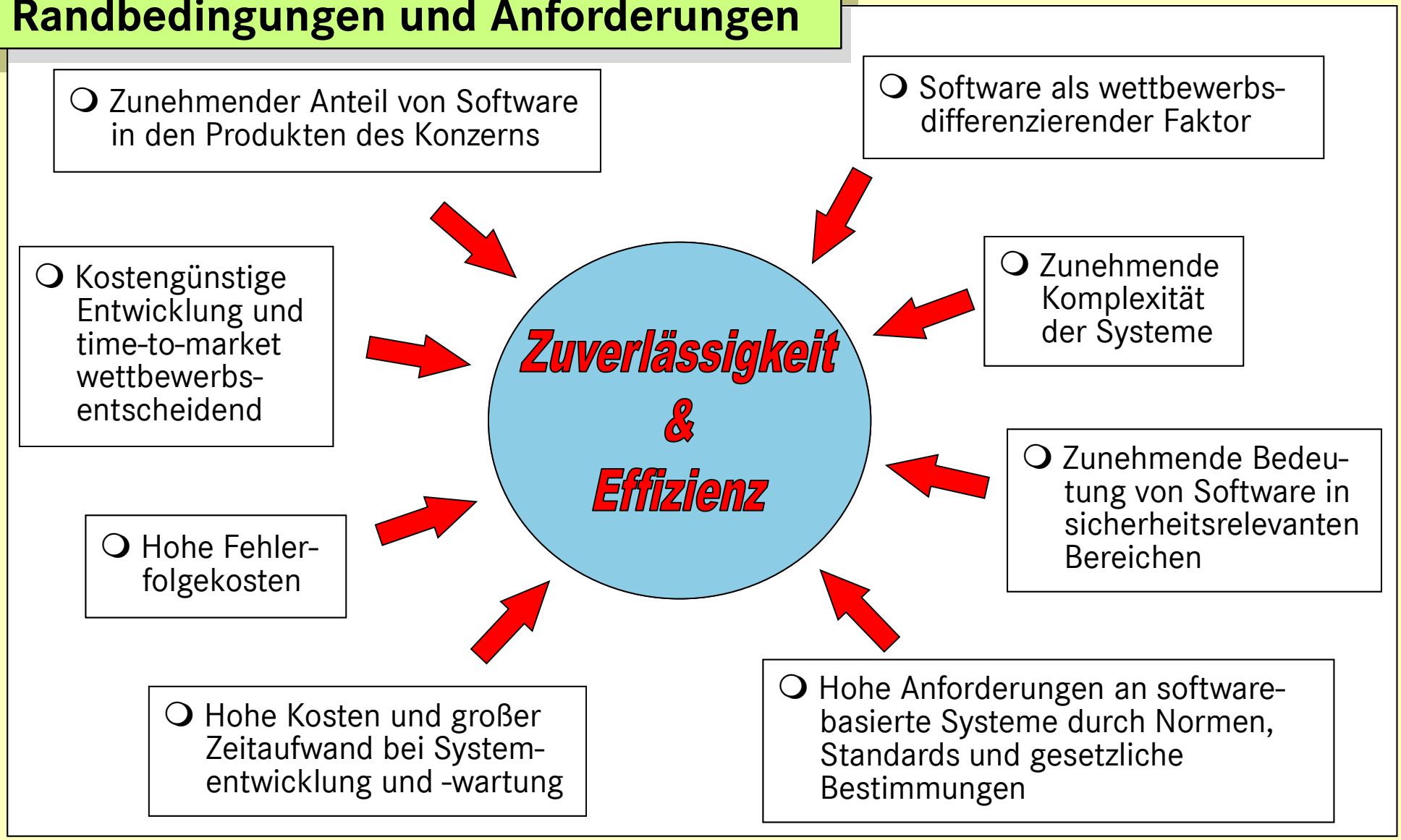
imbus QS-Tag, 8. November 2001

# Evolutionärer Test von Embedded Systems

Joachim Wegener
DaimlerChrysler AG, Forschung und Technologie
Joachim.Wegener@daimlerchrysler.com

- Einleitung, Motivation
- Evolutionärer Test
  - Test des Zeitverhaltens
  - Strukturtests
  - Sicherheitstests
  - Weitere Anwendungsfelder
- Zusammenfassung

## Randbedingungen und Anforderungen



## Testziele

Ziele des Tests sind

- das Aufdecken von Fehlern im Testobjekt und
- das Schaffen von Vertrauen in die korrekte Funktionsweise des Testobjekts

durch die Ausführung des Testobjekts mit Testdaten.

## Stärken des Tests

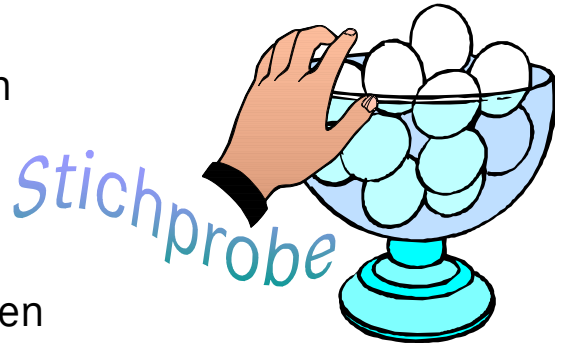
- ⊕ Berücksichtigung der realen Einsatzumgebung (z.B. Zielsystem, Compiler, ...)
- ⊕ Prüfung des dynamischen Systemverhaltens (z.B. Laufzeitverhalten, Speicherplatzbedarf, ...)

## Schwäche des Tests

- ⊖ ein vollständiger Test ist nur für sehr einfache Systeme möglich

**wichtigste Testaktivität (Testfallermittlung)**

Testdaten müssen nach geeigneten Kriterien ausgewählt werden



## Test von Embedded Systems

Test von Embedded Systems komplexer als für herkömmliche Systeme

Zum einen wird der Test durch eine Reihe technischer Systemeigenschaften erschwert, z.B.

- enge Vernetzung mit der Einsatzumgebung,
- Entwicklung in Host-Target-Umgebungen (+ Stückzahlen, - Speicher, - Prozessorreserven),
- Einsatz von Parallelität und Verteiltheit und
- Verwendung von Fehlertoleranzmechanismen,

zum anderen müssen häufig nicht-funktionale Systemaspekte berücksichtigt werden, z.B.

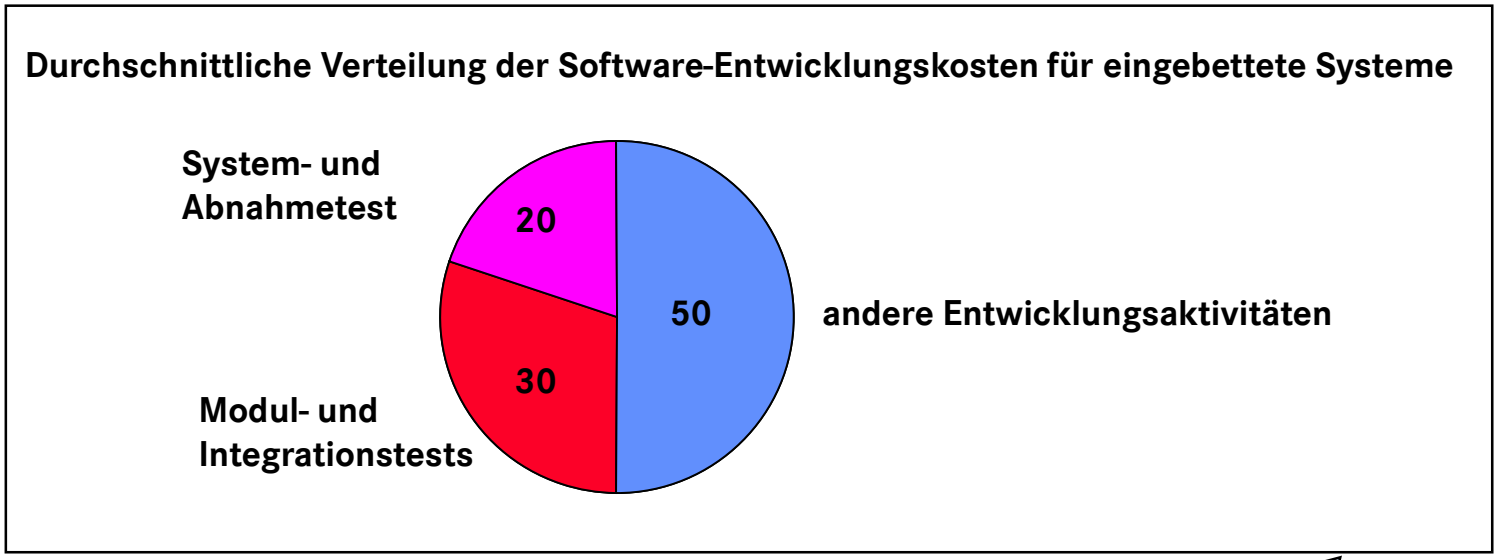
- zeitliche Anforderungen, die sich aus Gründen des Benutzungskomforts oder aus Erfordernissen des angeschlossenen technischen Prozesses ergeben, oder
- Sicherheitsanforderungen für sicherheitsrelevante Systeme.

⇨ Vielzahl anspruchsvoller Testaufgaben, für die methodische Unterstützung und möglichst umfangreiche Automatisierung notwendig sind.

⇨ Neben Funktions- und Strukturtests zur Prüfung des logischen Systemverhaltens sind Tests zur Prüfung der zeitlichen Systemaspekte und Sicherheitstests durchzuführen.

