„Warum haben Sie den Fehler wieder eingebaut …?“

Geschäfts – und Entwicklungsprozesse in der Praxis

5. Februar 2013, Universität Paderborn

Armin Vogt
avogt@s-und-n.de

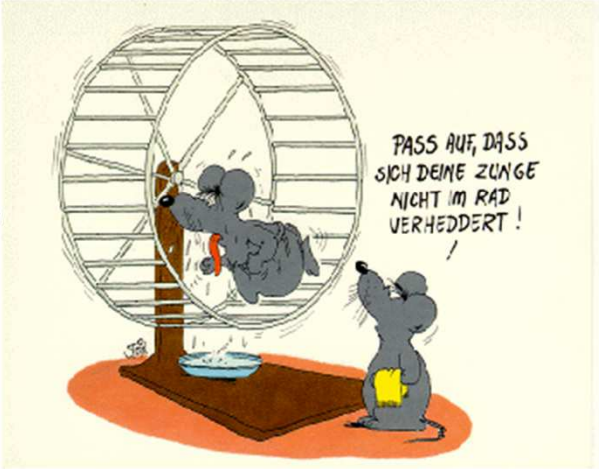
Willkommen

Warum sind wir heute hier?

- Continuous Integration

  oder

  „wie Kodieren unter Stress Spaß machen kann“



Today you will learn/see …(die Botschaft andeuten)

Grundstudium Informatik 1992 beinhaltete nichts über Testen, aber Verifikation!

Was gibt es zum Mitnehmen? → Impuls

06.02.2013 S&N AG © 2013 3

Den **Impuls**

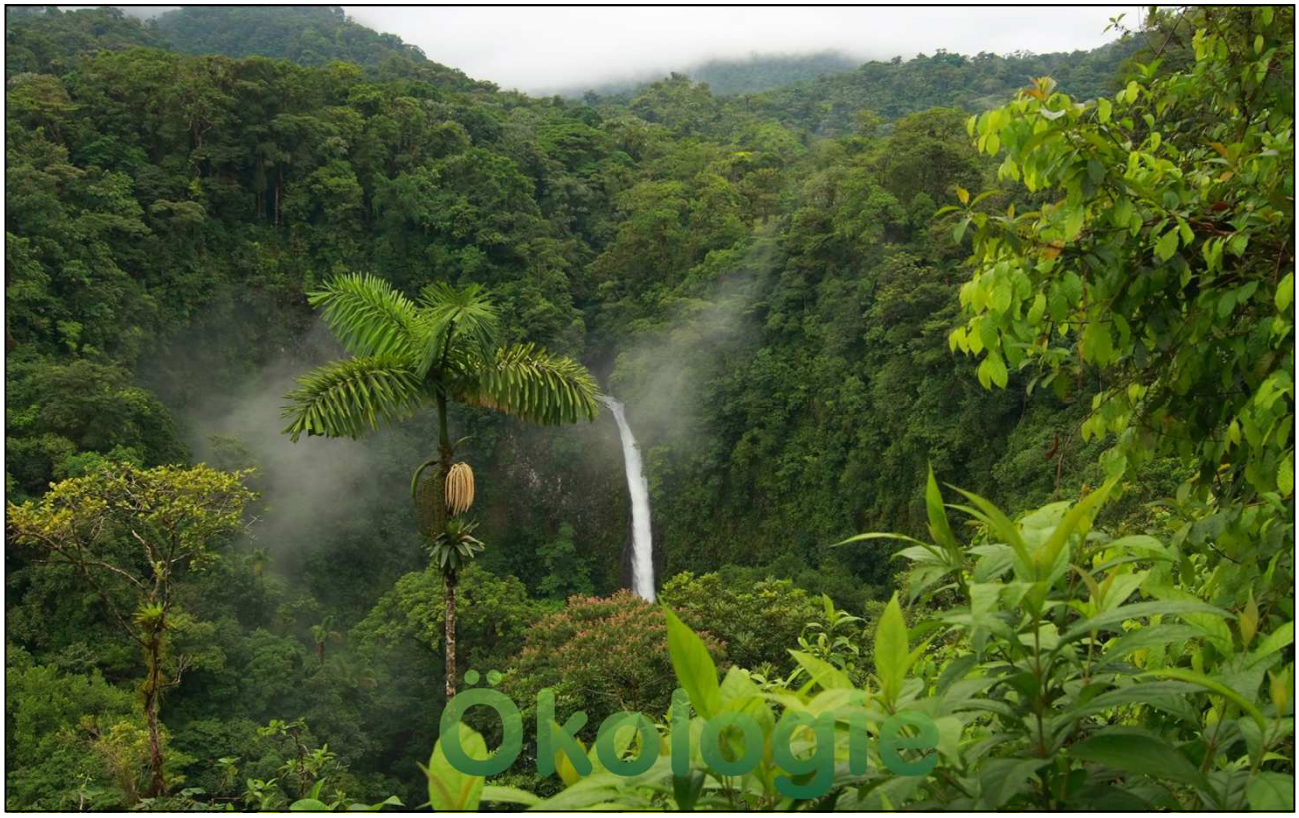
Hier gibt es heute **Strategie** „für CI“

- **Gegen „Wir brauchen das nicht“**
  - Von PL,DEV,Kunden

Nehmt die Argumente zu euren Teammitgliedern, Projektleitern, Kunden

Das Bild zeigt die Erfindung eines chinesischen Bauern, der seine Schweine jeden Tag springen läßt. Er sagt: „der Sprung stärkt das Immunsystem, macht sie hungriger, das Fleisch geschmackvoller.“

