

// Qualitätssicherung Separierung von Entwicklung und Qualität

Christian Schmid

Version 1.0

// Ausgangssituation

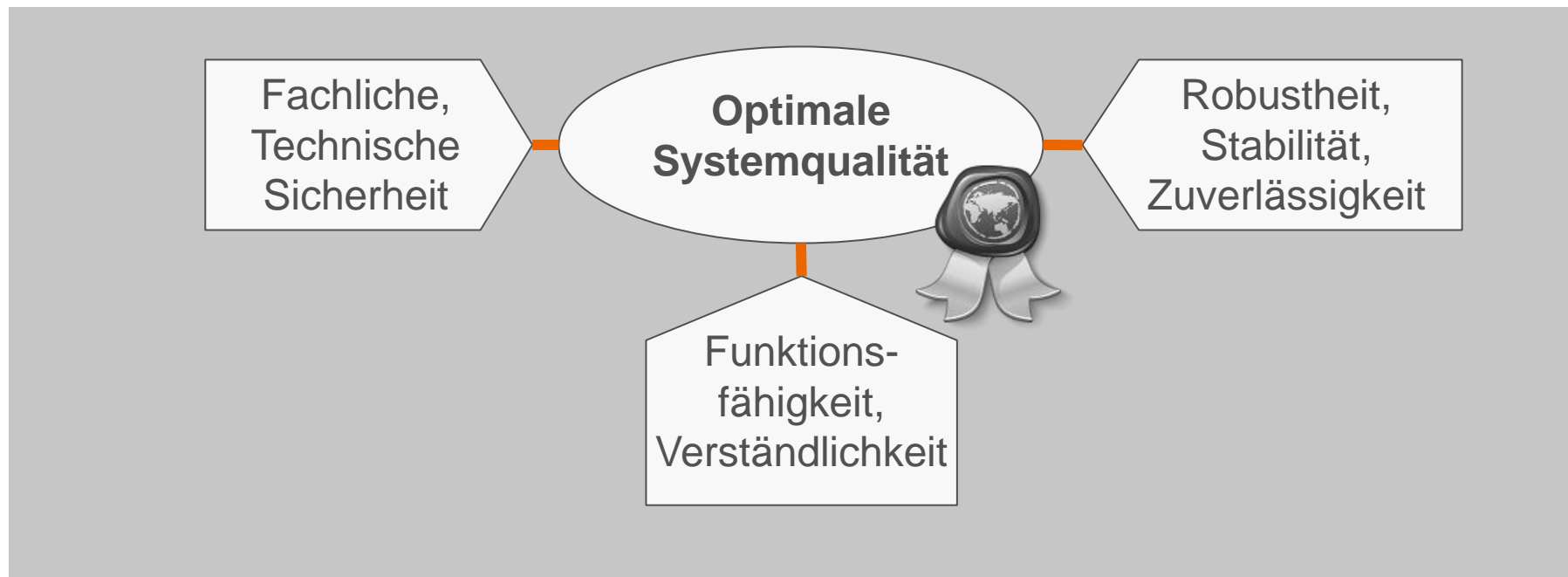
Die Software wurde fertig geplant oder entwickelt.

Das Ergebnis soll qualitativ Einwandfrei sein, damit eine „optimale Systemqualität“ erreicht werden kann.



// Was ist „optimale Systemqualität“?

Def.: Der Begriff der „**optimalen Systemqualität**“ bezeichnet alle am Geschäftszweck ausgerichtete Maßnahmen, welche in der Abwägung **Kosten und Risiko** zu maximaler Sicherheit, Robustheit und Funktion eines Softwaresystems führen.



