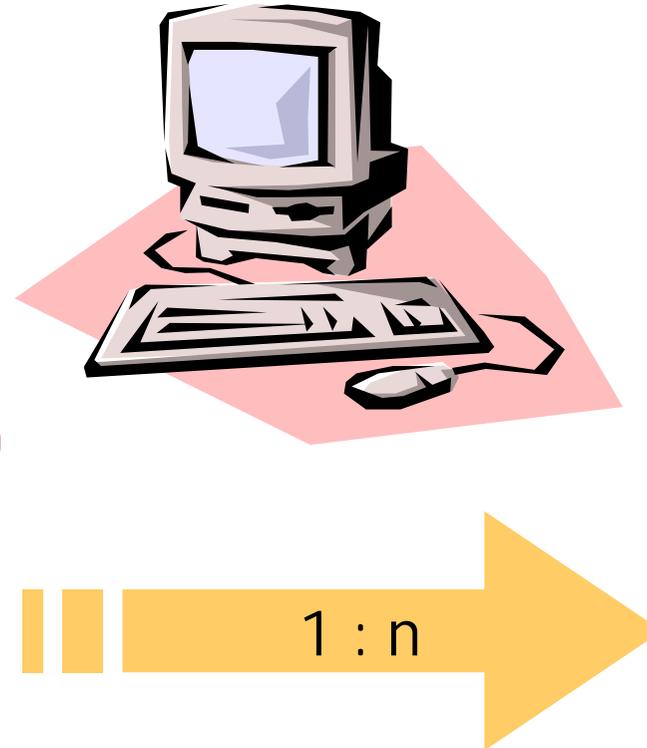


Testautomation

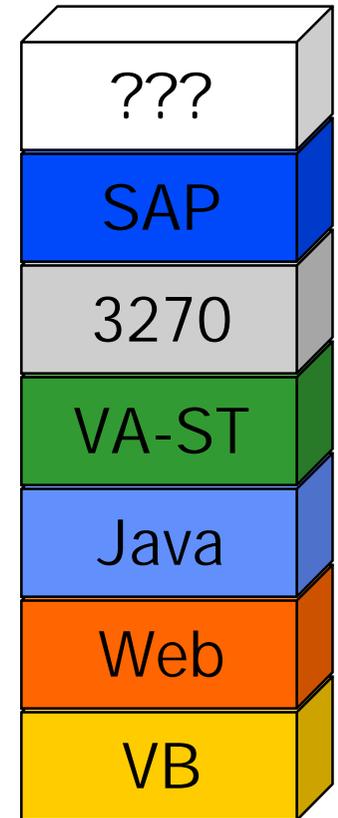