## Test von Embedded Systems

- Tests sind in der Praxis das wichtigste analytische Qualitätssicherungsverfahren
- erheblicher Kostenfaktor für die Systementwicklung




- Tests sind zu aufwendig
  - ⇨ hohe Kosten
- Tests werden zu unsystematisch durchgeführt
  - ⇨ geringe Fehleraufdeckungsrate

Testautomatisierung

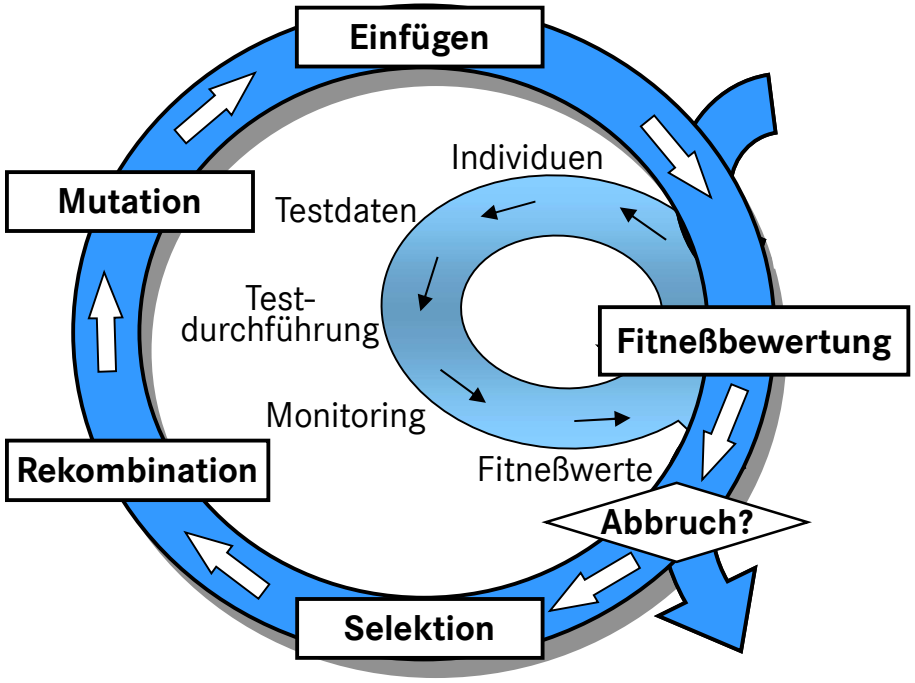
Testmethoden

# Evolutionärer Test



## Funktionsweise

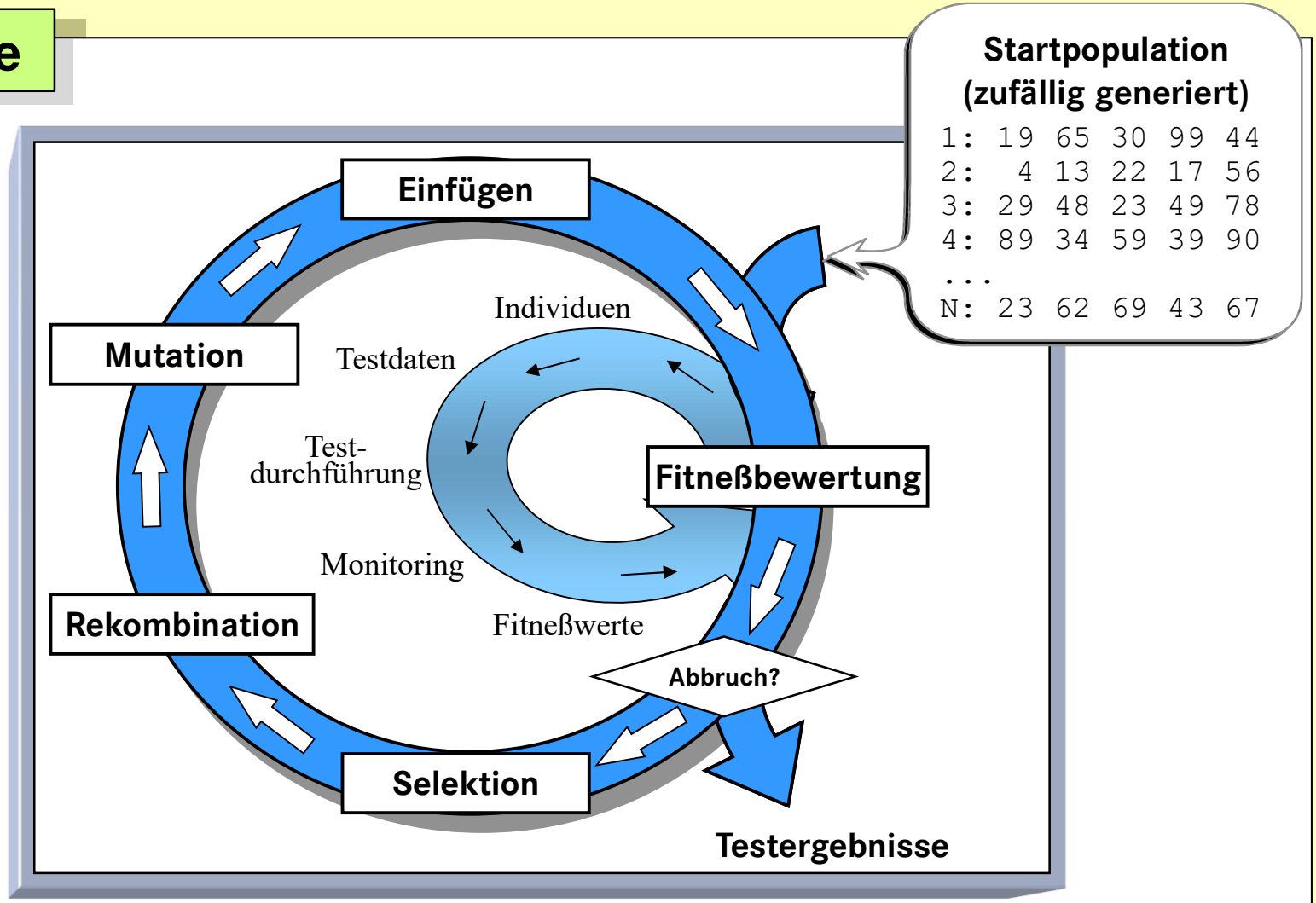
- Neuartiges Testverfahren für die systematische und vollautomatische Generierung von Testfällen und Testdaten durch leistungsfähige Suchverfahren, z.B. evolutionäre Algorithmen.
- Testziel wird numerisch formuliert und in ein Optimierungsproblem übersetzt (Fitneßfunktion).
- Der Eingabedatenraum des zu prüfenden Systems bildet den Suchraum.
- Die Eingangsparameter des zu prüfenden Systems bilden die Entscheidungsvariablen für die Optimierung.
- Die generierten Eingabesituationen werden anhand einer Ablaufüberwachung (Monitoring) beurteilt. Hinsichtlich der Fehleraufdeckung/Code-Überdeckung vielversprechende Eingabesituationen werden dann miteinander kombiniert.
- Iterativer Prozeß



