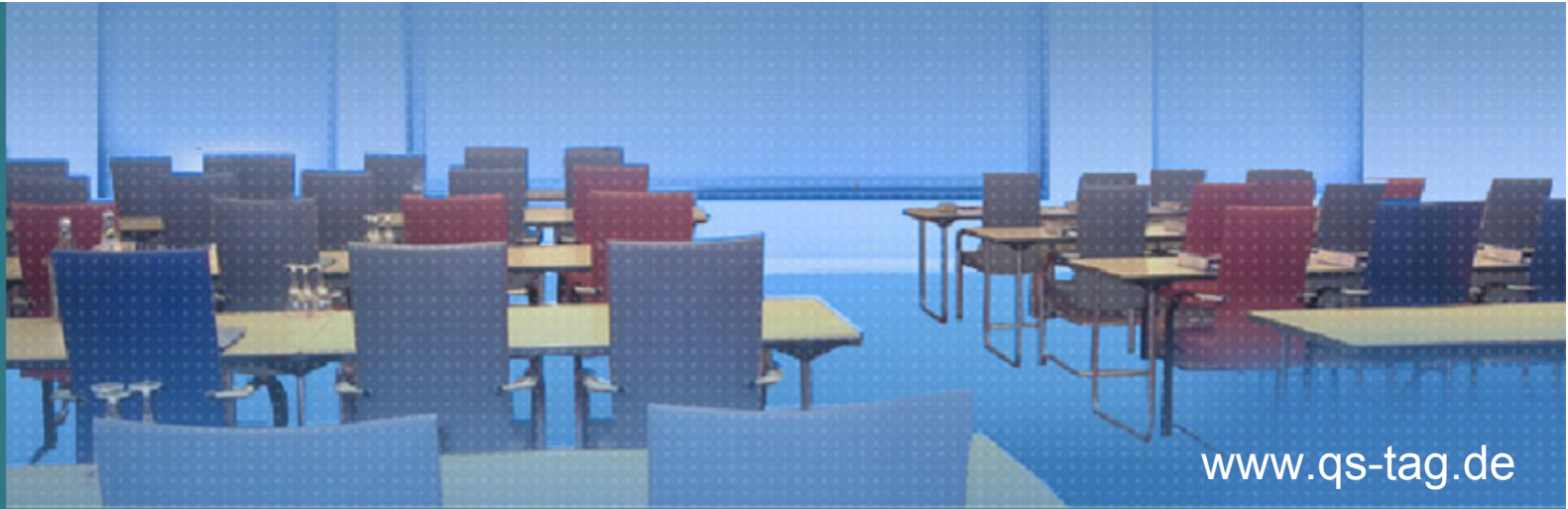


# Erfolgsgeheimnisse im Testmanagement

Auf die Testmanager kommt es an!



## 6 Wege für Module in den Integrationstest

## Erfolgsgeheimnisse im Testmanagement

Dr. Thomas Liedtke  
Informatik Consulting Systems AG



# 6 Wege für Module in den Integrationstest

## Erfolgsgeheimnisse im Testmanagement

Name	Dr. Thomas Liedtke
Funktion	Business Unit Manager Methods, Processes & Tools
Stand	031.1



## Agenda



**1 – Motivation/ Zielsetzung**

**2 – Planungsparameter/ Auswahlkriterien**

**3 – Definition einer Down-Stream-Effektivitäts-Metrik**

**4 – Egal was man tut, es gibt immer High-Runner**

**5 – Lessons Learned**

- Prozessänderungen/ -verbesserungen können viele Gründe haben:
  - **Optimierung** bestehender Prozesse zur Erreichung höherer Produkt- oder Prozessqualität,
  - **Einführung neuer Prozesse** zur Implementierung eines Reifegradmodelles,
  - **Änderung im Entwicklungsprozess** wegen Umorganisation,
  - ...
- Begrenztes Budget:
  - Wie kann das Budget wirtschaftlich optimal eingesetzt werden bei Effizienz vs. Effektivität
  - Wie kann der Budgeteinsatz basierend auf aktuellen Metriken dynamisch zugewiesen werden
- Erreichte höhere Qualität sollte so früh wie möglich nachgewiesen werden (können)
  - Zeitnahe positive Wirkung auf Kunden („... die tun was“)
  - Kurzer Rückmeldungskreis für internes Prozessverbesserungsteam
  - ⇒ frühe Evaluation und Chance auf Erschließung von weiterem Verbesserungspotential
- ⇒ Quantitative Rückmeldung möglichst noch im gleichen Projekt (zeitnah)
  - ⇒ Definition einer Down-Stream-Effektivitäts-Metrik die „gute“ Zahlen liefert
  - ⇒ Entlastung in späteren Projektphasen