Testlandschaft

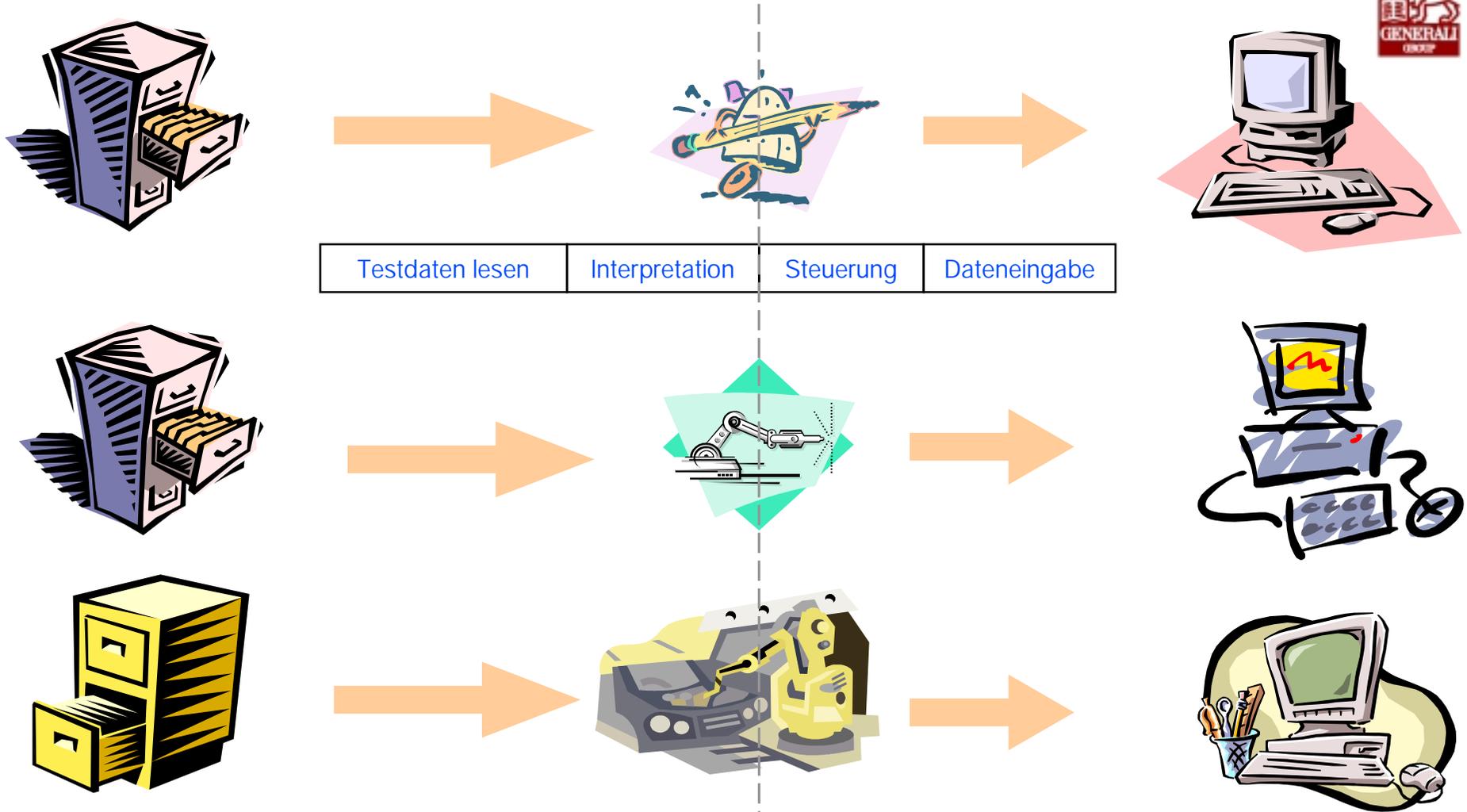
Projekte



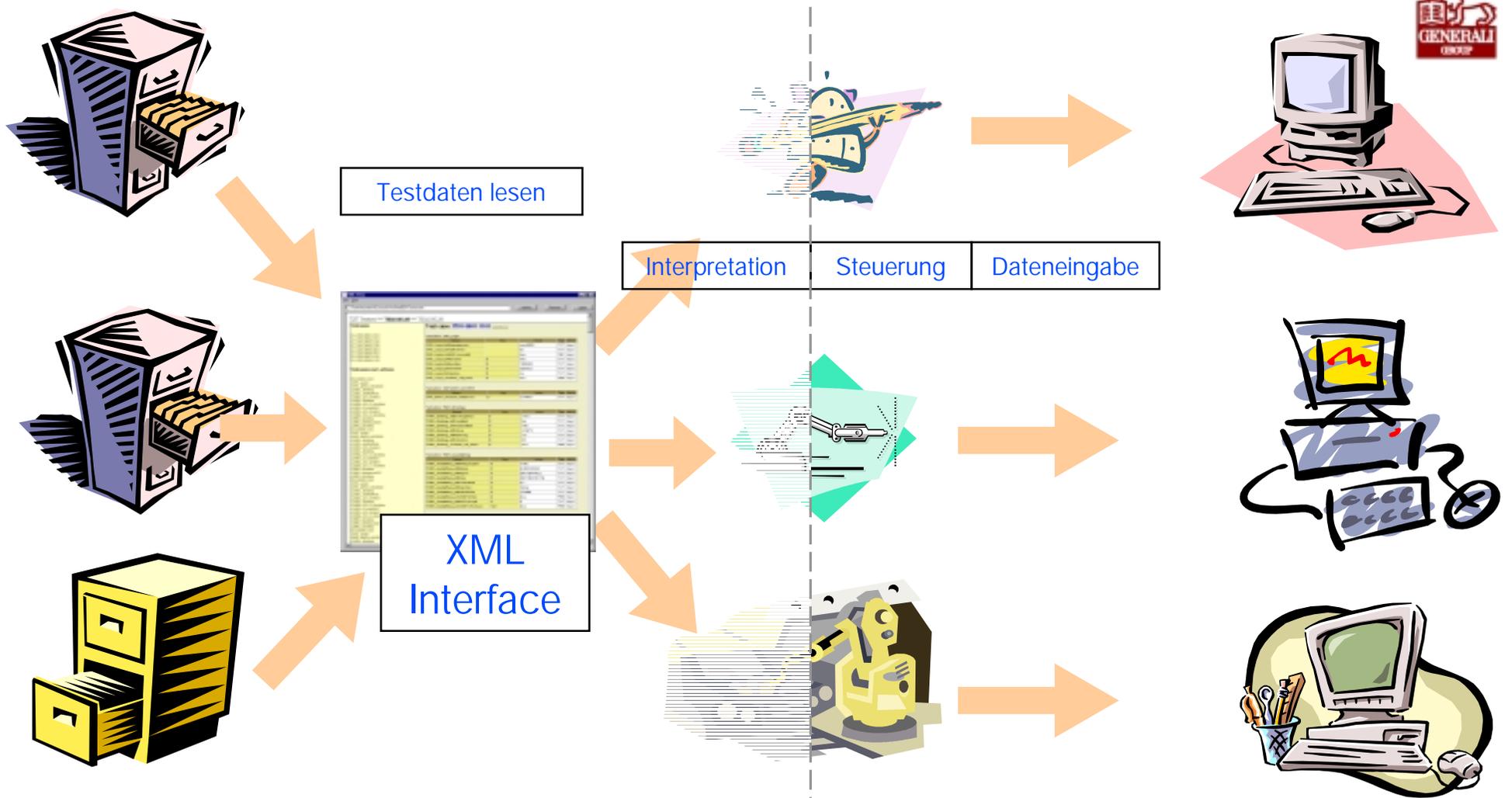