# Evolutionärer Test



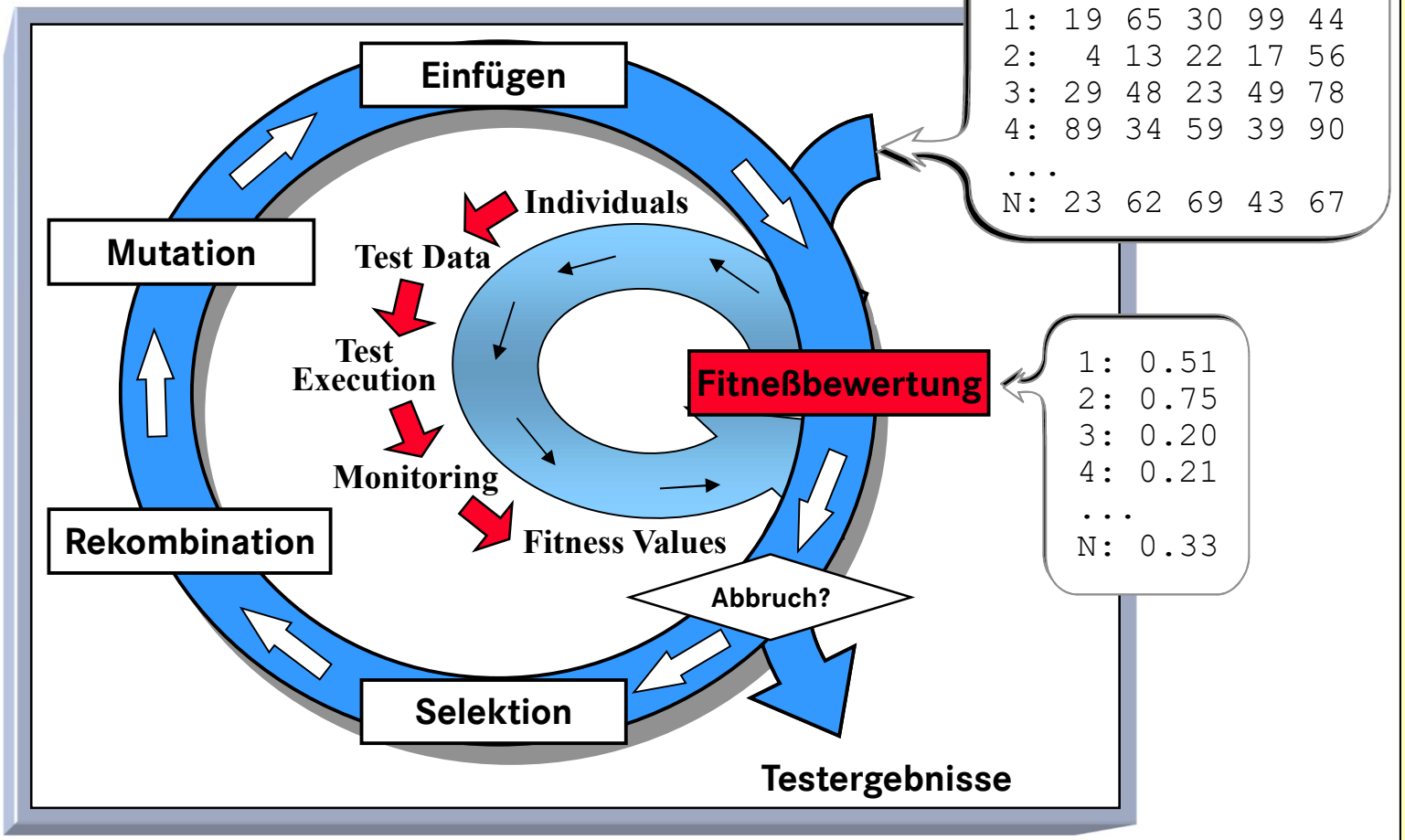
## Funktionsweise



# Evolutionärer Test



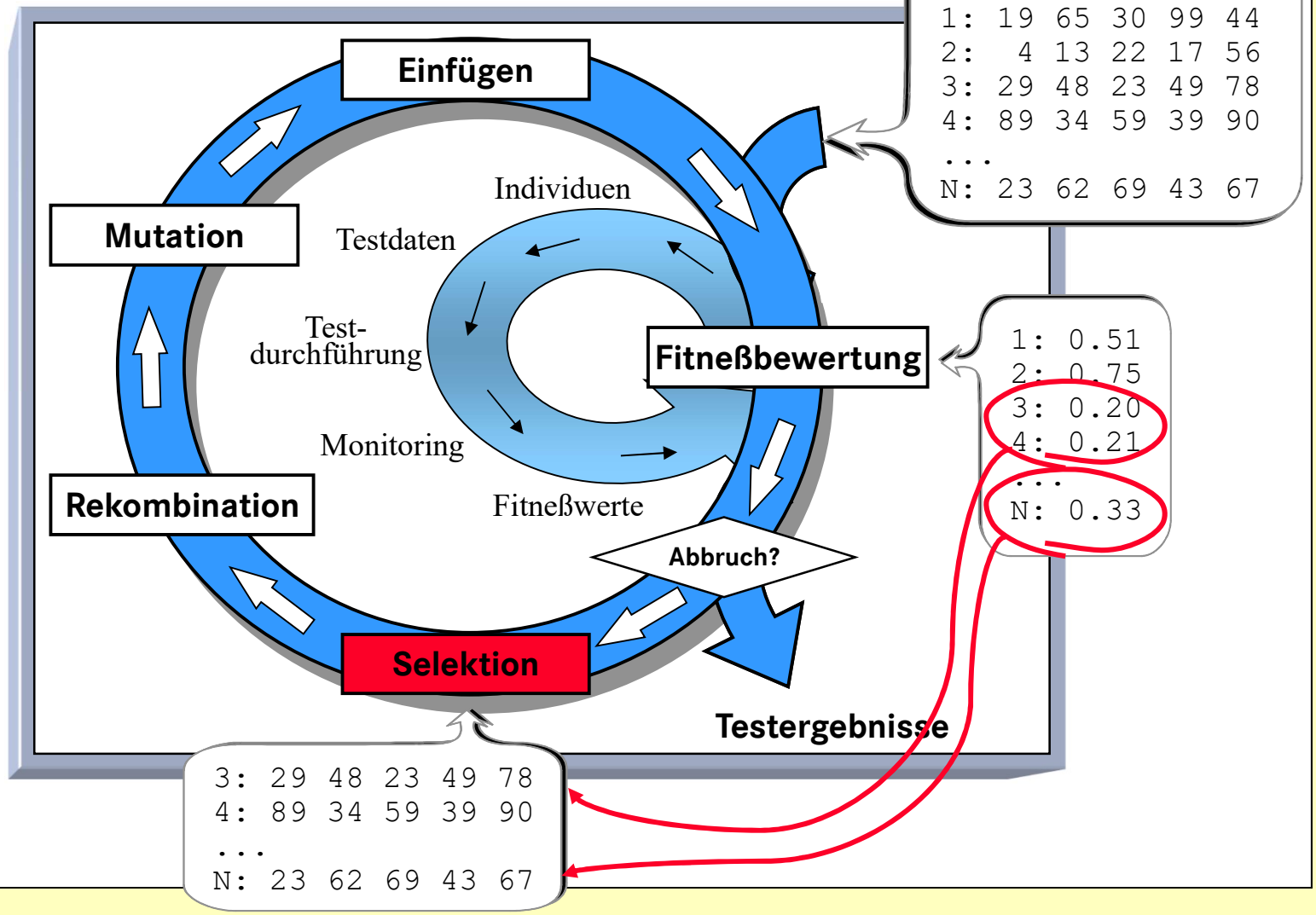
## Funktionsweise



# Evolutionärer Test



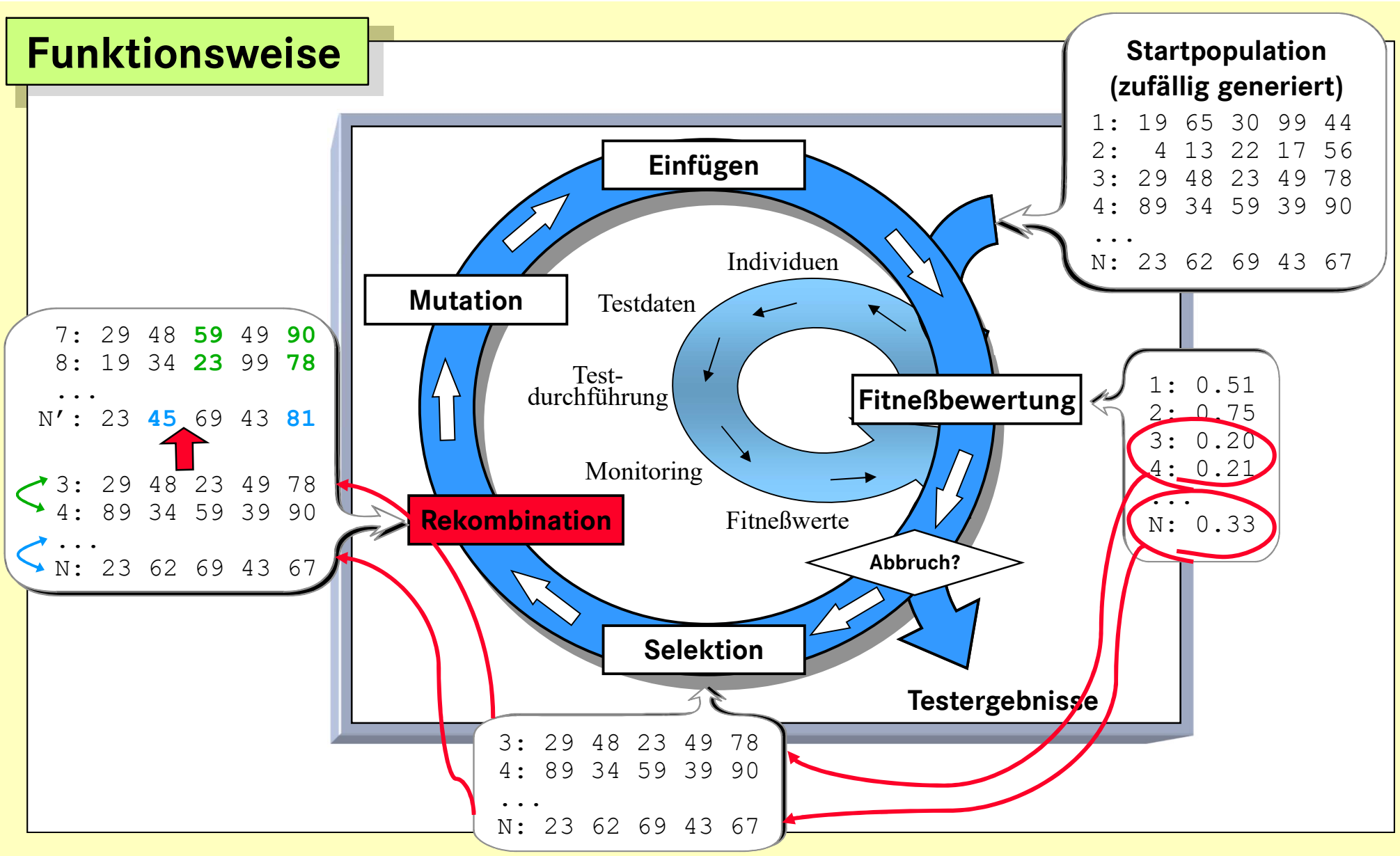
## Funktionsweise



# Evolutionärer Test

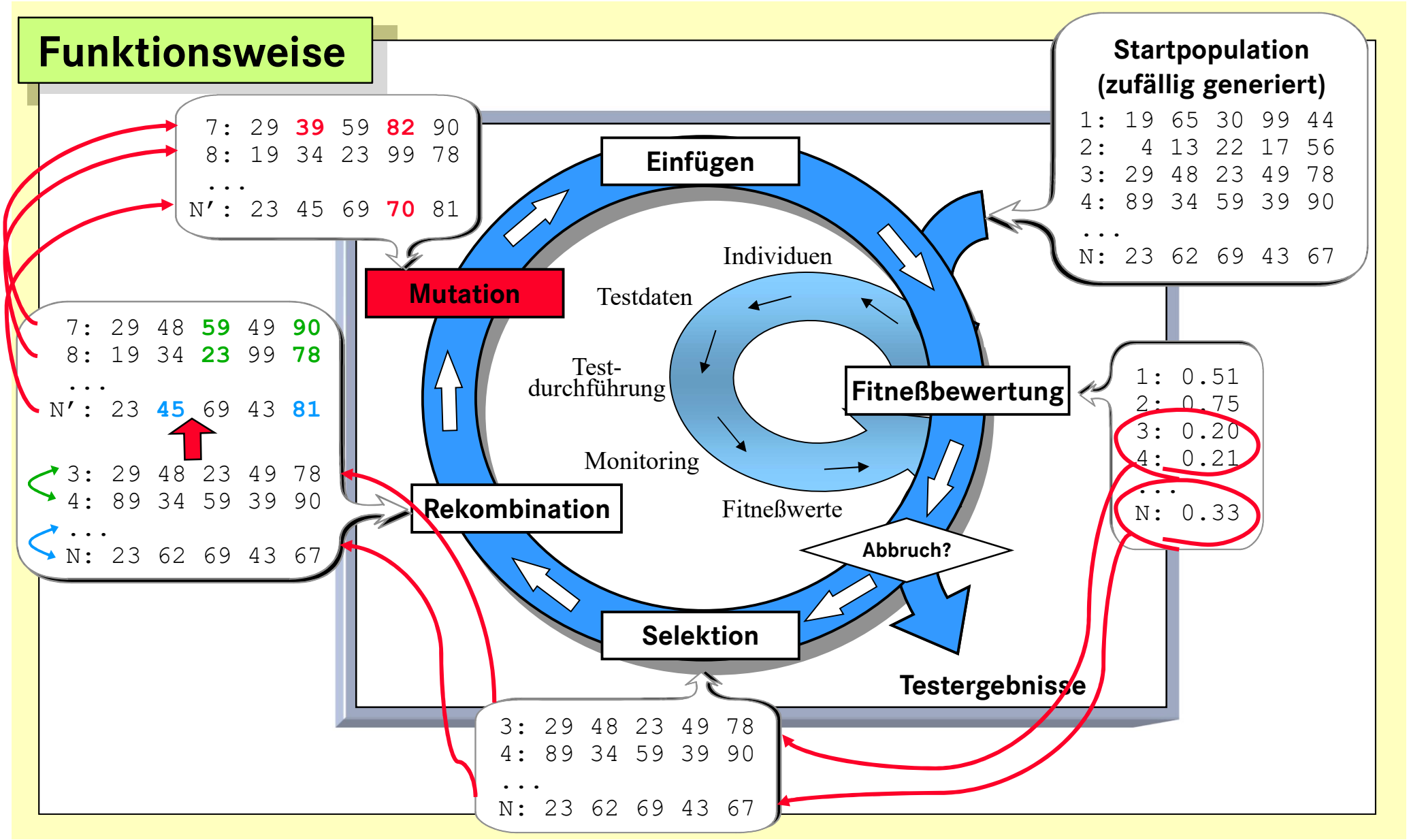


## Funktionsweise



# Evolutionärer Test

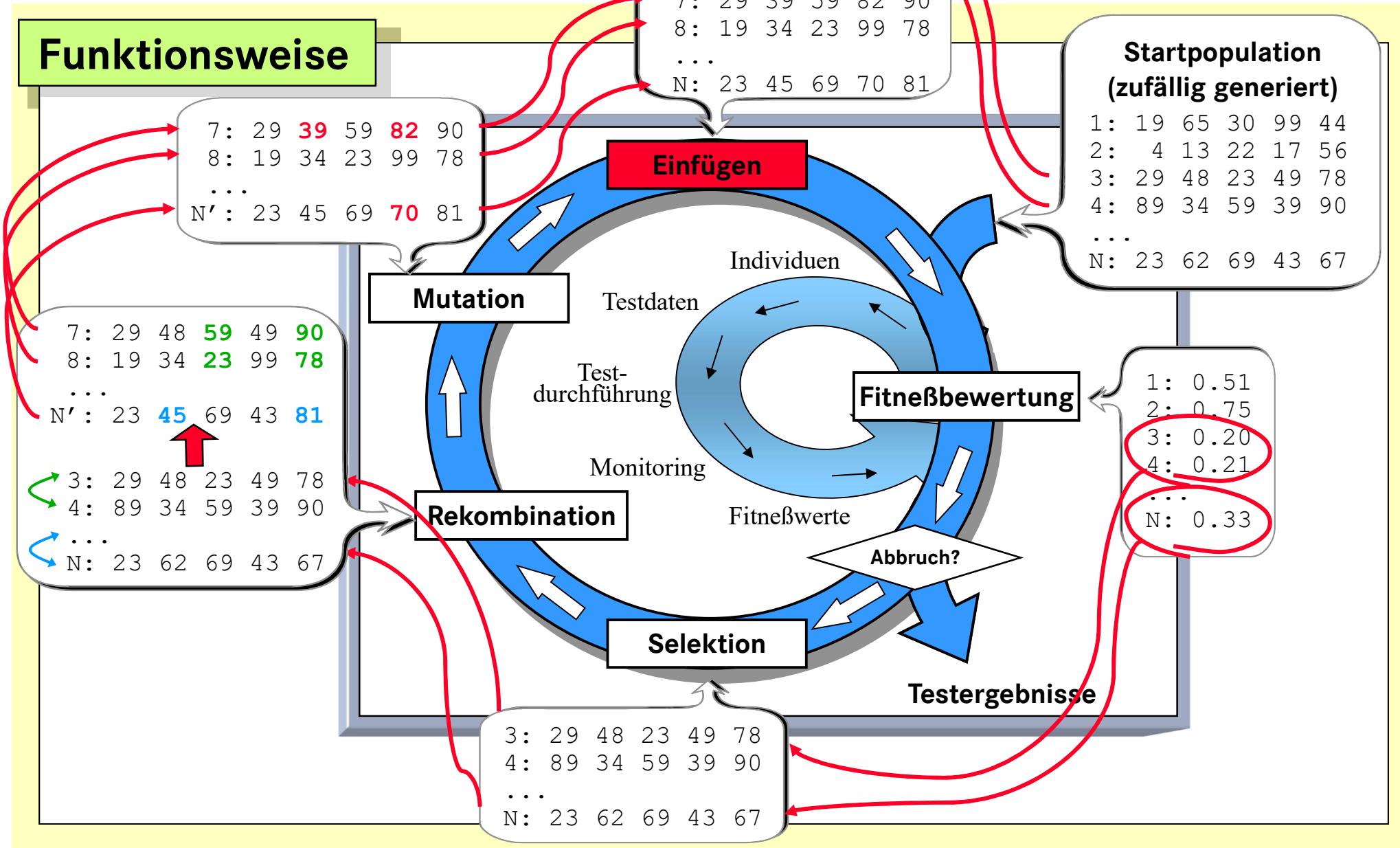




# Evolutionärer Test



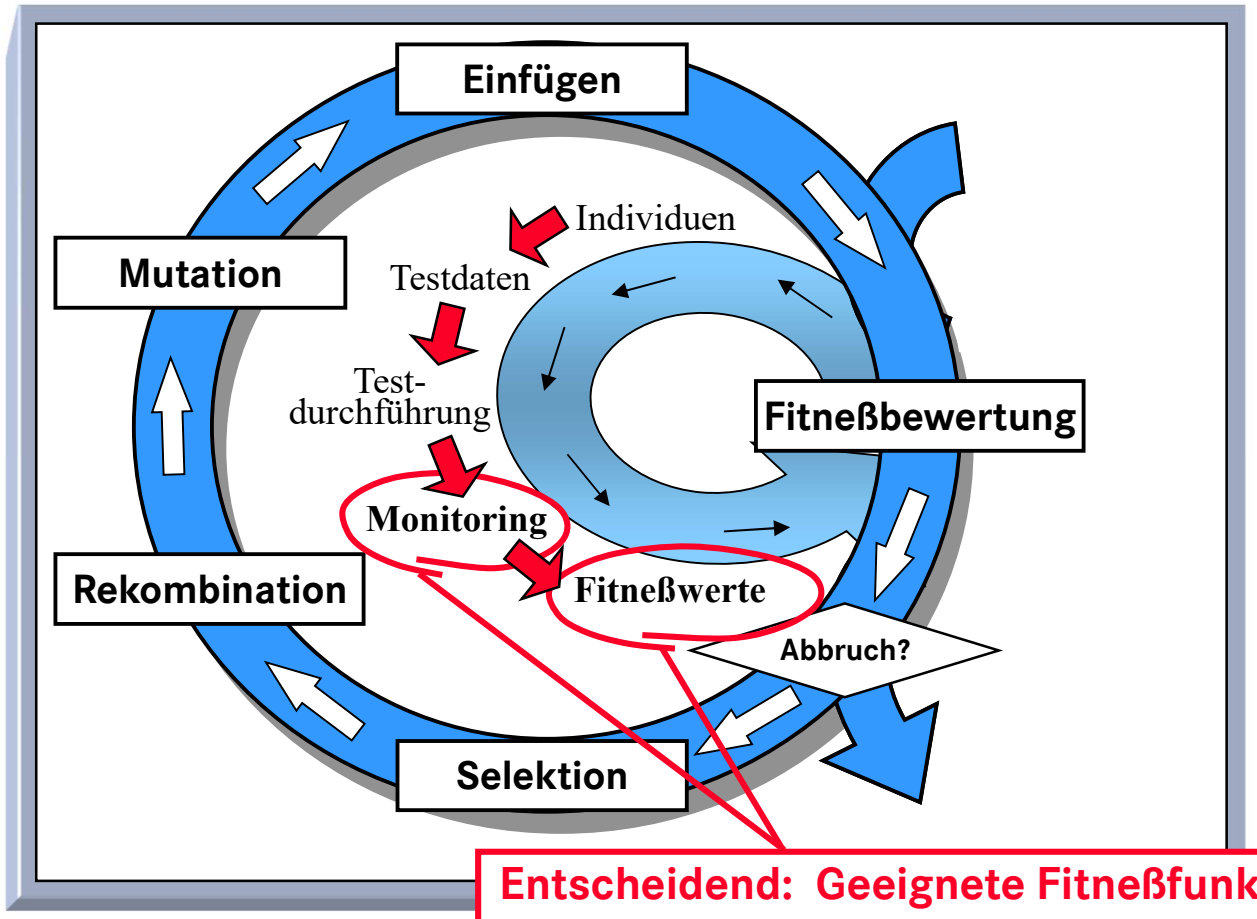