Steve Jobs: „stay hungry, stay foolish“

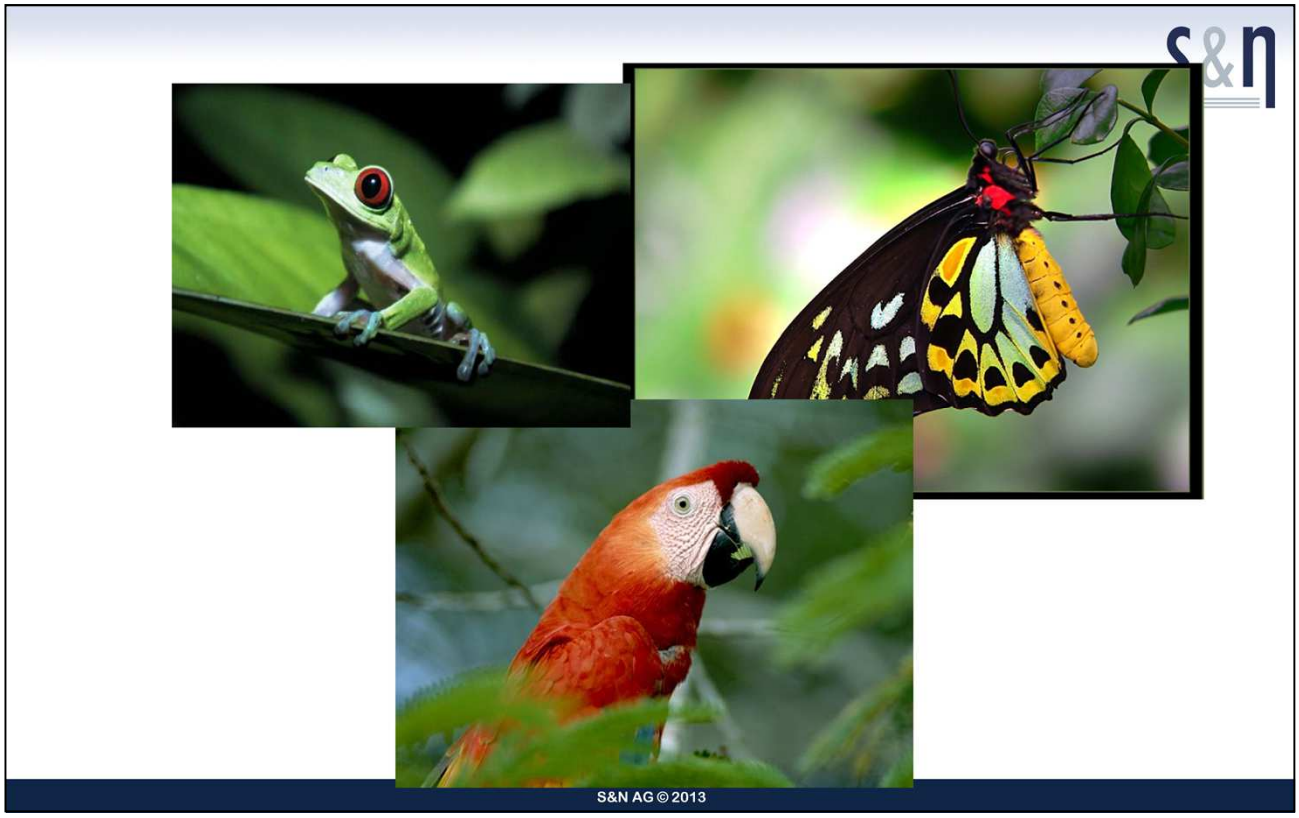


„Dieser Regenwald hat im Lauf der Evolution Mechanismen entwickelt, die eine unüberschaubare Vielzahl von Lebewesen. Tiere und Pflanzen teilen sich einen Lebensraum, zwischen ihnen bestehen Abhängigkeiten -> sie bilden ein Ökosystem.

Betrachten wir unsere Arbeitsumgebung ebenso als Ökosystem und beginnen, ökologische („den Haushalt betreffende“) Zusammenhänge zu begreifen.

Der CI Prozess und der Buildserver wollen als Bestandteile dieses Ökosystems gehegt und gepflegt werden!

Entwicklerinnen und Entwickler arbeiten zusammen nach ökologischen Gesetzen, sie haben Abhängigkeiten und Bedürfnisse. Betrachten wir sie nicht nur als Programmierknechte sondern als Beziehungsgeflecht, als Individuen.



In diesem Ökosystem gibt es Entwickler, Projektleiter, Kunden

Sie alle müssen gepflegt, ihre Wünsche und Erwartungen respektiert werden.
Welche Rolle spielt CI in diesem Ökosystem?

CI – eine Definition

- Häufig einchecken
- Dabei die Code-Basis nicht verschlechtern!

Versuchen wir eine minimale Definition

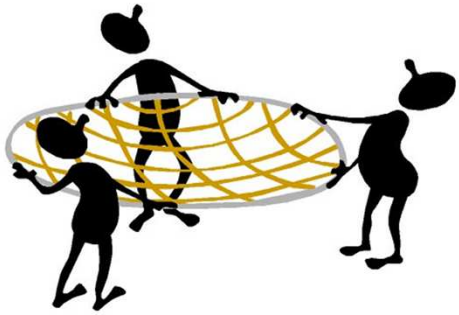
Nicht verschlechtern zieht hinter sich her, was man von CI kennt:

- Unit Tests
- Automatic Build vom Code Repository getriggert

CI entstammt einer Projektmethodik …

Historie

- Historie:
  - eXtreme Programming (1995)
    - „No code ownership“
    - Inkrementelle Änderungen für Features, Fixes, Refactoring.
    - Ständige Integration
  - Über dem Sicherheitsnetz der Tests



.. XP.