Plattformen



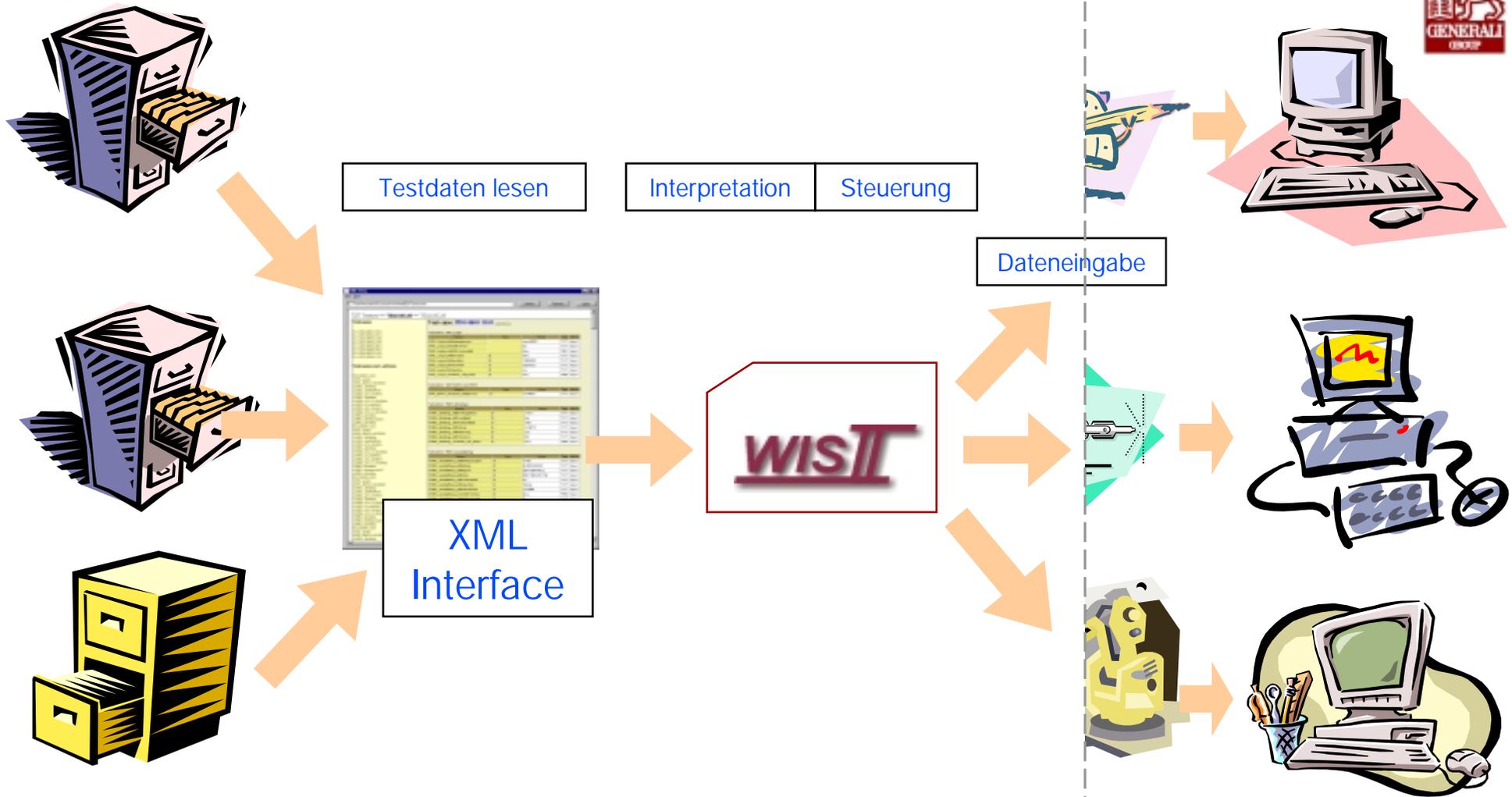
Testautomation