## Motivation aus Kundenprojektsituation -1

- **Erfahrungen** zwingen zu Untersuchung möglicher Änderungen in Planungs- und Durchführungsprozessen von Testphasen
  - Große Kundenprojekte soll(t)en sein:
    - Besser **steuerbar**
    - Besser **planbar** in frühen Phasen
    - **Transparenter** bzgl. cost- und time-to-complete, sowie Qualität
    - Besser **vorhersehbar** bzgl. Test-/ Ressourcenengpässen zur Entwicklungszeit
  - **Aufwand zur Fehlerkorrektur steigt** von Phase zu Phase je nach Lebenszyklusmodell zwischen 2 und 10
- ⇒ in der Entwicklung gemachte Fehler müssen **so frühzeitig wie möglich** gefunden werden

## Motivation aus Kundenprojektsituation -2



- Die Einführung von Verfahren zur statischen Code-Analyse war aufgrund des hohen Anteils von Legacy-Codes und fehlenden Werkzeugen nicht möglich
- Vorgestellte Methode(n)
  - wurde erfolgreich in mittleren und großen Projekten angewendet
  - ist **skalierbar** und auch für mittlere und kleine Entwicklungsprojekte erfolgreich einsetz- und anwendbar
  - benötigt für optimale Nutzung historische Daten
- Die Lessons Learned am Ende des Vortrags sind **real erzielte Erfahrungen**
  - Verfahren/ Vorgehen hat sich sehr bewährt und ist in das Regelwerk des Entwicklungsprozesses eingeflossen

## Zielsetzung für die Prozessoptimierung



- Höhere Qualität (im Sinne weniger Restfehler) der Module zum Start Integrationstest
- Breite(re) Anwendung von **Code Inspektionen auf Quellcode**
  - bisherige Praxis von Code Readings soll sinnvoll ersetzt/ ergänzt werden
- Gründliche(re) Unit Tests für Module durch **Erhöhung der Coverage** (C0/ C1)
- **Effektives Kriterium** für eine Entscheidung auf welche Module Code Reading, Code Inspektion und Unit Test angewendet wird, soll vereinheitlicht und verbessert werden
- „Nebeneffekte“:
  - Bisher beobachtete 80:20 Korrekturverteilung (80% der Korrekturen in 20% der Module) während der Testphasen soll entzerrt werden
    - Z.B. durch Anwendung einer **Kritikalitätsvorhersage** um den Flaschenhals in der Korrektur zu verbreitern
  - Erhalt eines Mittels zur Steuerung von Unittests und Testaktivitäten über den gesamten Lebenszyklus

- Einführung von Inspektionen (v.a. für große Projekte) bedeutet:
  - Hoher logistischer Planungsaufwand (zertifizierte Inspektionsleiter, Quiet Rooms, Reviewer, qualifizierte Checker, ...)
  - Hoher Zusatzaufwand in frühe Projektphase verglichen mit historischen Projekten
- Breite hohe Einführung und sofortige Akzeptanz ist unrealistisch
- Piloten **müssen** erfolgreich sein
- Neue Prozesse müssen **effizient** UND **effektiv** sein
- ⇒ gutes **Auswahlkriterium** für Module auf die neue Prozesse angewendet werden sollen, müssen vorab klar definiert werden
- ⇒ Änderung der „**Spielregeln**“ während dem laufenden Projekt riskant