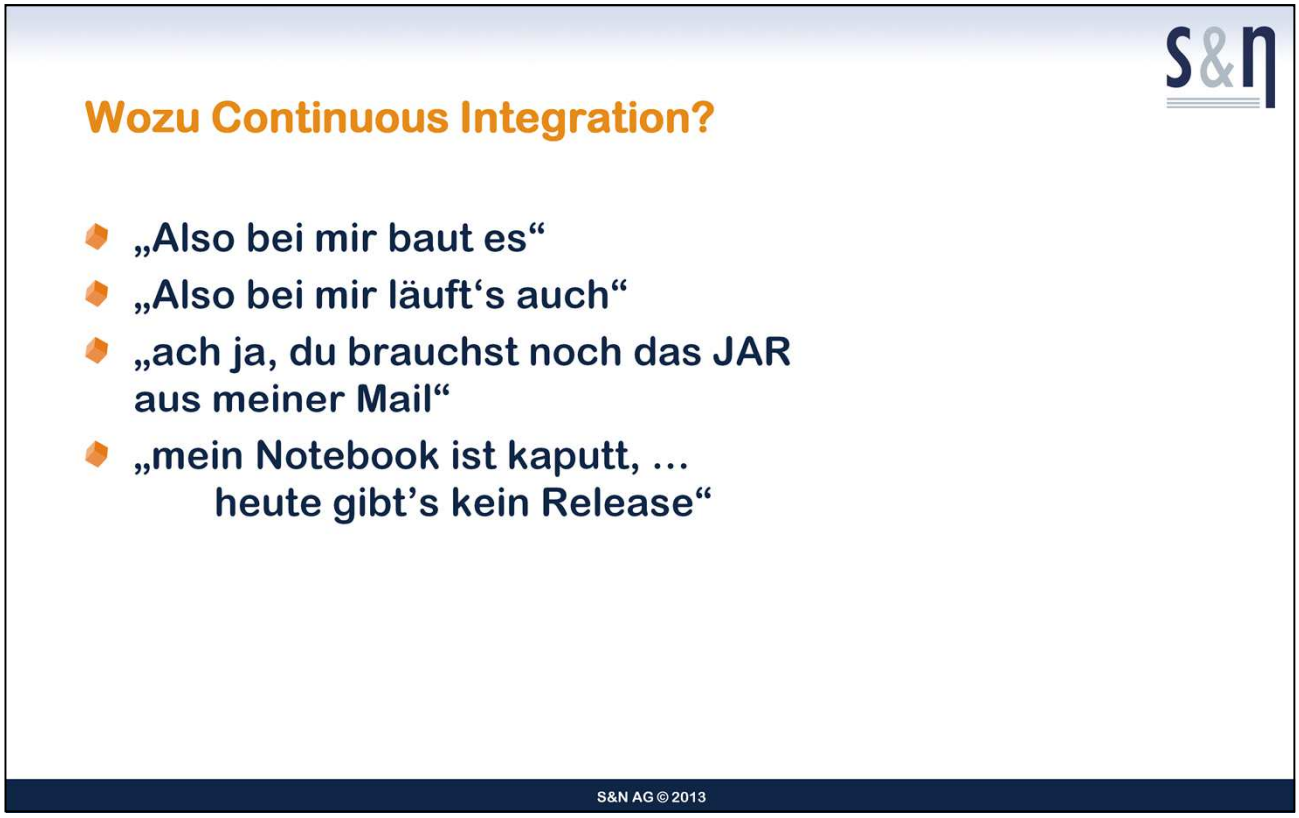
XP entschlackt vom Ballast traditioneller Modelle und führt dazu vor allem Viel-Testen ein

Häufige Inkremente in Iterationen sind ein Ziel.

Der Ausstoß aller Developer muss kurzfristig integriert werden.

Geschwindigkeit entsteht durch Verkleinern der Inkremente → das nennen wir Agilität

Nicht mehr: Integrationsphase als Meilenstein („ein Mal im Monat“)



CI löst außerdem diese Probleme, die wir aus dem Alltag kennen

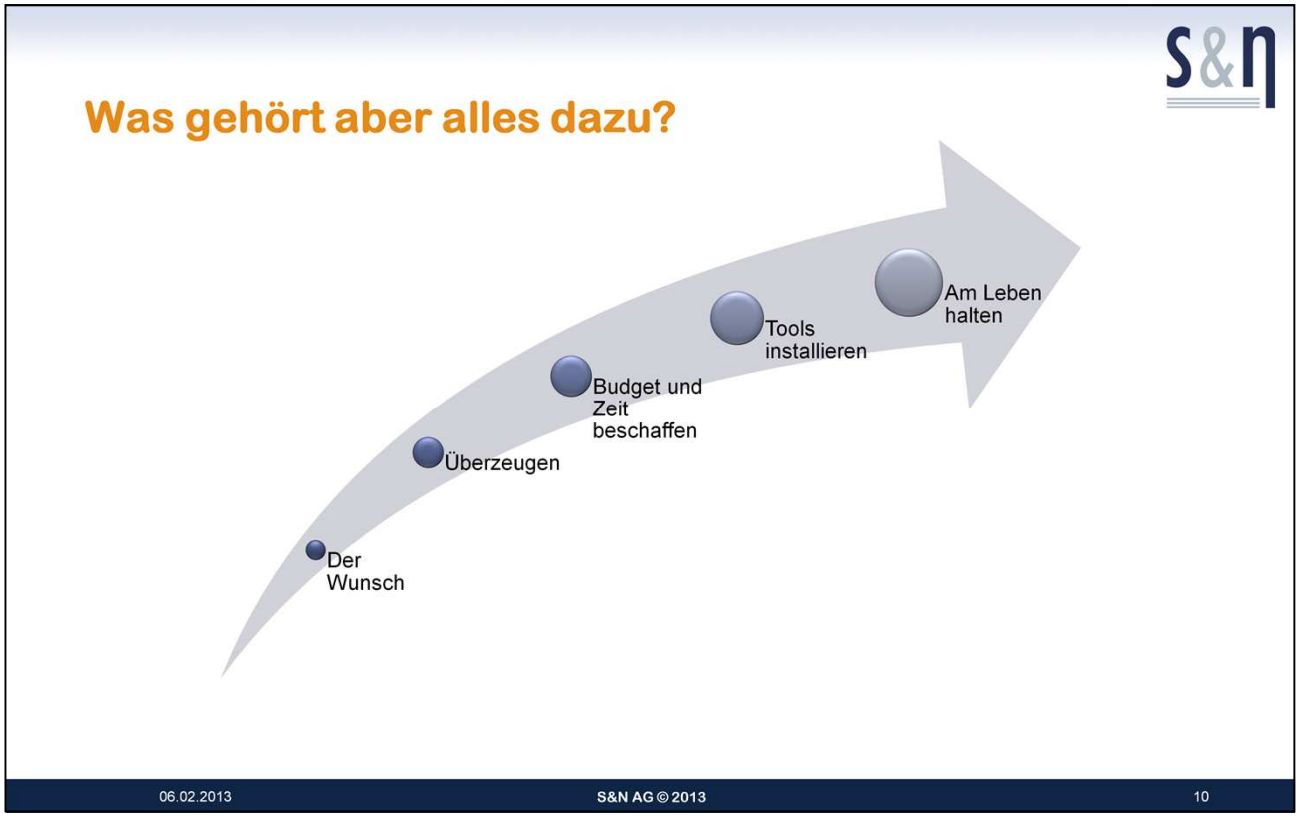
„Mein Notebook ist kaputt, heute gibt‘s kein Release..“