// Separierung Entwicklung/Qualitätssicherung



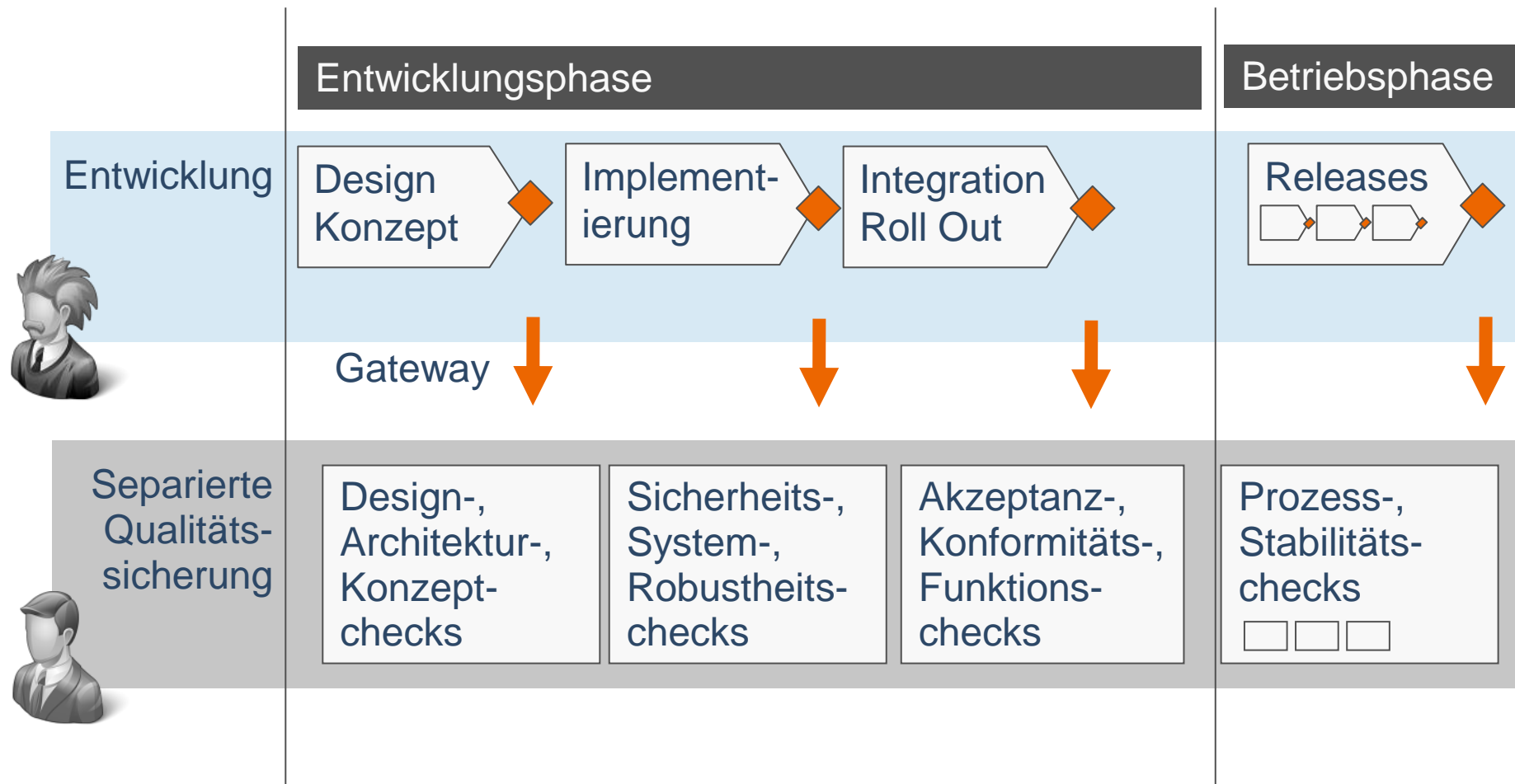
Entwickler



Qualitätssicherung

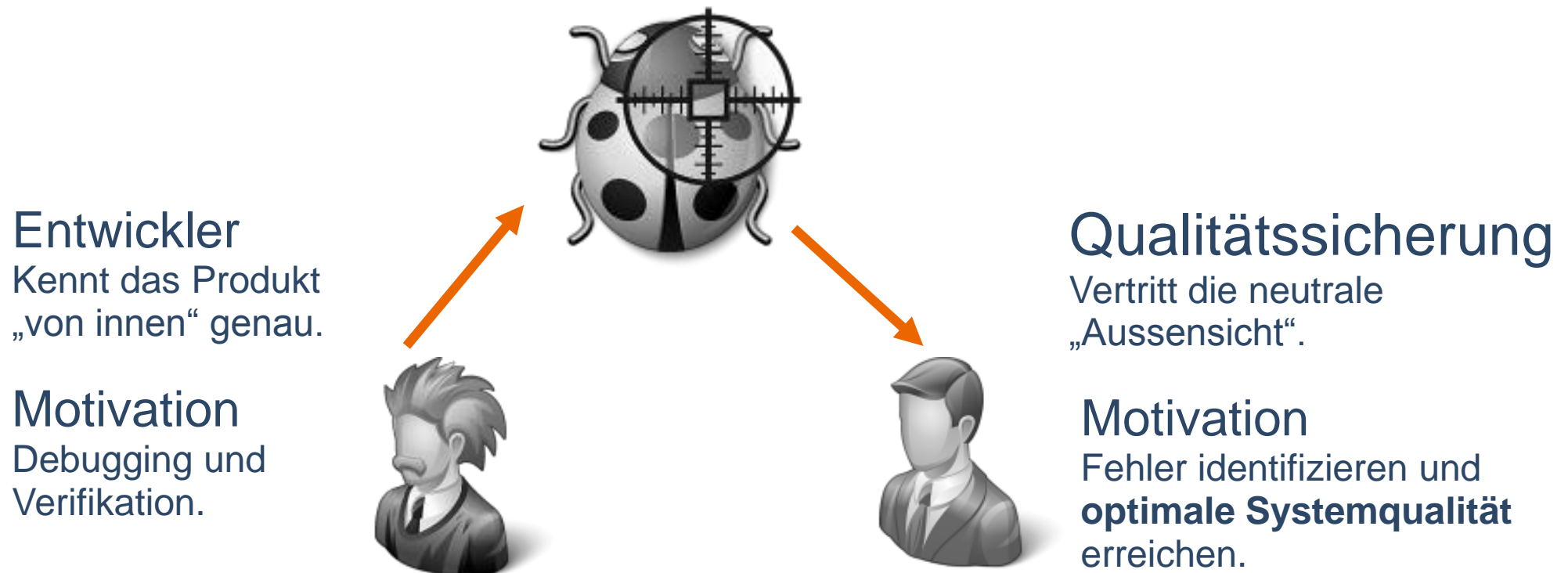
Die „**optimale Systemqualität**“ kann deutlich besser erreicht werden, wenn Entwicklung und Qualitätssicherung separiert werden.

// Zeitpunkte/Möglichkeiten der Separierung



// Die Rollen von Entwickler und Qualitätssicherung

„Bugs“ schleichen sich nicht ein, sie werden gemacht



// Aufgaben der Qualitätssicherung

Was macht die separierte Qualitätssicherung?