## Funktionsweise



# Evolutionärer Test



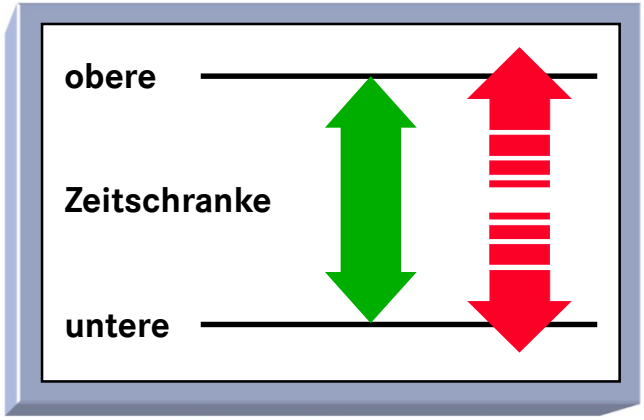
## Anwendung




**Entscheidend: Geeignete Fitneßfunktion für jeweiliges Testziel**

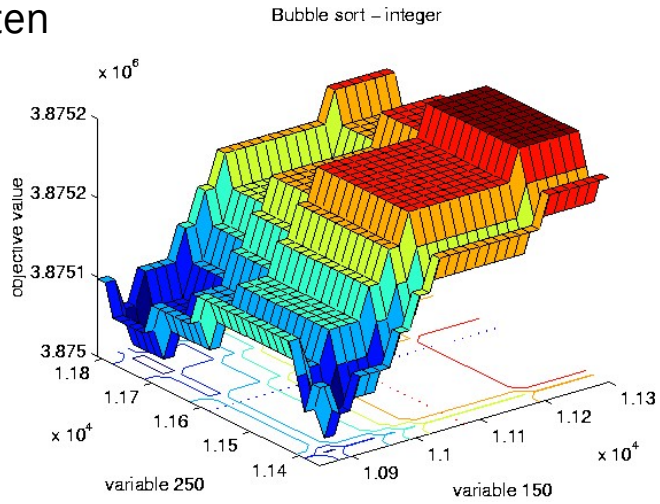
## Test des Zeitverhaltens

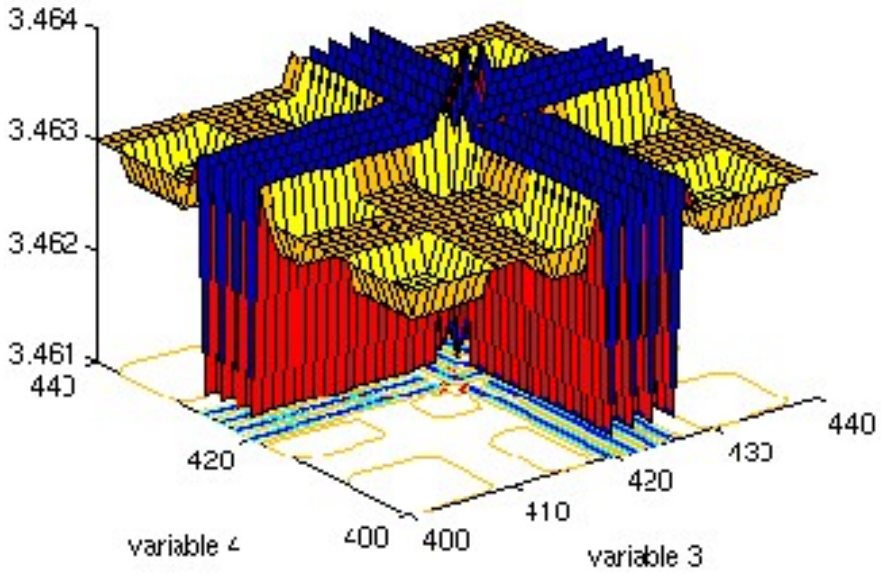
**Ziel** Aufdecken von Eingabesituationen, die zu einem Fehler im zeitlichen Verhalten der Systeme führen (Über- bzw. Unterschreiten spezifizierter Zeitschranken)