Gerade wenn ein AppServer im Projekt ist, können Konfigurationen von Entwicklerarbeitsplatz zu Arbeitsplatz unterschiedlich sein (ebenso: von gestern und heute)

Clearing-Stelle ist der Build-Server bzw. der Integrations-AppServer: da muss es bauen/deployen/laufen

Sicherstellen, das „unbeteiligte Dritte“ alles zum Bauen vorfinden (Situation des neuen Entwicklers im Team)

Was brauche ich um diese Probleme abzustellen ….



Am Anfang steht der Wunsch!

Der in einer Person entsteht.

Das sich mit dem weiteren Weg verbindet. Und das Engagement und Durchhaltevermögen hat, diesen zu beschreiten.

Auch am „Ende“ stellt man fest: es gibt kein Ende. Das Ökosystem muss stetig am Leben gehalten werden – weil es eben menschengemacht ist und nicht Natur.

Was macht den CI Prozess aus?

- Häufig integrieren
- auf Halteziele hinkodieren

- TDD definiert:
  kein erfolgreicher Test → kein Checkin!

- „Checkin dance“:
  Erst auschecken, dann einchecken

Test-driven development befördert das Ziel des Kodierens mit Halteziel

Häufiges Einchecken provoziert Konflikte, darum erst auschecken, integrieren (kann Zeit dauern), dann nochmals auschecken, wenn nichts mehr kam, einchecken.

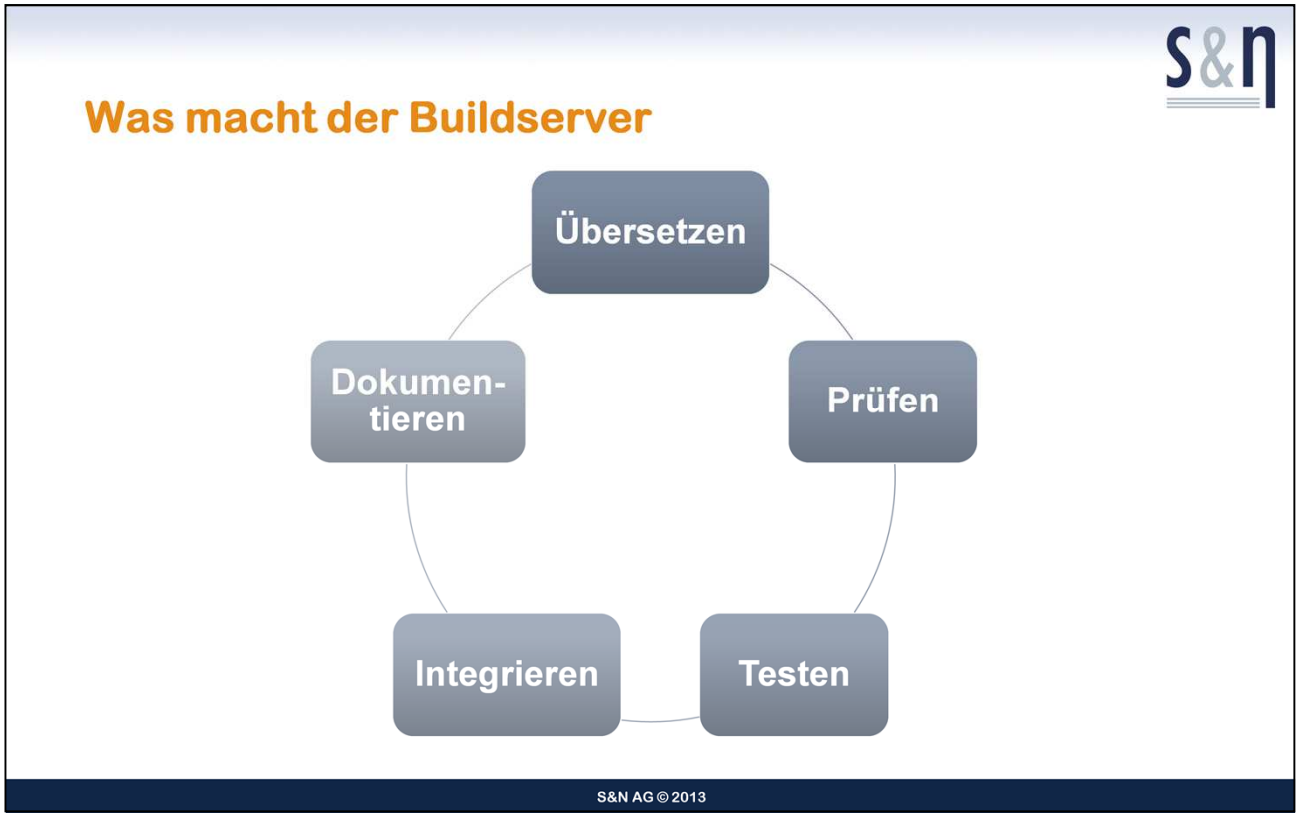
Ein Source code Repository ist mehr als nur ein Speicher (MKS)

Was passiert im Team:

„Alle liegen sich in dem Armen“,

Nicht ganz:

- Es wird mehr ein Geben und Nehmen
- Man rückt näher zusammen
- Man kommt sich näher, lernt sich kennen und (ein)schätzen
- ABER. Es entstehen auch Fliehkräfte, die das Team auseinandertreiben können → Hilfe naht …



Hilfe: der Buildserver

Buildserver als Teammitglied, 3. Mann beim Skat

Gegenargumente

GEFAHREN DURCH CI UND TESTS

Da man immer auf Gegenargumente stößt, lohnt sich die Vorbereitung („know your enemy“).