## Agenda



1 – Motivation/ Zielsetzung

2 – Planungsparameter/ Auswahlkriterien

3 – Definition einer Down-Stream-Effektivitäts-Metrik

4 – Egal was man tut, es gibt immer High-Runner

5 – Lessons Learned

## Planungsparameter: Code Review

- Verfügbarkeit von trainierten und **zertifizierten Inspektion-Leiter**
- **Designaufwand** für das einzelne Module (geplant/ actual) als Indikator für die Größe der neuen oder geänderten SLoC (Source Lines of Code)
- Verfügbarkeit und Expertise der **Checker**
- **Lese-/ Checking-Geschwindigkeit** (wichtig für den notwendigen Aufwand)
- **Anzahl notwendiger Lesesitzungen** bestimmt die Dauer
- **Umfang von neuen/ oder geänderten SLoC**
- Qualitätsziele: als **kritisch identifizierte Bereiche/ Module** müssen entsprechend zusätzlich berücksichtigt werden
- **Verfügbarkeit des Quellcodes**; Meilensteine um Eingangskriterien für Inspektionen treffen zu können
- „Dynamische Planung“/ **Budget**

## Planungsparameter : Unit Test

- Notwendige **Sequenz der Module** für den Start-up des Integrationstests
- **Verfügbarkeit existierende Modultestumgebungen** (Initialisierungsaufwand)
- **Plan-Endedaten** der geplanten Code Reviews
- **Verfügbarkeit gereviewter Testlisten** als Eingangsparameter Unit Test
- **Umfang von neuen/ oder geänderten SLoC**
- „Dynamische Planung“/ **Budget**

## Auswahlkriterium: Code Reviews -1

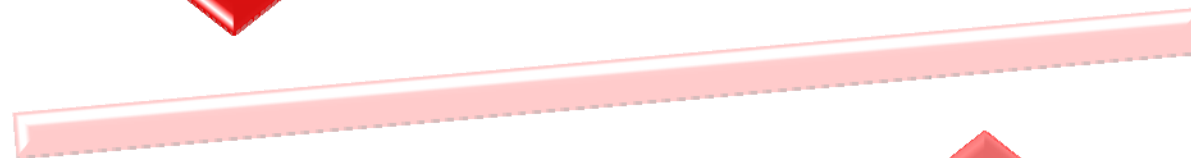


Vorgeschlagene Methode ist...



**zu schwach**

- Hohe Effizienz  
aber niedrige  
Effektivität



**zu stark**

- Niedrige  
Effizienz wegen  
geringer  
Fehlerdichte



**Effizienz:** Aufwand pro gefundenem Fehler

**Effektivität:** Ratio zw. entdeckten Fehlern und Restfehler (Ausbeute)

## Auswahlkriterium: Code Reviews -2



## Auswahlkriterium: Unit Test



„Oder“ Kriterien:

**Verfügbarkeit einer  
wiederverwendbaren  
Testumgebung aus  
Vorgängerprojekt**

**Hoher  
Codierungsaufwand**  
(Indikator für  
komplexere  
Änderungen) größer  
als 40 Stunden

Modul ist identifiziert  
als **potentiell  
fehlerträchtig** gemäß  
Kritikalitätsvorhersage