**Testidee**

- Suche nach Testdaten mit besonders langen und besonders kurzen Ausführungszeiten
- Fitneßbewertung der Individuen basiert auf den gemessenen Ausführungszeiten der zugehörigen Testdaten



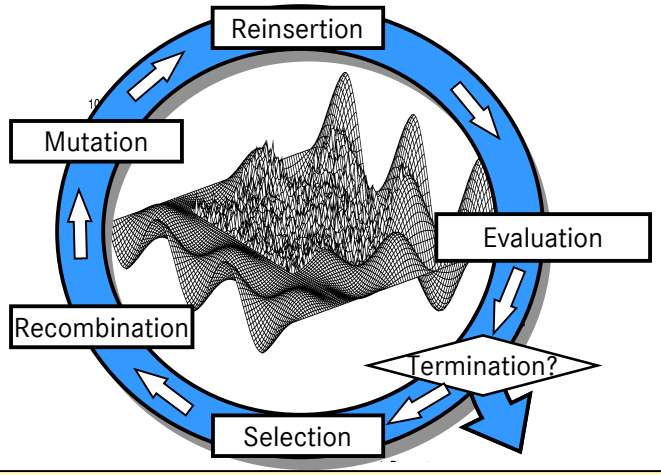



# Evolutionärer Test des Zeitverhaltens



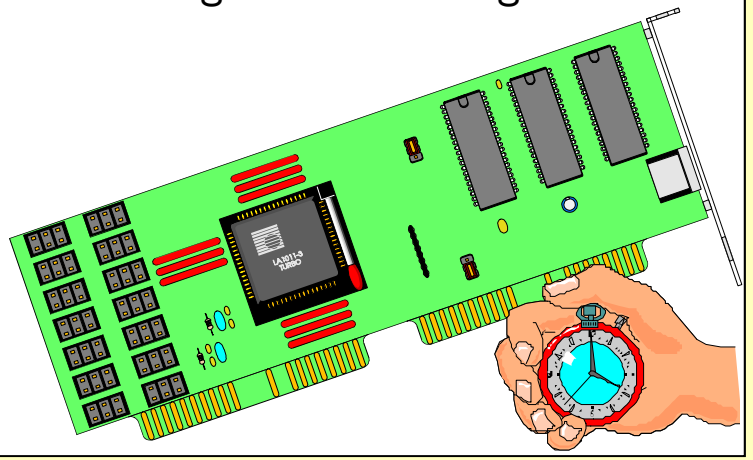
## Testumgebung

Automatische Testdatengenerierung auf Host



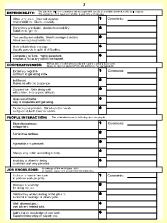

Testdaten

Testdurchführung mit Messung der Ausführungszeiten auf Target



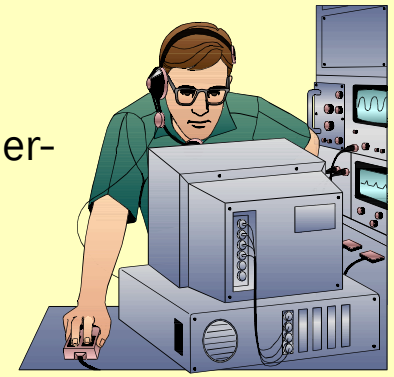
Ausführungszeiten

Testergebnisse



Extreme Ausführungszeiten

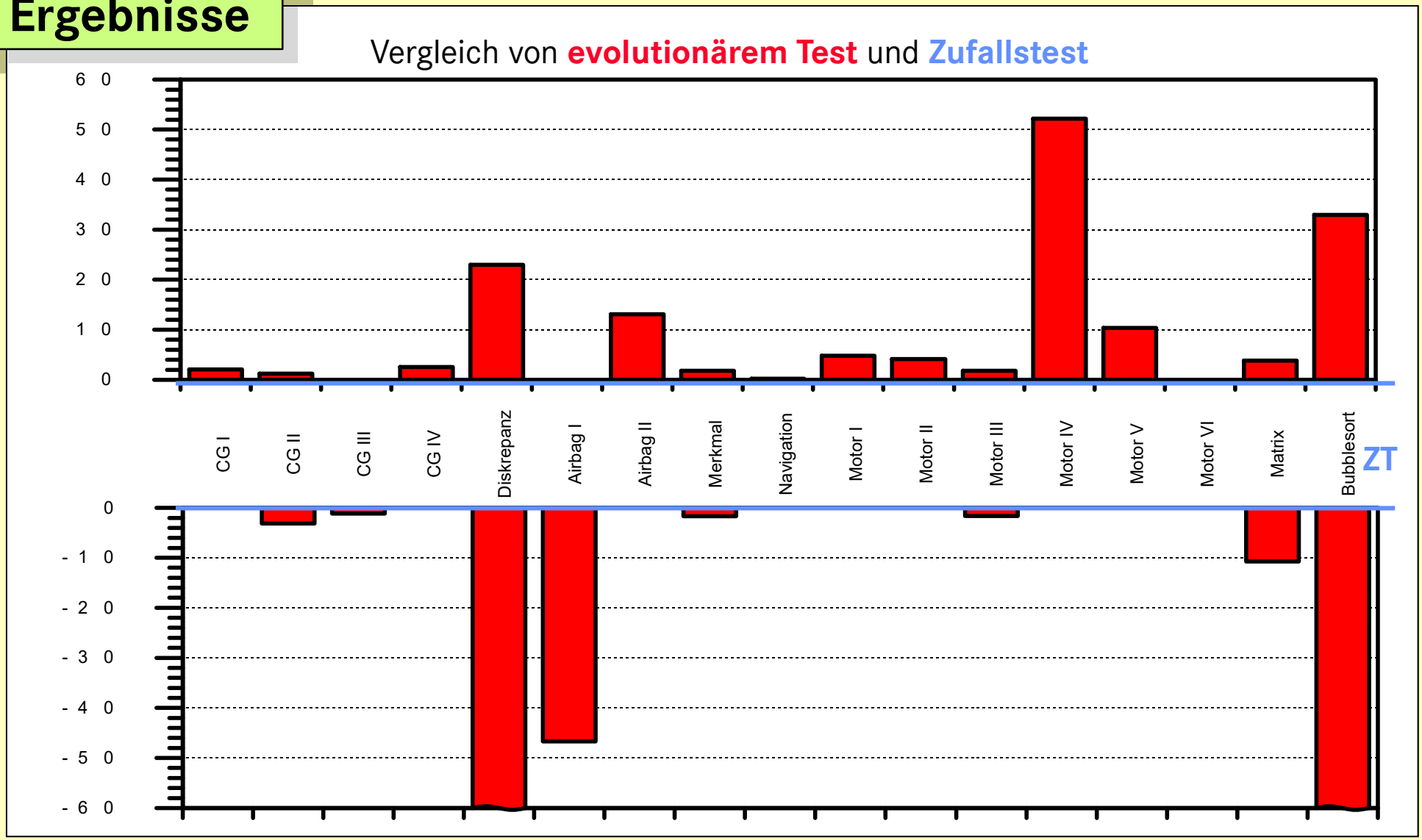
Falls erforderlich: Fehlerbeseitigung



# Evolutionärer Test des Zeitverhaltens



## Ergebnisse

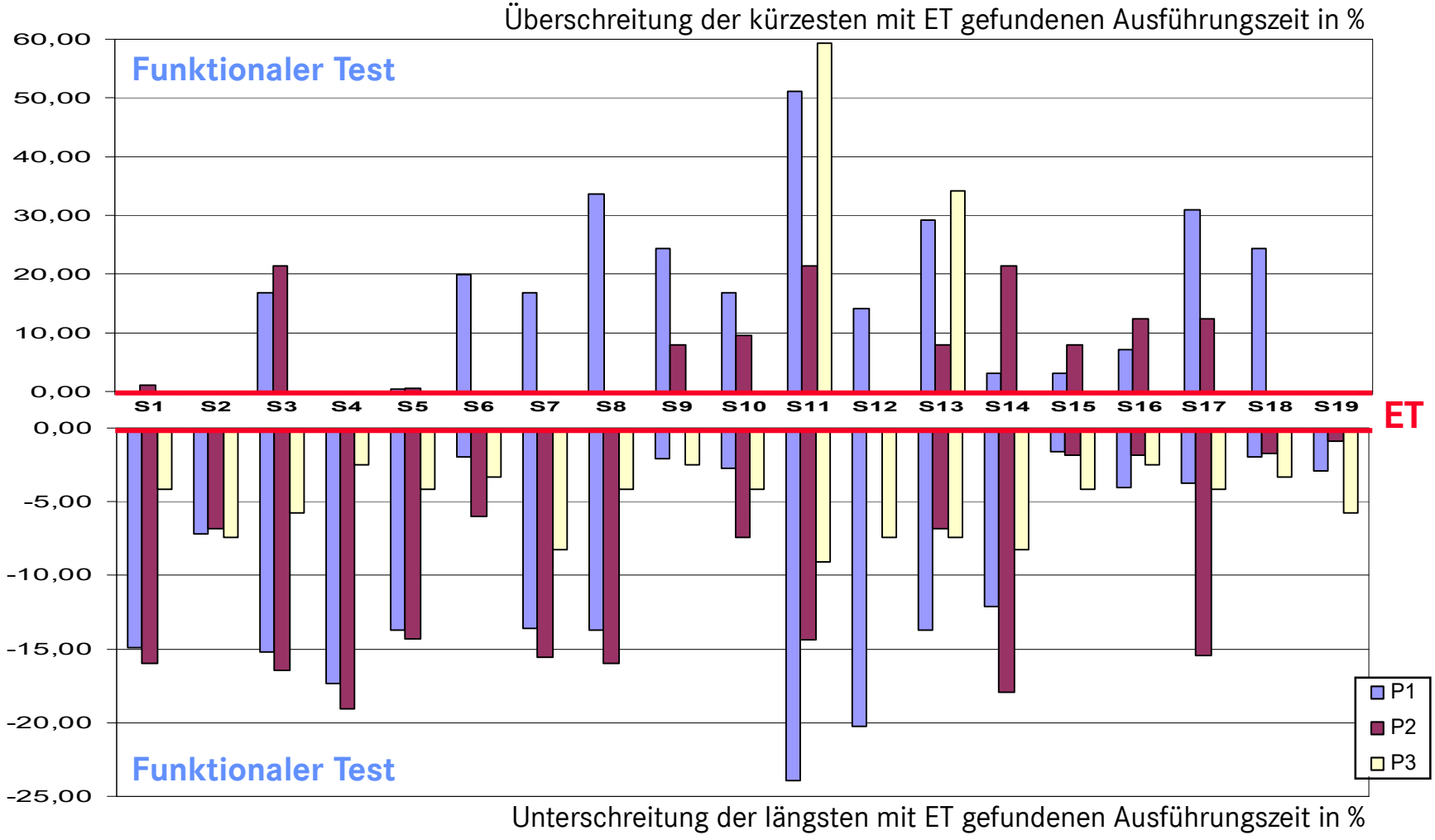