Also nun beliebte Gegenargumente

Warum man CI nicht einsetzen sollte!??

- „Mit einem Sicherheitsnetz werden Developer zu leichtsinnig.“
- „.. und Architekten erst recht“
- ?

Upfront-Design ist Merkmal traditioneller Modelle. Ist die Designphase erst mal verlassen, wird das Design nicht mehr angetastet

Rückblick auf die 90 Jahre (XP gab es ab 94)

Upfront bis zu einem gewissen Grade notwendig:
- nicht alle Features können beliebig spät ergänzt werden
- große/verteilte Teams müssen mit Arbeitspaketen bedient werden

PRD Project Requirements Document

Detailed Software Requirements Specification

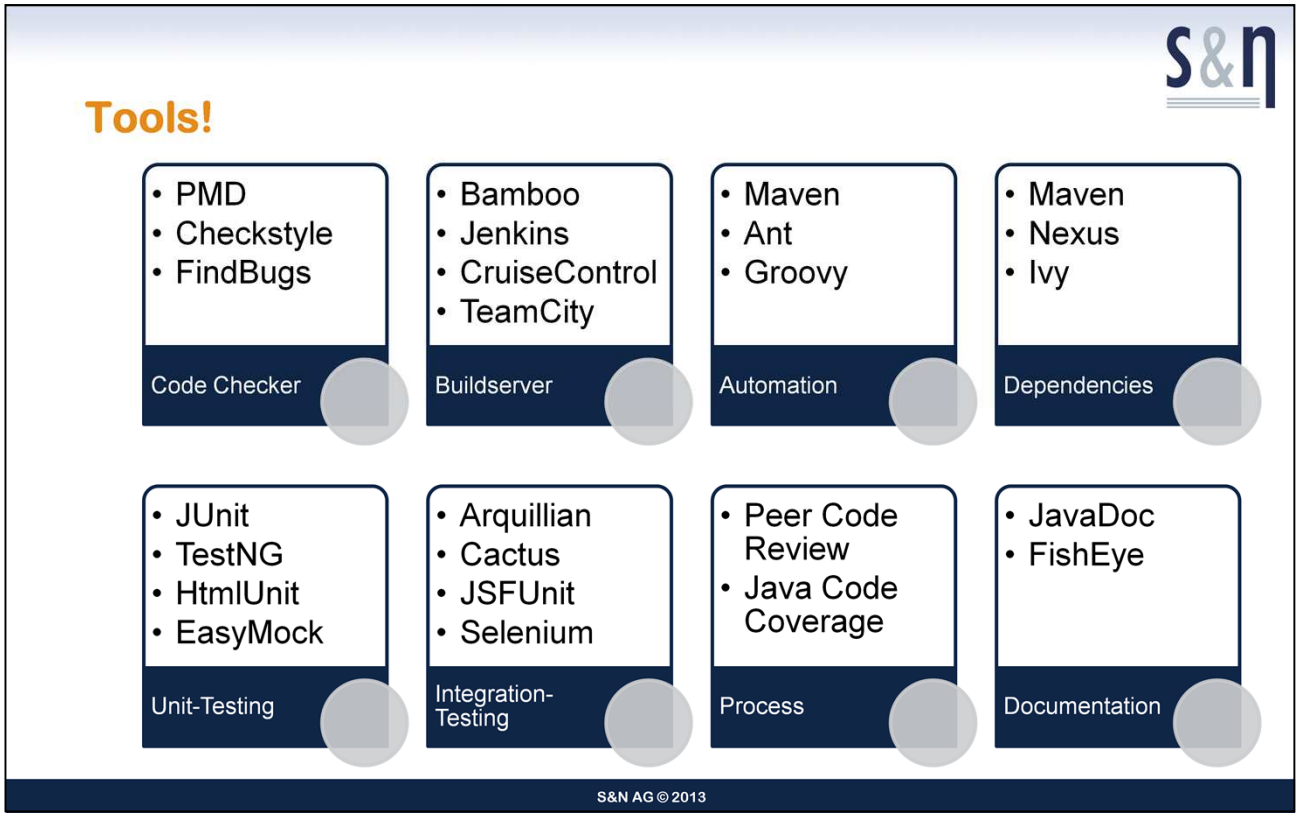
High-Level Design

Low-Level Design

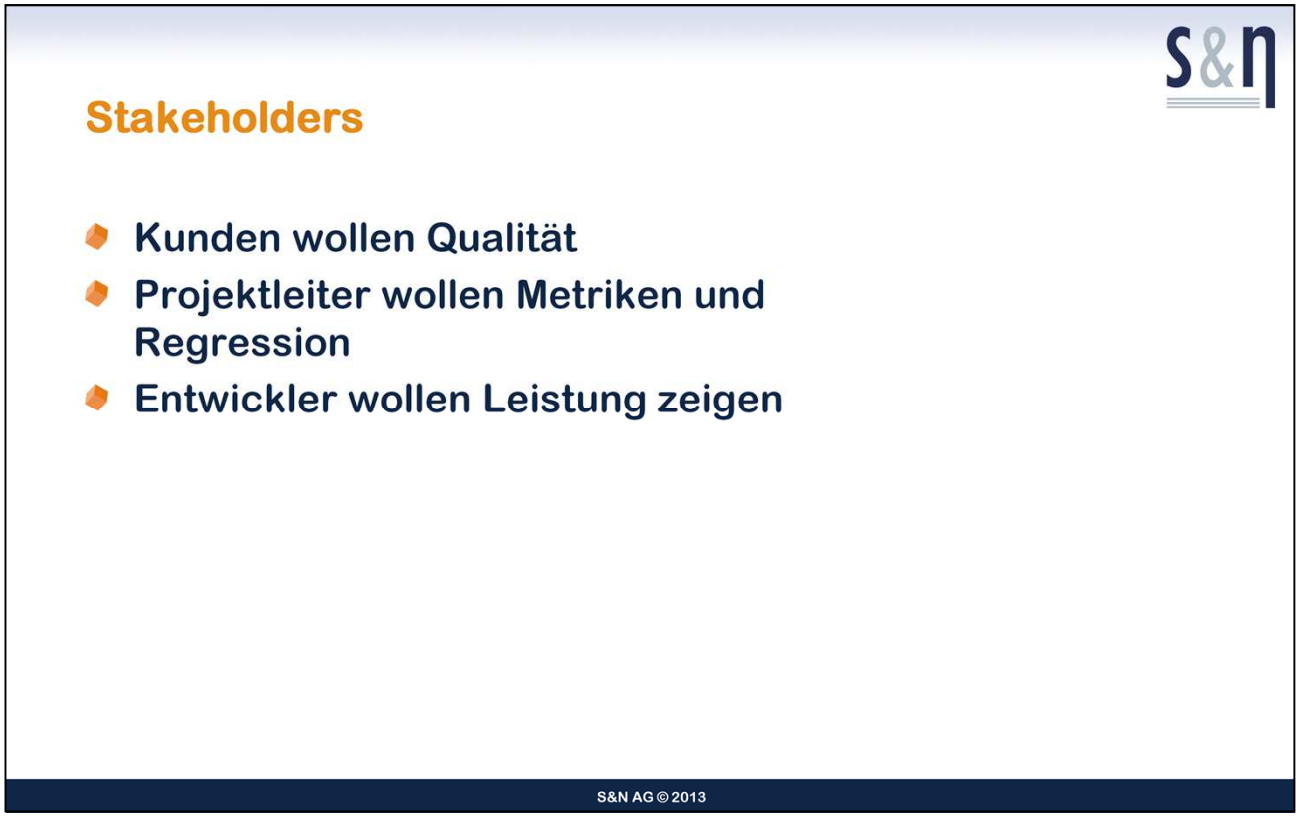
Teamteilung

- Horizontal: skalieren → erfordert Arbeitspakete