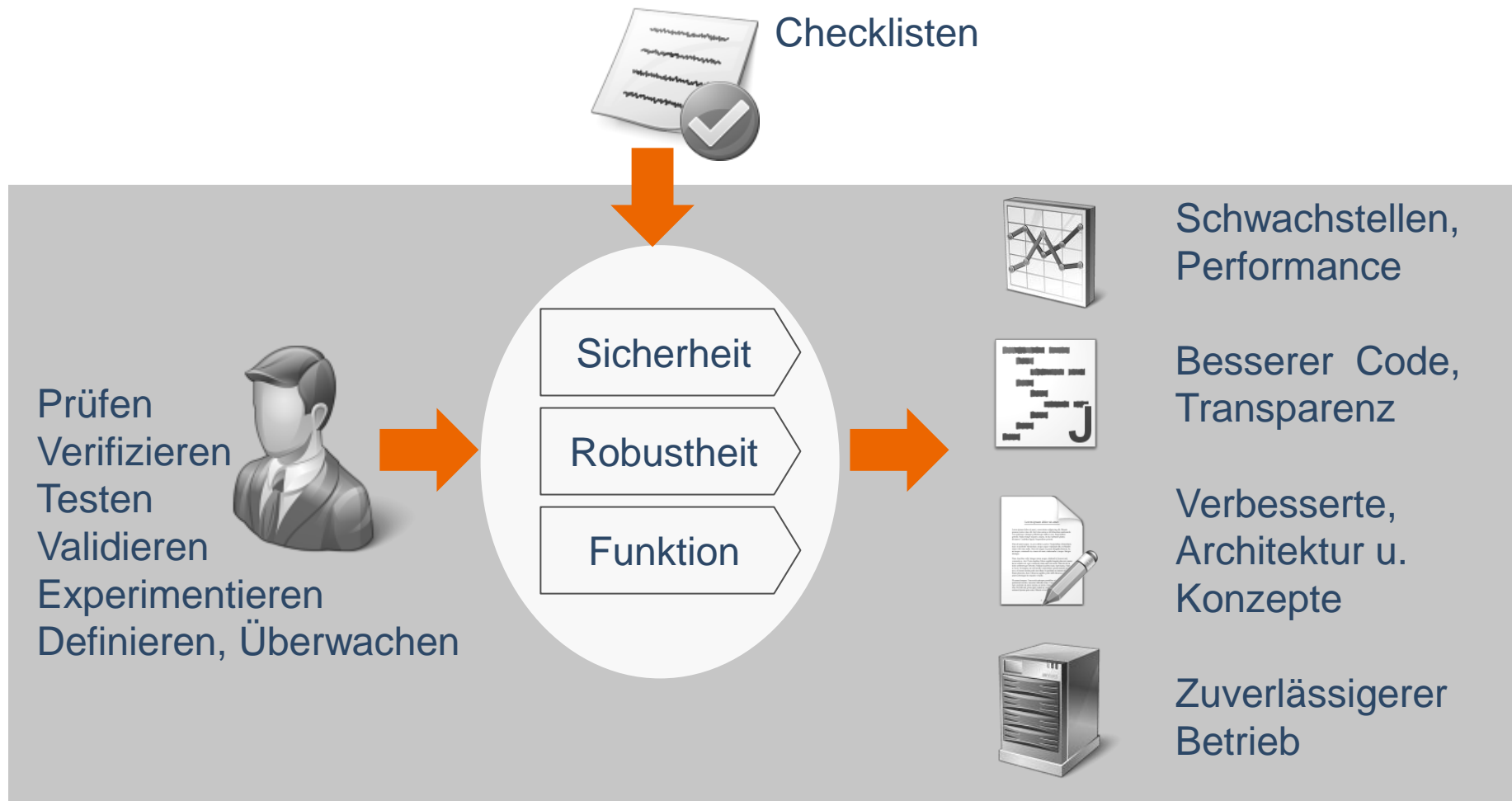
QA CHECKLIST—Implementation Project

Project Name: _____

Prepared By: _____

Item	Yes/No
Have customers completed review of System Testing results?	
Have the List of Deliverables and Issue Log been signed by the QA manager?	
Has the customer training been conducted?	
Has the customer project documentation been delivered to the customer approved the IMPLEMENTATION deliverables?	
Has customer acceptance test been completed, tested, and signed?	
Has outstanding issues been discussed with customer?	

// Was kann die Qualitätssicherung tun?



// Was macht die Qualitätssicherung?

Verifikation/Falsifikation Formaler Beweis der mathematisch **korrekten Funktion/Nichtfunktion** eines Algorithmus/Moduls in jedem Fall.

Validierung Dokumentierte Beweisführung, dass ein System den **fachlichen** Anforderungen erfüllt.

Prüfen Vergleich Ist-Zustand mit definiertem Soll-Zustand

Testen/Versuchen Viele Fehler und unklare Effekte aufdecken.

Experimentieren Identifizieren von Effekten außerhalb der Spezifikation (DAU-Test).

Definieren und Überwachen QS-Prozesse, Regeln, Styleguides, Vorgaben, Rahmenbedingungen, Kommunikationsguides, Qualitätsmanagement, Regelabläufe für Betrieb, Incident- und Problemmanagement, ...