## Agenda



1 – Motivation/ Zielsetzung

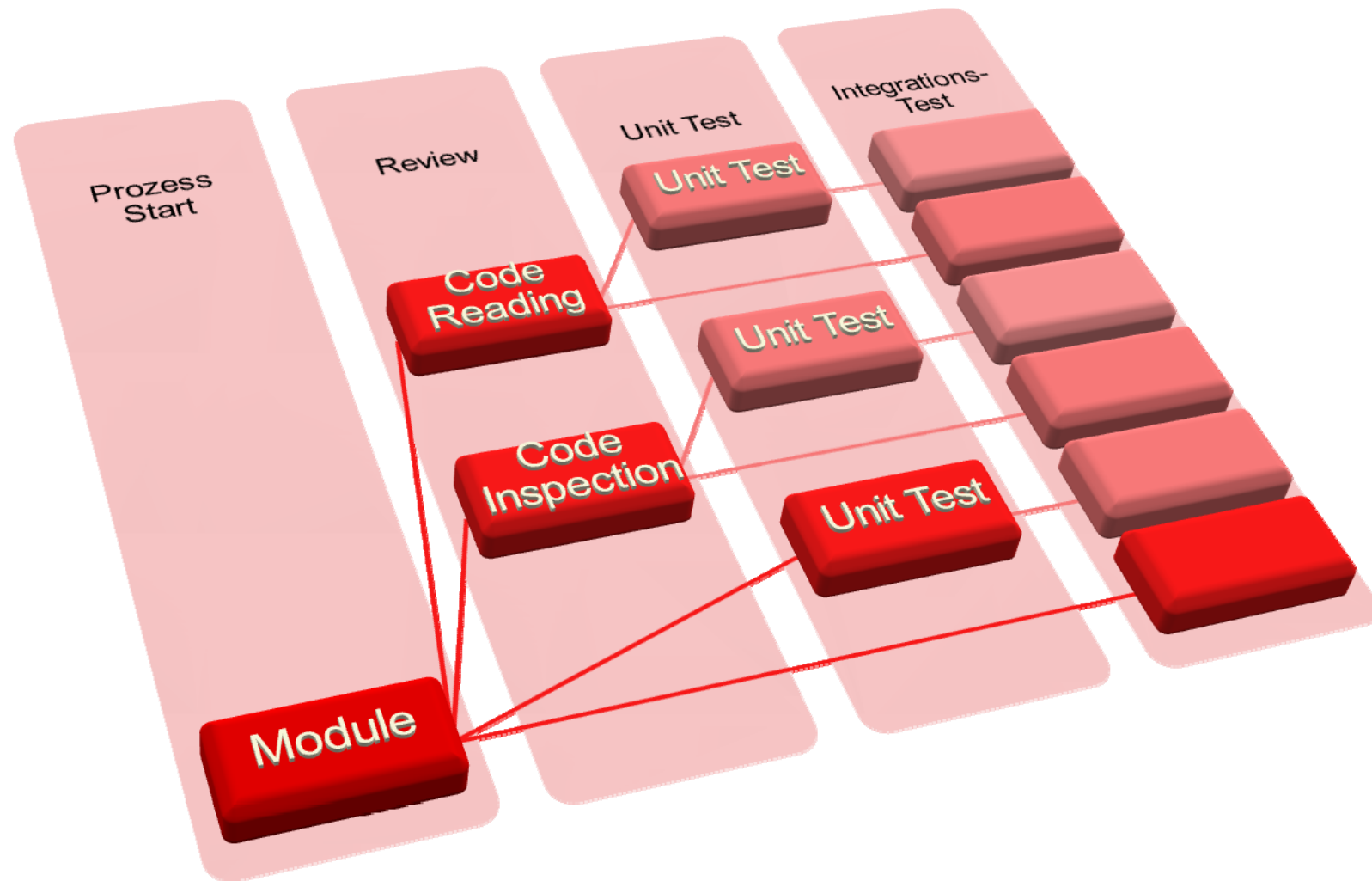
2 – Planungsparameter/ Auswahlkriterien

3 – Definition einer Down-Stream-Effektivitäts-Metrik

4 – Egal was man tut, es gibt immer High-Runner

5 – Lessons Learned

## Definition einer Down-Stream-Effektivitäts-Metrik



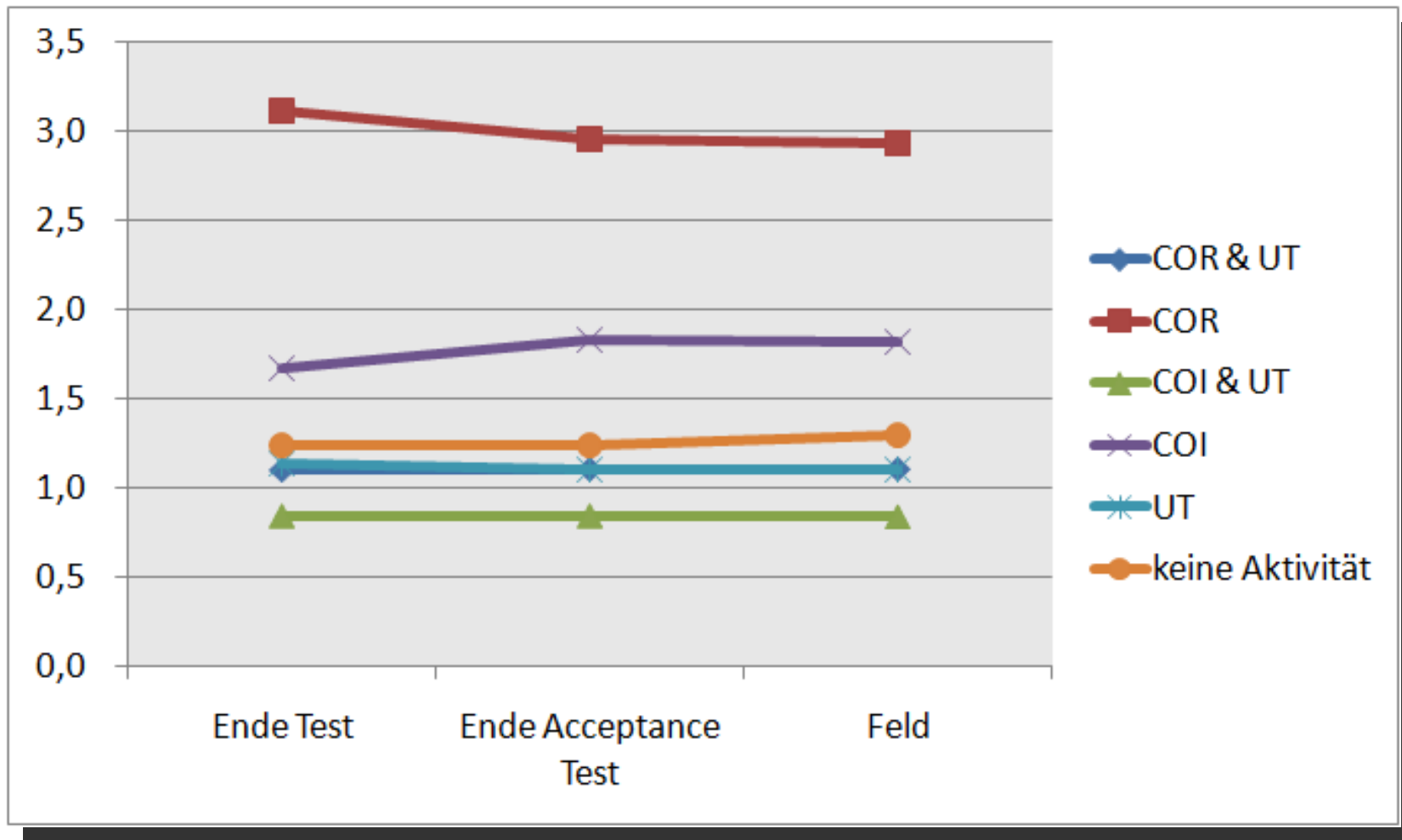
## Die 6 Wege in den Integrationstest



Gruppe von Modulen	CR	CI	UT	Anteil in %			relativer Faktor
				Module	nSLoC	Fehler intern Test	
Code Reading und Unit Test	x		x	34	26	29	1,1
Code Reading, kein Unit Test	x			9	2	6	3,0
Code Inspektion und Unit Test		x	x	31	61	51	0,8
Code Inspektion, kein Unit Test		x		1	1	2	1,8
kein Code Reading und Unit Test			x	5	3	3	1,1
kein Code Reading, kein Unit Test				21	7	9	1,2
Summe				100	100	100	1,0

- Die Entscheidung Code Reviews/ Unit Tests für bestimmte Module entfallen zu lassen ist gerechtfertigt
- „Erfolgreichste“ Gruppe ist die Gruppe für die Code Inspektion und Unit Test durchgeführt wurde
- Code Inspektionen führen zu besserer Ausgangsqualität des Quelltextes als Code Readings
- „Schlechteste“ Gruppe ist die Gruppe von Modulen für die nur Code Reading angewendet wurde
- Einführung von Code Inspektionen führte zu einer erhöhten Fehlerentdeckung in frühen Phasen