- Vertikal: Test-Team / QA-Abteilung
  - Wenn dann die Tests auch noch von einem anderen Team geschrieben werden, hat man auch gleich den Schuldigen:
- „war eben nicht genug Testabdeckung“

UND WENN MAN ES DOCH MACHT



Umfrage: wer kennt was, wer setzt was ein?



Das Problem, das jetzt angeleuchtet wird, beschreiben:

Dev: „Wieviel Zeit darf ich aufwenden?“

PL: „keine!“

Dev. „??“

PL: „mmmh, na gut, mach‘s kurz“

Am nächsten Tag:

PL: „wir haben noch diese Features aufgenommen, die müssen mit rein in das geplante Release“

Dev: „??“

PL: „!!“

Dev:“(mmh, dann schreib ich die Tests später…)“

Alle wollen Testen automatisieren

- Warum machen wir es dann nicht?

Olaf Port Armin Vogt Michael Illgner

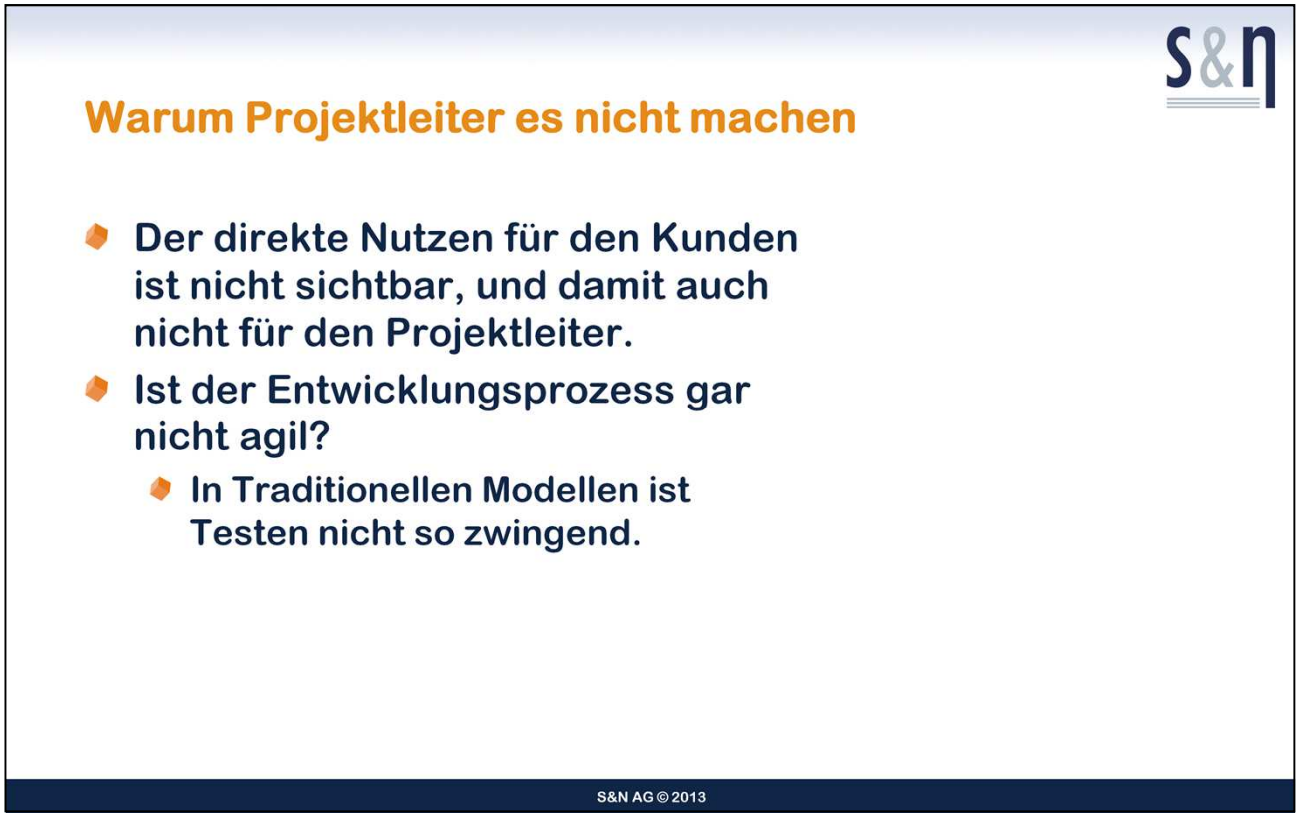
PODIUMSDISKUSSION!