// Hauptarbeit von QS ist das Testen

	Entwicklungsphase			Betriebsphase	
Test	Design-Architekturtest	Funktionstests, Strukturtests	System- und Stresstests	Akzeptanztest, Usabilitytest	Konfigurations-test
Objekt	System-entwurf-dokument	Einzelmodule	Teilsysteme, Integration	Gesamtsystem	Eingebettete Hard/Software
Methode	Testfall-beschreibung	Black-Box und Glass-Box Skripten	Module und Treiber	Benutzer-testprotokolle	HW-in-the-loop Tests

// Typische Fragestellung zu Sicherheit

Wie werden sensible Daten geschützt? Wie schwer ist es für Hacker das System anzugreifen?



IDENTITÄT

// Checkliste: Sicherheit

- > Gezielte Prüfung des Codes und der Softwarearchitektur auf Sicherheitslecks
- > Kontinuierliche Security-Checks (Intrusion Detection, DOS-Attacken....)
- > Protokollieren und Monitoren wichtiger Vorgänge im System (Alerts)
- > Verschlüsselung von Daten und Kommunikation
- > Den „Sicherheits“-Markt beobachten um bei neu entdeckten Sicherheitslecks sofort patchen zu können
- > Etablierte und geprüfte Komponenten
- > Stabilitäts- und Konfigurationsprüfungen
- > Unversehrtheit der funktionsfähigen Betriebsumgebung wahren
- > Authentifizierungen, Rechte, Rollen und Identitätsmanagement
- > Organisatorische Sicherheitsmaßnahmen (Zuständigkeiten, Berechtigungen,...)
- > Öffentliche oder interne Richtlinien, Normen oder Vorgaben vollständig entsprechen

// Typische Fragestellung zur Robustheit

Wie „unzerstörbar“ ist das System durch äußere und innere Einflüsse?



// Checkliste: Robustheit

- > Architekturchecks auf Instabilitäten in den Konzepten
- > Stabile Betriebsumgebung und Infrastruktur (Null-Downtime bei Releases, Frozen Zone, Migrationskonzepte, Dispatcher/Routing, ...)
- > Zuverlässigkeit, Schnelle Einspielbarkeit von Fixes durch Softwarehersteller
- > Kontinuierliche Stress-Checks bei Einführung (Lastverteilung, Komponenten, GAU-Verhalten...z.B. Testscripting mit LoadRunner)
- > Wiederaufsetzbarkeit (Backupplanung), Zeitpunkte, Wiederherstellung (z.B. SLA)
- > Risikoanalyse bei größeren Veränderungen (vor allem Schnittstellen)
- > Überprüfung der Prozessdefinitionen für Incidentmanagement, Problemmanagement und Releasemanagement (evtl. ITIL)
- > Codechecks, Architekturchecker Tools (z.B. CodeWarrior)
- > Definition/Prüfung von Design Patterns

// Typische Fragestellung zur Funktionsfähigkeit

Wie stark werden die Anforderungen erfüllt?

Wie stark sind die Funktionen
abwärtskompatibel?

// Checkliste: Funktionsfähigkeit

- > Wiederkehrende manuelle Tests mit wechselnden Usern (z.B. auch Usabilitytests)
- > Tests gegen definierte Use-Cases
- > Verständlichkeit/Einfachheit wahren (Zerstörung durch Unwissenheit)
- > Akzeptanz schaffen („Projektmarketing“), Nutzen klar kommunizieren
- > Baustellenbelastung vermeiden
- > Automatisch wiederholbare Tests (z.B. Scripting mit httpUnit oder LoadRunner)
- > Modulprüfungen
- > Identische Test- und Abnahmeumgebung wie Produktionsumgebung
- > Klare Versionierungsplanung mit Abwärtskompatibilität
- > Zuverlässiger Deploymentmechanismus auch für kleine Module
- > Nachvollziehbarkeit von Änderungen und Funktionen

// Mehr Informationen? Nehmen Sie Kontakt auf!

doubleSlash
Net-Business GmbH
Otto-Lilienthal-Str. 2
D-88046 Friedrichshafen
<http://blog.doubleSlash.de>

Ihr Ansprechpartner
Oliver Belikan
Tel.: 07541-70078-100
info@doubleSlash.de