# Evolutionärer Test des Zeitverhaltens



## Ergebnisse

Vergleich von **evolutionärem Test** und **funktionalem Test**

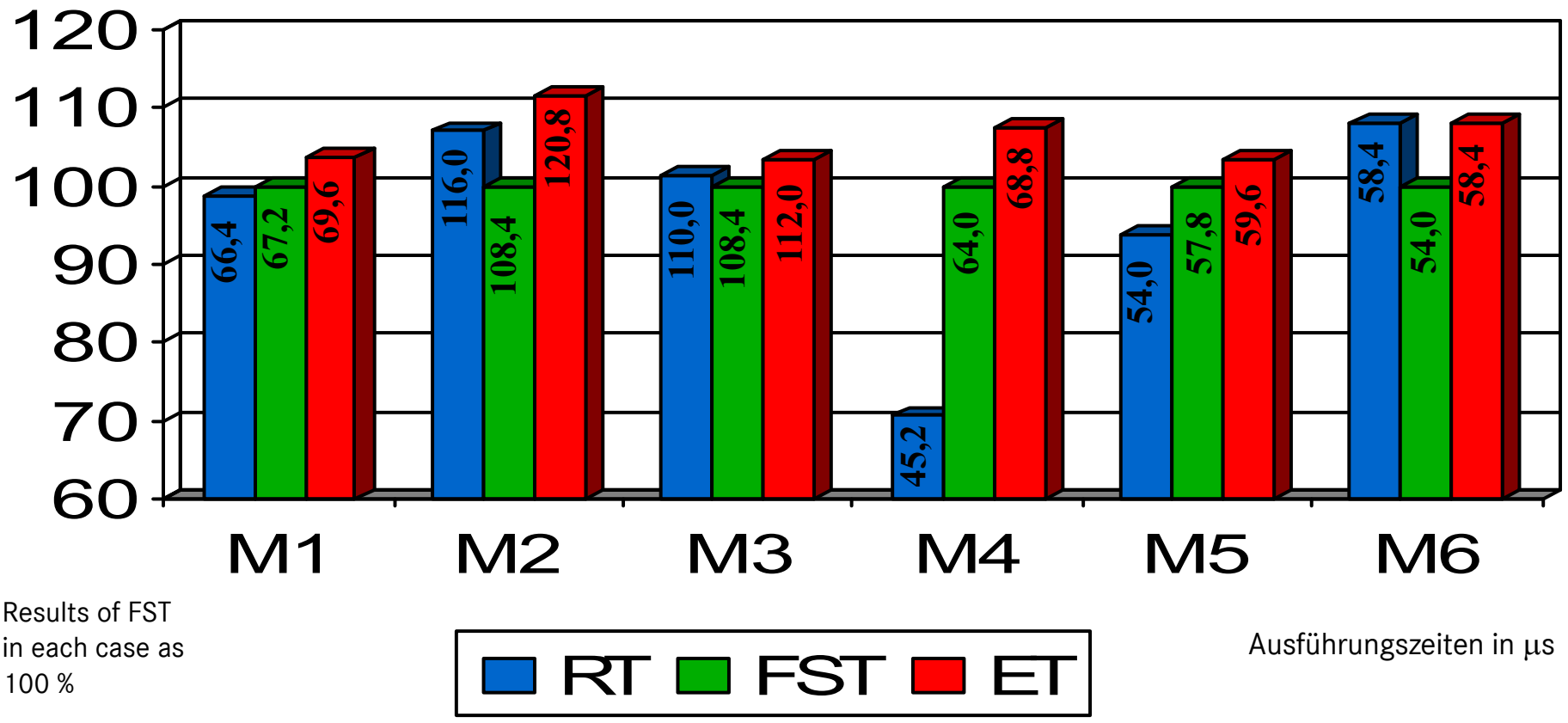


# Evolutionärer Test des Zeitverhaltens



## Ergebnisse

Vergleich von **evolutionärem Test (ET)**, **Funktions- und Strukturtest (FST)** sowie **Zufallstest (RT)**




Results of FST in each case as 100 %

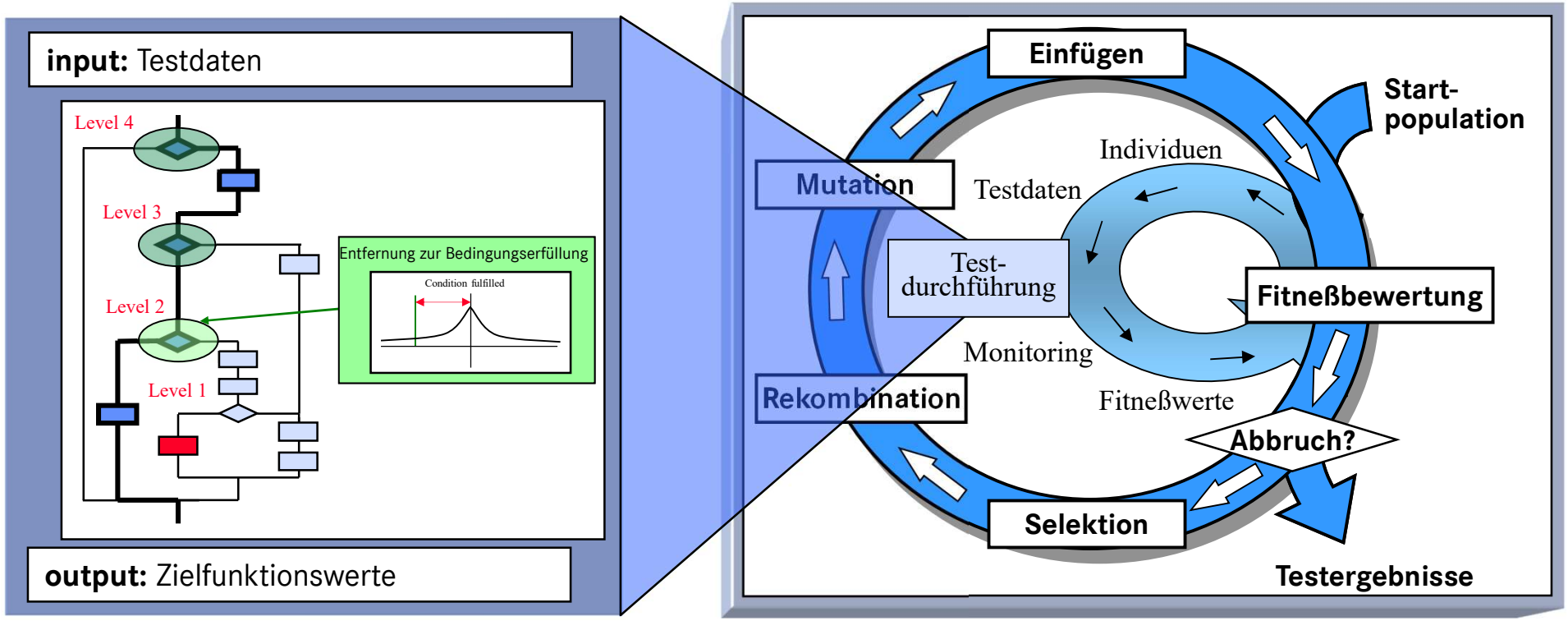
Ausführungszeiten in µs

# Testfallgenerierung für Strukturtests

Ziel Automatische Testfallgenerierung für Strukturtests

## Testidee

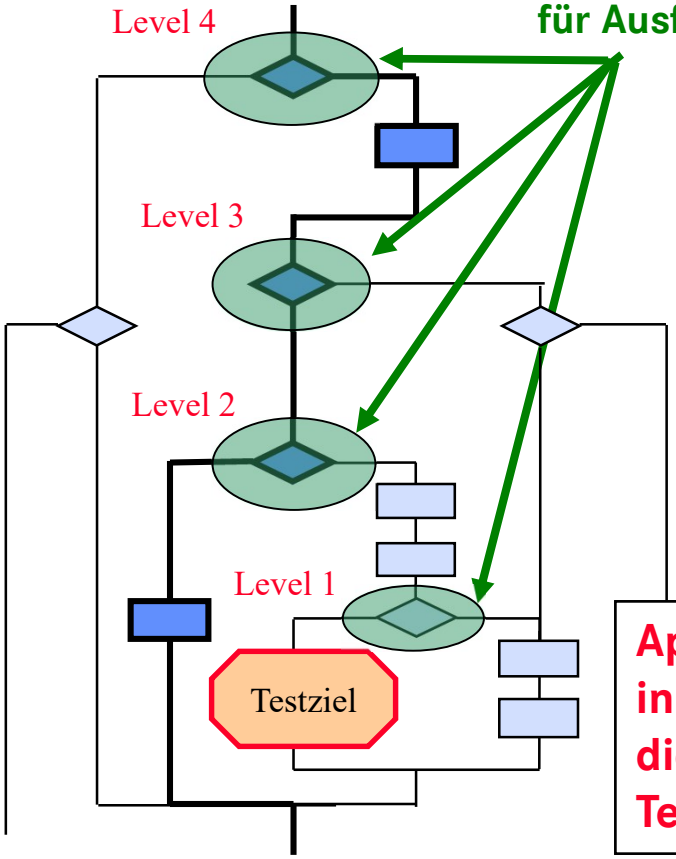
- Test wird in Teilziele (Anweisungen, Zweige, Bedingungen, Pfade, ...) zerlegt
- Fitneßfunktion auf Basis der Kontrollstrukturen des Testobjekts