Zusammenfassung

WORAN SCHEITERT CI

Warum Entwickler es nicht machen

- Tests werden als erstes vernachlässigt, wenn die Zeit knapp wird.
- Entwickler möchten lieber Features einbauen, denn Tests sind nicht sichtbar.
- Testbarer Code ist „schwierig“ zu schreiben



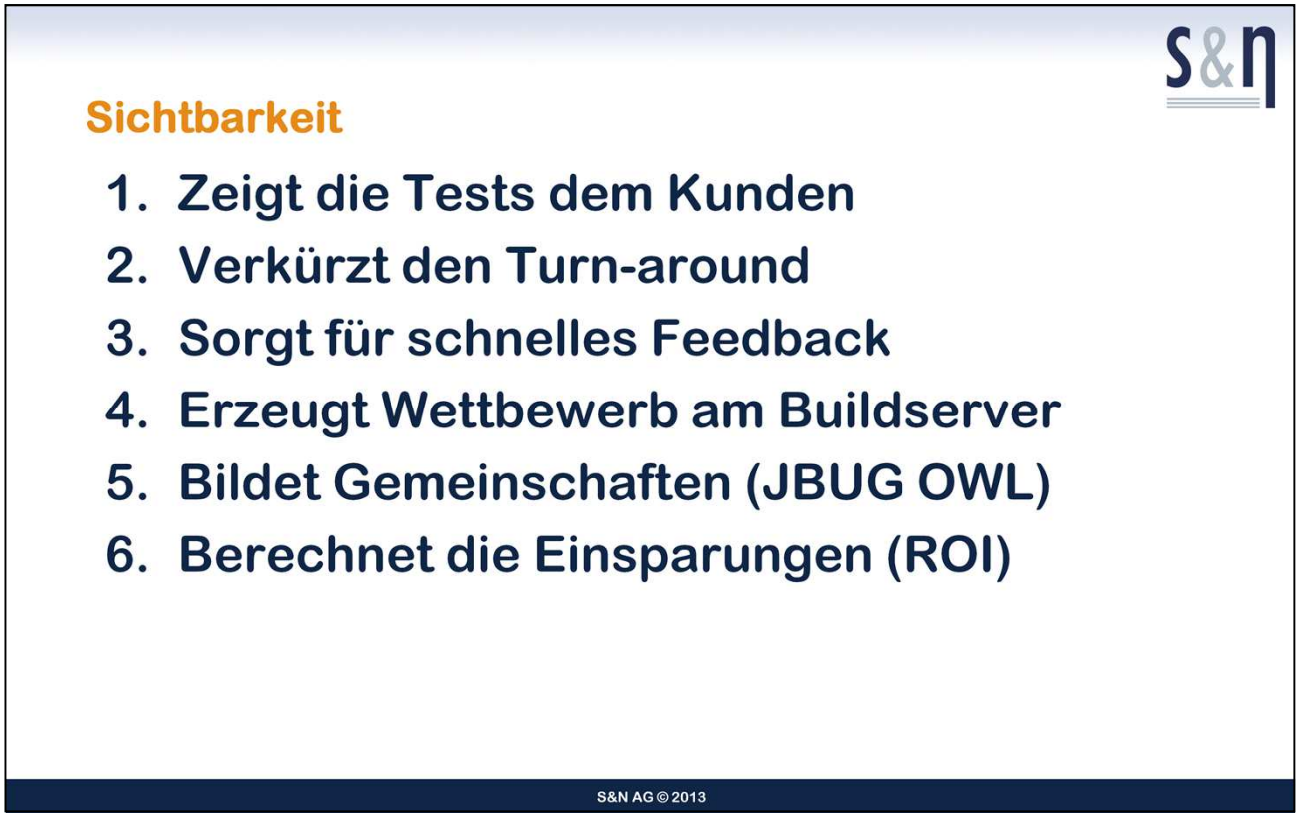
Agiler Prozess , aber Tests können vernachlässigt werden. Wirklich? Ist der Prozess agil, ist das Team agil??

Warum Kunden es nicht wollen

- Der Kunde sieht die Tests nicht und was sie bringen
- Tests erhöhen den Aufwand: „das muss ich dann alles zahlen“

Empfehlungen

TIPPS



Rückmeldungen von automatisierten Tests sind auch eine Form der Bestätigung

ROI=Return on Invest ist ein Totschlagsargument: sehr beliebt bei Entscheidern, aber sehr schwer zu berechnen, mit vielen Annahmen verbunden. Letzlich bleibt es ein Versprechen auf die Zukunft. Es hilft aber, sich selber Ziele zu setzen und das erwartete Verhalten durchzuspielen.

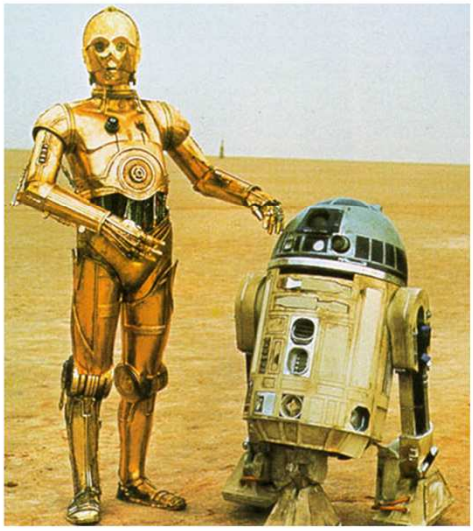
- Wo gewinne ich schnell
- Von welchen Maßnahmen erwarte ich welche Ergebnisse (messen!)