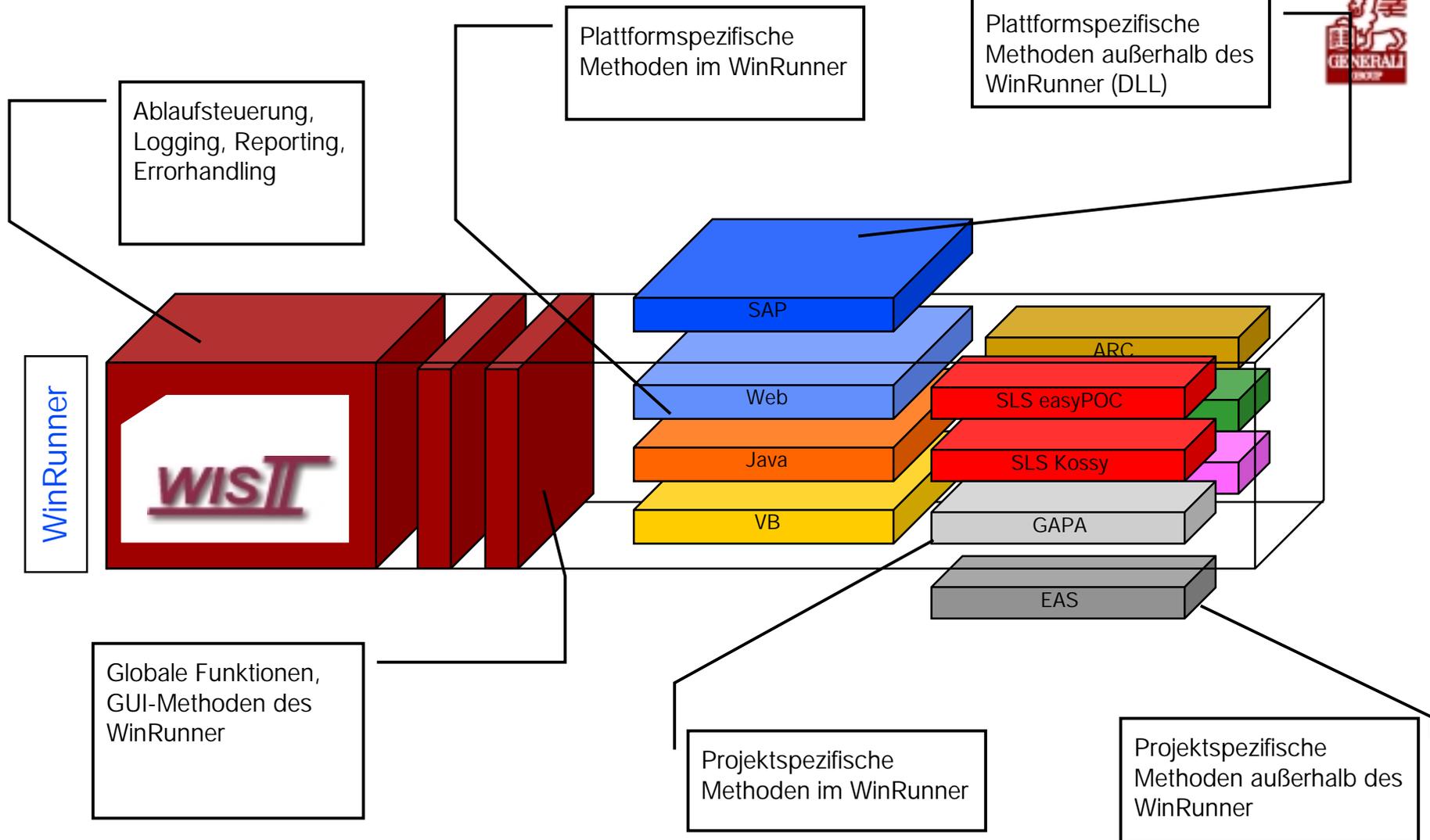
XML Interface



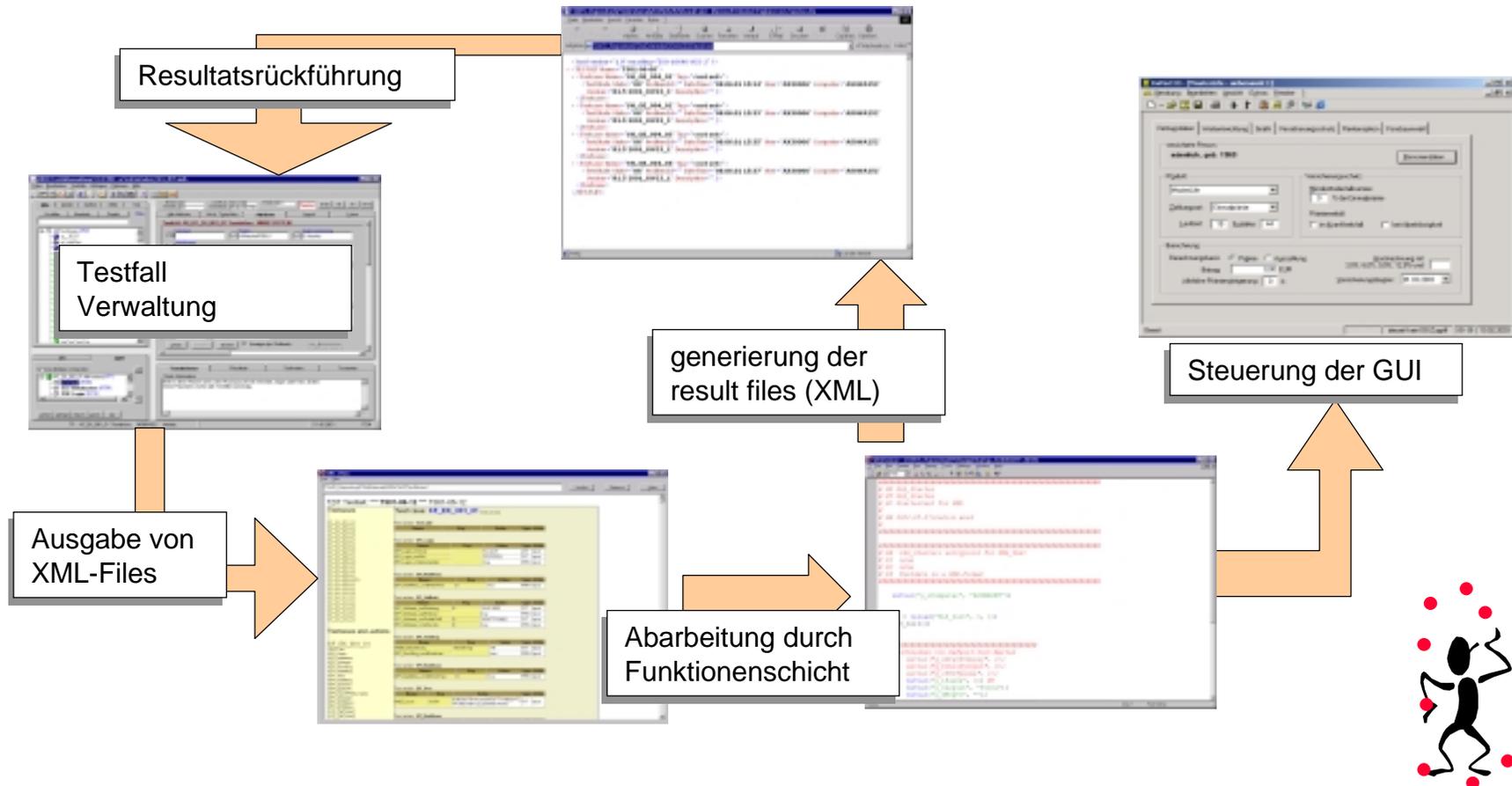
Funktionenschicht



Aufbau der Funktionenschicht



Testautomation Kreislauf



Fragen?

Danke für Ihre Aufmerksamkeit