# Zielfunktion (1) - Approximation-Level

Zielfunktionsberechnung für generierte Individuen setzt sich aus zwei Komponenten zusammen

**Zielfunktionswert = Approximation_Level + Distance**

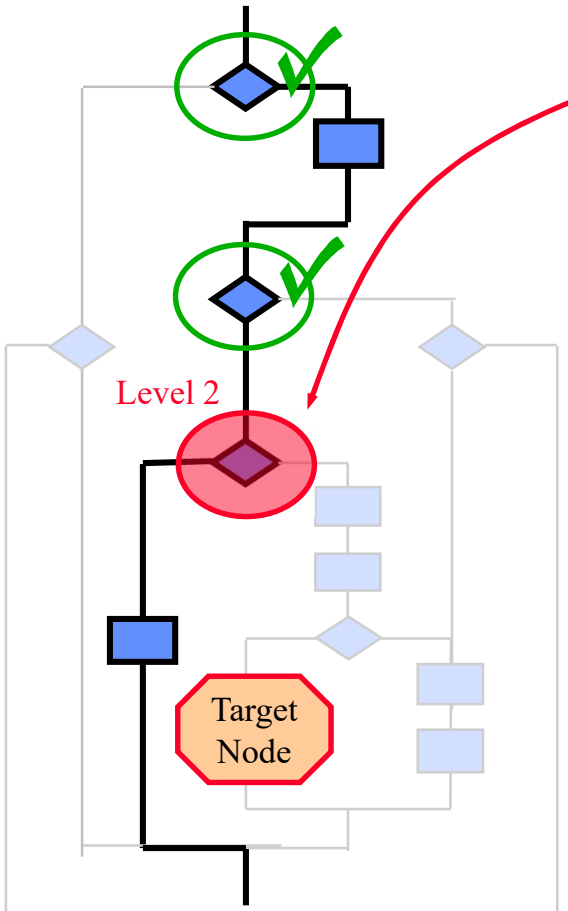



**für Ausführung des Zielknotens *entscheidende* Verzweigungsknoten**

- Ausführung jedes Testziels hängt vom richtigen Durchlaufen einer Menge von Verzweigungsknoten ab
- die für das Erreichen eines Testziels relevanten Verzweigungsknoten werden aus dem Kontrollflußgraphen ermittelt

**Approximation-Level: Für denjenigen Verzweigungsknoten in dem das generierte Testdatum das Testziel verfehlt wird die Anzahl *entscheidender* Verzweigungsknoten bis zum Testziel berechnet**

# Zielfunktion (2) - Distance




**Distance: Für denjenigen Verzweigungsknoten in dem das generierte Testdatum das Testziel verfehlt wird ein Abstand zum Durchlaufen des anderen Zweigs berechnet**

- die Abstandsberechnung erfolgt, indem die Relationen des Verzweigungsknotens in eine Abstandsfunktion umgeformt werden (A *rel* B => *Distance*(A, B))

**Beispiele**

| Relation | Zielfunktion |
|---|---|
| a = b | F = abs(a - b) |
| a ≠ b | F = k |
| a < b | F = (a - b) + k |
| a ≤ b | F = (a - b) |
| a > b | F = (b - a) + k |
| a ≥ b | F = (b - a) |
| a \|\| b | F = min(f(a), f(b)) |
| a && b | F = f(a) + f(b) |

k: minimale Schrittweite

# Ergebnisse

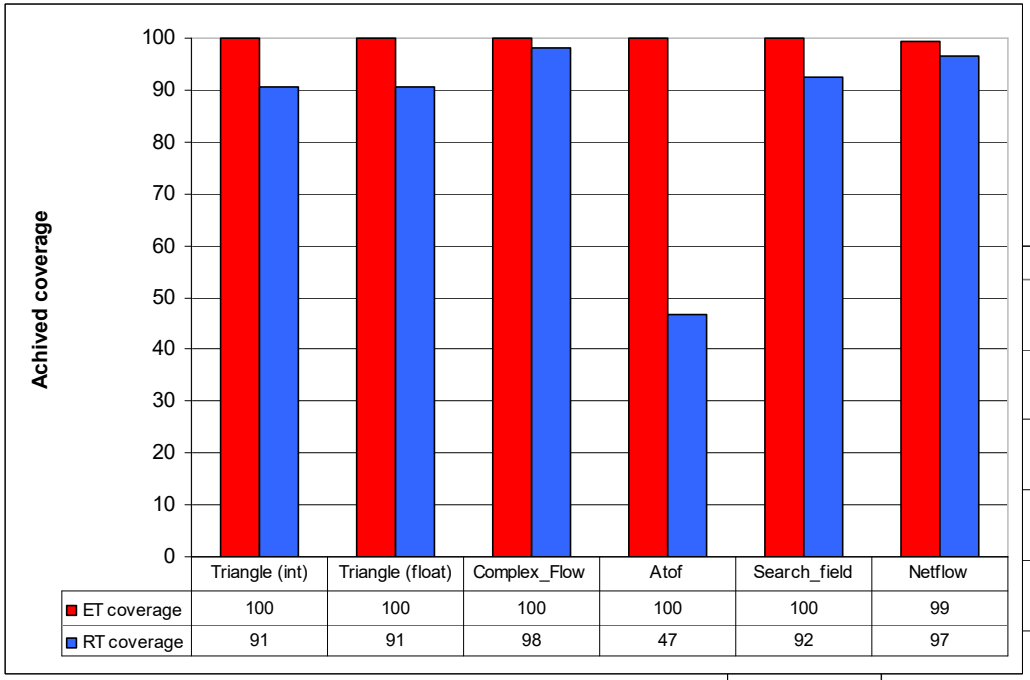
Mit **evolutionären Tests** erreichte Anweisungs- und Zweigüberdeckung

| | atof | | bnldev | | classifyTriangle | | complexBranch | | dtdBody | | exdigx | | firstJan | | gcd | | hqr | | higlowletter1 | | higlowletter3 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SC | BC | SC | | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC |
| Anzahl Testziele | 37 | 55 | 31 | | 30 | 42 | 30 | 41 | 16 | 24 | 25 | 35 | 6 | 7 | 5 | 6 | 76 | 108 | 30 | 42 | 9 | 8 |
| Lines of Code | 37 | | 39 | | 23 | | 31 | | 28 | | 32 | | 9 | | 7 | | 101 | | 40 | | 11 | |
| Zyklom. Komp. | 16 | | 12 | | 6 | | 13 | | 10 | | 14 | | 3 | | 3 | | 31 | | 14 | | 5 | |
| MyersIntervall | 27 | | 3 | | 20 | | 10 | | 12 | | 0 | | 1 | | 1 | | 4 | | 12 | | 3 | |
| Versch.-tiefe nach Piwowarski | 7 | | 45 | | 3 | | 5 | | 14 | | 12 | | 1 | | 0 | | 81 | | 19 | | 3 | |
| Coverage in % bei 10 Testdurchläufen: | | | | | | | | | | | | | | | | | | | | | | |
| Coverage max. | 100 | 100 | 96,77 | | 100 | 100 | 100 | 100 | 100 | 91,67 | 100 | 100 | 100 | 100 | 100 | 100 | 98,68 | 96,26 | 100 | 100 | 100 | 100 |
| Coverage min. | 97,43 | 98,2 | 96,77 | | 96,66 | 100 | 96,66 | 97,56 | 81,25 | 79,19 | 96 | 100 | 100 | 100 | 100 | 100 | 97,36 | 94,39 | 100 | 100 | 100 | 100 |
| Coverage dschn. | 98,97 | 99,1 | 96,77 | | 98,66 | 100 | 99,67 | 99,27 | 88,13 | 80,42 | 99,2 | 100 | 100 | 100 | 100 | 100 | 97,36 | 94,86 | 100 | 100 | 100 | 100 |

| | hail | | isElemInRect | | numlen | | oneif | | repdig | | rotatex | | sdigalt | | sortdig | | sumdigit | | sumdiv | | swapdig | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC | SC | BC |
| Anzahl Testziele | 18 | 25 | 3 | 3 | 21 | 20 | 3 | 3 | 44 | 65 | 45 | 67 | 15 | 21 | 15 | 20 | 21 | 30 | 14 | 19 | 44 | 65 |
| Lines of Code | 20 | | 5 | | 23 | | 5 | | 67 | | 61 | | 21 | | 22 | | 32 | | 18 | | 62 | |
| Zyklom. Komp. | 9 | | 2 | | 11 | | 2 | | 23 | | 24 | | 8 | | 8 | | 11 | | 7 | | 23 | |
| MyersIntervall | 0 | | 3 | | 0 | | 6 | | 0 | | 0 | | 0 | | 0 | | 0 | | 0 | | 0 | |
| Versch.-tiefe nach Piwowarski | 8 | | 0 | | 8 | | 0 | | 8 | | 8 | | 5 | | 2 | | 0 | | 3 | | 8 | |
| Coverage in % bei 10 Testdurchläufen: | | | | | | | | | | | | | | | | | | | | | | |
| Coverage max. | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Coverage min. | 100 | 100 | 66,66 | 100 | 100 | 100 | 100 | 100 | 89,77 | 100 | 100 | 100 | 100 | 100 | 93,75 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Coverage dschn. | 100 | 100 | 97,67 | 100 | 100 | 100 | 100 | 100 | 97,72 | 100 | 100 | 100 | 100 | 100 | 99,38 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