Wie wird das Team auf bestimmte Maßnahmen reagieren?

Gemeinschaften auch außerhalb → User groups für Tools, Prozessmodelle, → Jbug OWL

Pflege deine Freunde

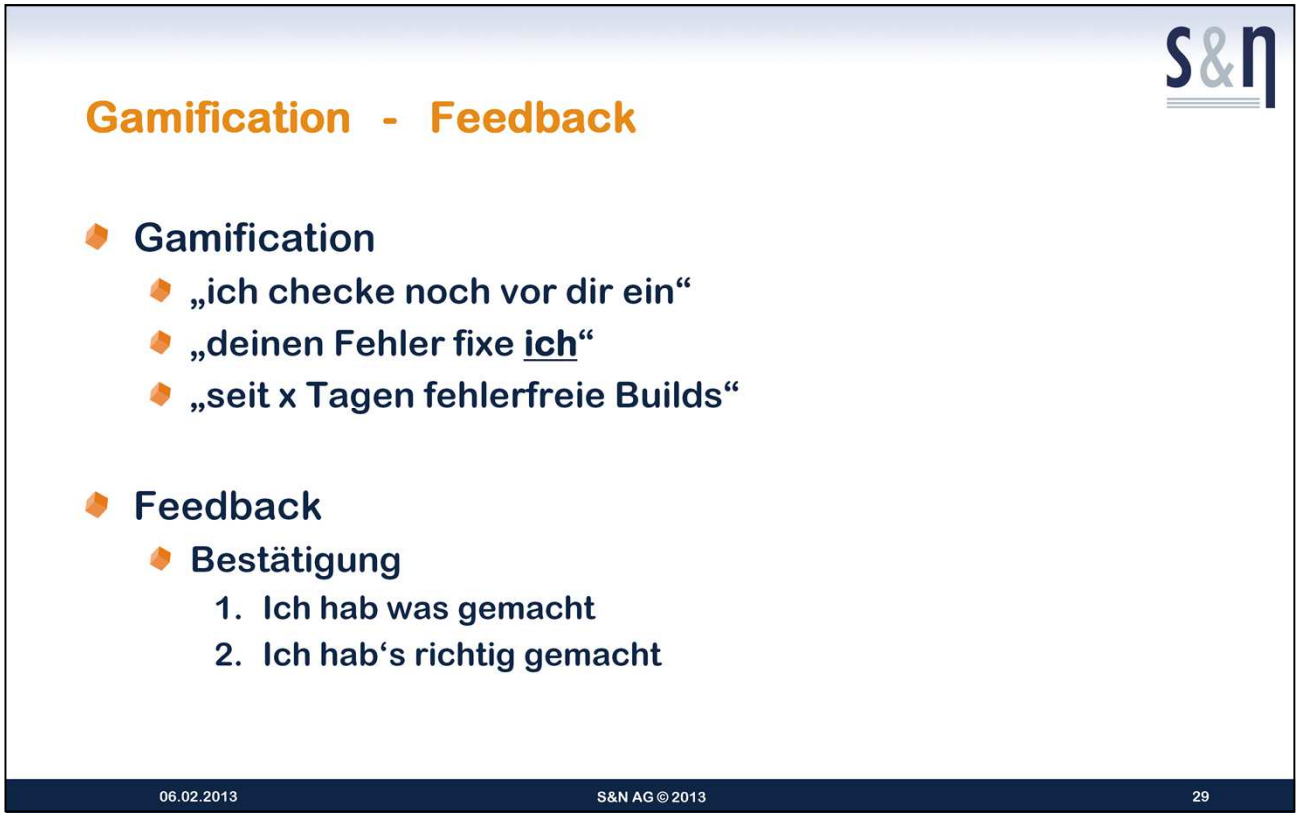
- Kämpft für euren Buildserver
  - Betriebsbereit halten!
  - Mails nicht ignorieren!
  - Argumente parat halten!



Kämpft für euren neugewonnenen Freund, den Buildserver.

Argumente muss man kennen, um die nächste Diskussion mit dem PL, dem Kollegen, dem Kunden, mit SICH SELBER !!! zu überstehen!

Wenn die Zeit/Budget mal wieder knapp wird, fallen die Tests eben als erstes hintenrunter



Vergleiche „Unfallfrei seit x Tagen“ im produzierenden Gewerbe

http://de.wikipedia.org/wiki/Gamification :

„Gamification bedient sich einiger Elemente. Im folgenden wird eine Abgrenzung und Klassifizierung der unterschiedlichen Elemente vorgenommen.

Sichtbarer Status

Einsehbare Rangliste

Quests

Resultatstransparenz

Rückmeldung

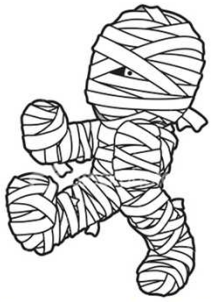
Epic Meaning

Fortschrittsanzeige

Community Collaboration

Cascading Information

Zusammenfassung



1. Die Tools sind da
2. Die Prozesse sind da
3. Gute Erfahrungen wurden gemacht
4. Dennoch: Jedes Projekt muss individuell entscheiden
5. Und: das Team muss diese Entscheidung gemeinsam tragen..

   - jeden Tag!
6. Also: stetige Motivation ist eine Aufgabe für alle im Team

Das Ökosystem muss gepflegt werden da es menschengemacht ist. Es wird bestimmt vom Zyklus des Zerfalls und Wachstums

Projekte sind individuell, WEIL ihre Teilnehmer Individuen sind. Die muss man abholen, überzeugen, einbeziehen, respektiern, von ihnen lernen/inspirieren lassen, Verantwortung übertragen.

Menschen wollen überzeugt werden – jeder einzelne.

Auf geht‘s

Armin Vogt
avogt@s-und-n.de