## Fehlerdichte in Relation zum Durchschnitt



## Evaluierung



- Unterschiede wurden geringer
- Die Fehlerdichte für inspizierte Module nimmt nach Ende Test zu
- Anzahl Fehler in „alten“ Modulen nimmt nach Ende Test zu
- Code Reading zusätzlich zum Unit Test zeigt kaum Wirkung

## Fehlerdichte in verschiedenen Testphasen



Gruppe von Modulen	CR	CI	UT	Anteil in %		Fehlerdichten			
				Module	nSLoC	vor Test	Test intern	Acceptance	Feld
Code Reading und Unit Test	x		x	34	26	9,5	20,5	1,5	0,2
Code Reading, kein Unit Test	x			9	2	7,4	55,1	1,1	0,0
Code Inspektion und Unit Test		x	x	31	61	12,7	15,7	1,1	0,0
Code Inspektion, kein Unit Test		x		1	1	7,2	34,2	5,2	0,0
kein Code Reading und Unit Test			x	5	3	0,0	20,6	0,9	0,0
kein Code Reading, kein Unit Test				21	7	0,5	23,1	1,6	1,3
<b>Summe</b>				100	100	<b>1,1</b>	<b>18,6</b>	<b>1,2</b>	<b>0,2</b>

## Agenda



1 – Motivation/ Zielsetzung

2 – Planungsparameter/ Auswahlkriterien

3 – Definition einer Down-Stream-Effektivitäts-Metrik

4 – Egal was man tut, es gibt immer High-Runner

5 – Lessons Learned

## Egal was man tut, es gibt immer High-Runner

- Egal was man tut.. es gibt immer...
  - eine **High-Runner-Liste** für Module die Fehler enthalten
  - eine **Top 10 Liste** der fehlerbehafteten Module
- Ursachenforschung der High-Runner-Liste zu Projektende:
  - oben auf der Liste sollte kein „**Überraschkandidat**“ erscheinen
    - Kein Modul welches „übersehen“ wurde
  - Die **Gesamtanzahl** der Fehler innerhalb der High-Runner-Liste sollte sich verringern
  - Die Verteilung der Fehler sollte **homogener** sein (besser als 80:20)

## Evaluierung einer Top 10 Liste

- Für alle 10 Module wurde Unit Test durchgeführt, ABER
  - für sechs Module nicht gemäß vorgegebenen Regeln
- Für 9 von 10 Modulen wurde Code Inspektionen durchgeführt:
  - 2 Code Inspektionen wurden frühzeitig beendet (early exit) wegen schlechte Quelltextstruktur
  - 2 Code Inspektionen wurden nicht nach vorgegebenen Regeln durchgeführt (zu hohe Checking-Rate)
- Die Gesamtanzahl der Fehler die in der Top 10-Liste gefunden wurde konnte um einen **Faktor 10 (!)** verringert werden
- Kein Modul, für das Qualitätssichernde Maßnahmen in frühen Designphasen durchgeführt wurden ist auf der TOP 10-Liste

## Verbesserung der 80-20 Verteilung

- 80% der Fehler in 20% der Module bedeutet
  - 80% der Korrekturen müssen in 20% der Module durchgeführt werden