# Ergebnisse

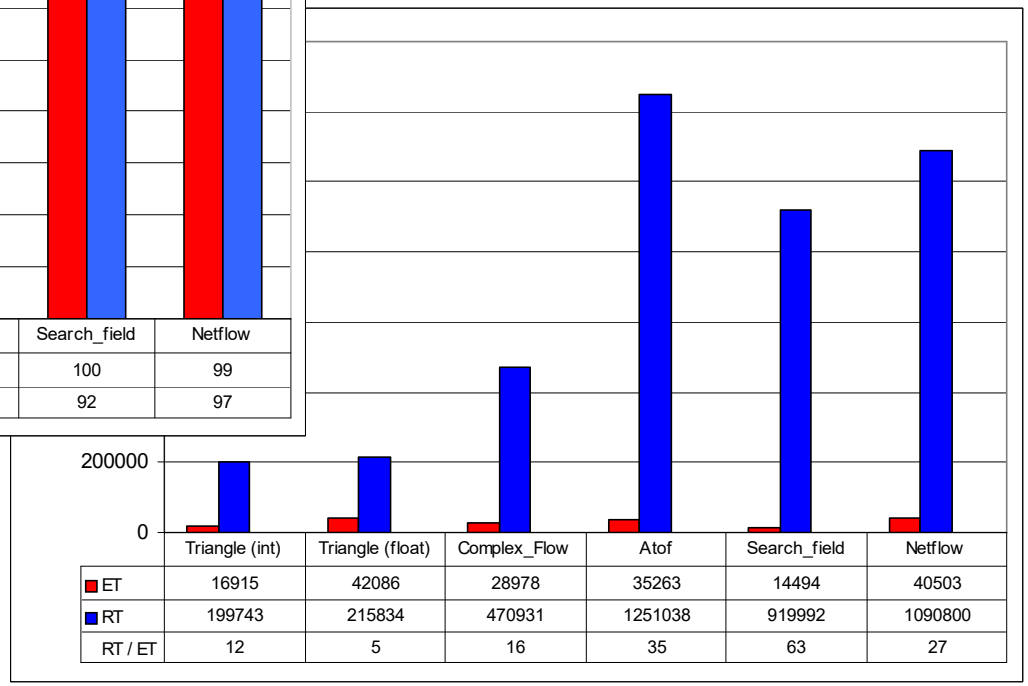
Vergleich von **evolutionärem Test** und **Zufallstest**




| | Triangle (int) | Triangle (float) | Complex_Flow | Atof | Search_field | Netflow |
|---|---|---|---|---|---|---|
| ET coverage | 100 | 100 | 100 | 100 | 100 | 99 |
| RT coverage | 91 | 91 | 98 | 47 | 92 | 97 |

Erreichte Überdeckung

Anzahl generierter Testdaten




| | Triangle (int) | Triangle (float) | Complex_Flow | Atof | Search_field | Netflow |
|---|---|---|---|---|---|---|
| ET | 16915 | 42086 | 28978 | 35263 | 14494 | 40503 |
| RT | 199743 | 215834 | 470931 | 1251038 | 919992 | 1090800 |
| RT / ET | 12 | 5 | 16 | 35 | 63 | 27 |

# Evolutionärer Sicherheitstest



## Sicherheitstest

**Ziel** Für sicherheitsrelevante Systeme werden Sicherheitsanforderungen definiert, die unter keinen Umständen verletzt werden dürfen. Existieren Testdaten, die zu einer Verletzung der Sicherheitsanforderungen führen, so ist das System fehlerhaft.

**Testidee**

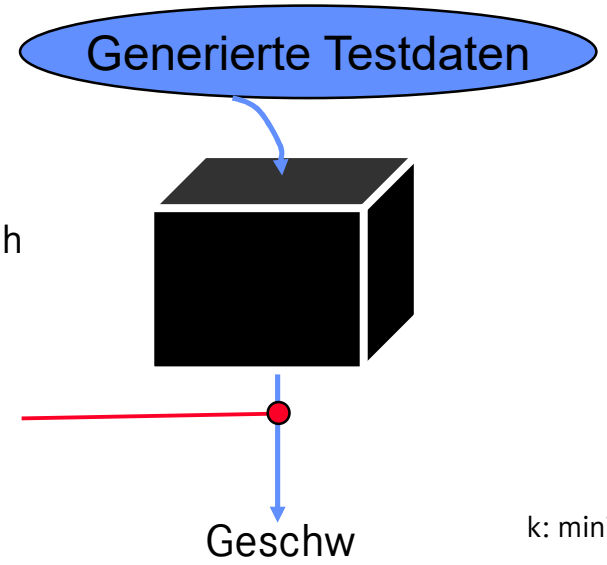
- Testdaten generieren, um Sicherheitsanforderungen zu verletzen
- Zielfunktionswerte werden (analog zum Strukturtest) aus dem Abstand zur Verletzung der Sicherheitsanforderungen berechnet

Sicherheitsanforderung: Geschw ≤ 250 km/h

Testziel: Suche Testdatum mit Geschw > 250 km/h

$$Z = 250 - Geschw + k$$

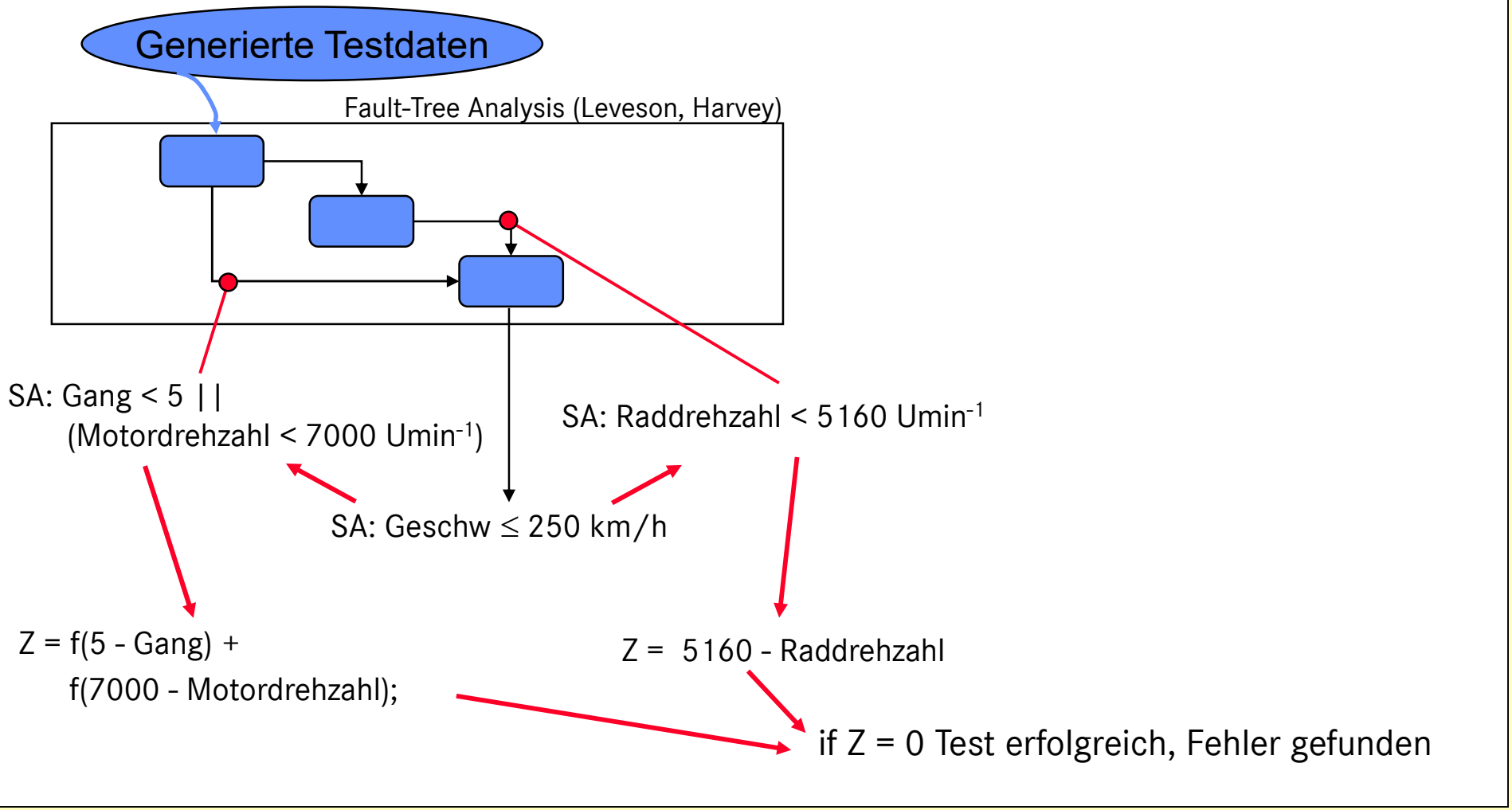
if Z <= 0 Test erfolgreich, Fehler gefunden



# Evolutionärer Sicherheitstest



## Sicherheitstest



## Weitere Anwendungsfelder

- Mutationen-Test
  Erweiterung des evolutionären Strukturtests mit dem Ziel Mutanten aufzudecken

- Robustheitstests
  Prüfung der Fehlertoleranzmechanismen von Systemen

- Funktionstests
  Testdatengenerierung für formal spezifizierte Testfälle mit vergleichbarer Fitneßfunktion wie für Strukturtest und Sicherheitstest.

- Assertion Tests
  Testdatengenerierung, mit dem Ziel assertions im Programm (assert()) zu verletzen.
  Fitneßfunktion wie für Strukturtest und Sicherheitstest.

## Zusammenfassung

- Der Test ist die wichtigste analytische Qualitätssicherungstechnik für Embedded Systems
  - ⇨ es stellen sich zahlreiche Testaufgaben und Testziele
- Neuartiges Testverfahren für die automatische Testfallgenerierung: **evolutionärer Test**
  - beruht auf einer Übersetzung des Testziels in Optimierungsproblem und
  - der Verwendung von leistungsfähigen Suchverfahren für dessen Lösung
- Der evolutionäre Test ermöglicht die vollständige Automatisierung einer Vielzahl praxisrelevanter Tests:
  - Prüfung des Zeitverhaltens von Systemen,
  - Testfallermittlung für Strukturtests,
  - Durchführung von Sicherheitstests u.a.
- Aufgrund des hohen Automatisierungsgrads und der Vielzahl generierter Testdaten bildet der evolutionäre Test eine ideale Ergänzung zu herkömmlichen systematischen Tests.

## Referenzen

Evolutionäres Testen:

- University of York (Tracey, Clark, ...)
  http://www.cs.york.ac.uk/testsig/publications
- Reliable Software Technologies/Cigital (Michael,McGraw, ...)
  http://www.cigital.com/papers
- DaimlerChrysler (Baresel, Pohlheim, Sthamer, Wegener, ...)
  http://www.systematic-testing.com

Evolutionäre Algorithmen

- http://www.geatbx.com/docu/algindex.html
- ftp://ftp.cs.wayne.edu/pub/EC/Welcome.html

Seminal - Software Engineering using Metaheuristic INnovative ALgorithms

- http://www.discbrunel.org.uk/seminal

GECCO 2002 - Search-Based Software Engineering

- http://www.brunel.ac.uk/~csstmmh2/gecco2002