⇒ **Flaschenhals für die Korrekturperformance!**

% - Anteil Module	% Anteil Korrekturen
20%	63%
30%	81%
40%	91%

## Agenda



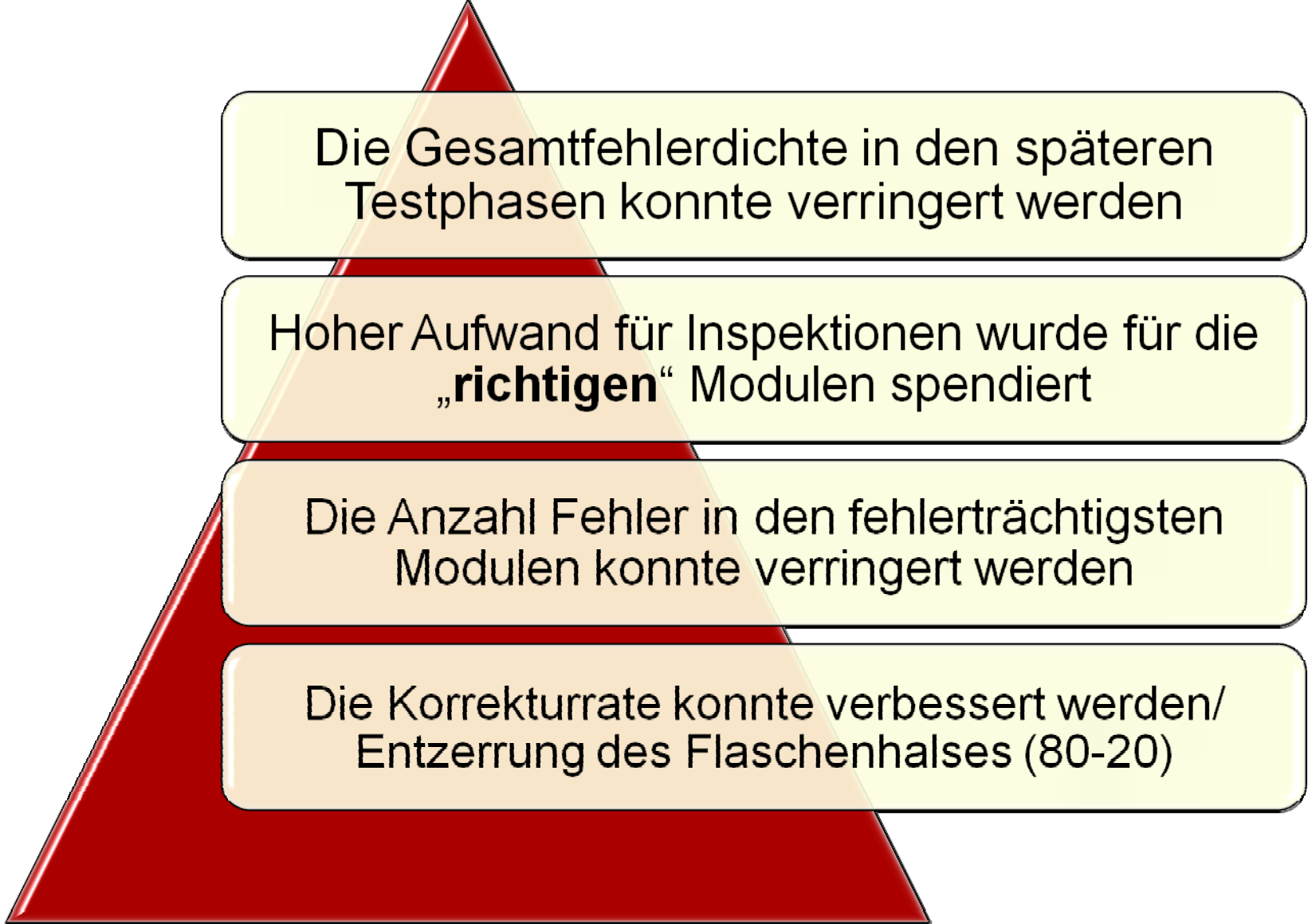
1 – Motivation/ Zielsetzung

2 – Planungsparameter/ Auswahlkriterien

3 – Definition einer Down-Stream-Effektivitäts-Metrik

4 – Egal was man tut, es gibt immer High-Runner

5 – Lessons Learned



Die Gesamtfehlerdichte in den späteren Testphasen konnte verringert werden

Hoher Aufwand für Inspektionen wurde für die „**richtigen**“ Modulen spendiert

Die Anzahl Fehler in den fehlerträchtigsten Modulen konnte verringert werden

Die Korrekturrate konnte verbessert werden/  
Entzerrung des Flaschenhalses (80-20)



**Herzlichen Dank für Ihre  
Aufmerksamkeit!**



## Kontakt Daten

Name: Dr. Thomas Liedtke  
Firma: Informatik Consulting Systems AG  
BU Methods Processes & Tools  
Sonnenbergstraße 13  
70184 Stuttgart  
DEUTSCHLAND  
E-Mail: [thomas.liedtke@ics-ag.de](mailto:thomas.liedtke@ics-ag.de)  
Tel.: 0711 21037-39