

Gerald Heller

**IN ETAPPEN ZUM ZIEL: INKREMENTELLE  
ENTWICKLUNG MIT HP QUALITY CENTER**

**DIE NUTZUNG DES RELEASE- UND CYCLE-MANAGEMENTS**

19. März 2009

Copyright © 2009 Gerald Heller

1

## Agenda

- Release und Cycle im Projektkontext
- Aufbau eines Releasebaumes
- Nutzung im Anforderungsmodul
- Nutzung für Tests
- Die Kunst des effektiven Filterns
- Reporting
- Endlich: „Pinned Tests“ in QC 10

19. März 2009

Copyright © 2009 Gerald Heller

2

## Begriff: Release

- Ein **Software Release** bezeichnet die zur Verfügungstellung einer bestimmten Konfiguration eines Software Produktes
- Die Erstellung eines Software Releases findet typischerweise innerhalb eines Projektes statt
- Die Zielgruppe kann intern oder extern sein

## Releaseinhalt

- Einem Software Release sind zugeordnet
  - Anforderungen
  - Spezifikationen
  - Lauffähige Software
  - Tests
  - Fehler
  - Dokumentation
  - ...

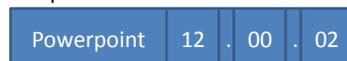
# Releasetaxonomie

- Ein Release wird näher festgelegt durch einen Releasebezeichner
- Elemente können sein
  - Namensbezeichner
  - Hauptversion
  - Unterversion
  - Patchversion
  - Servicepack
  - Kompilierversion
  - Betriebssystem
  - Hardwarearchitektur

## Beispiel: Aufbau eines Releasebezeichners

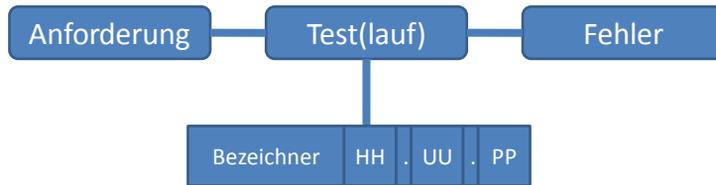


Beispiel



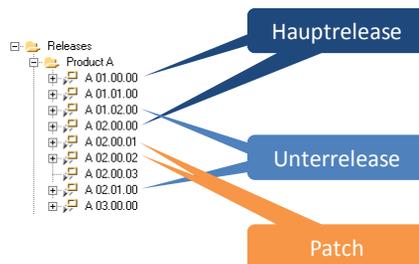
Entscheidend  
ist der Zweck  
des Ganzen!

# Was ist das Interesse?

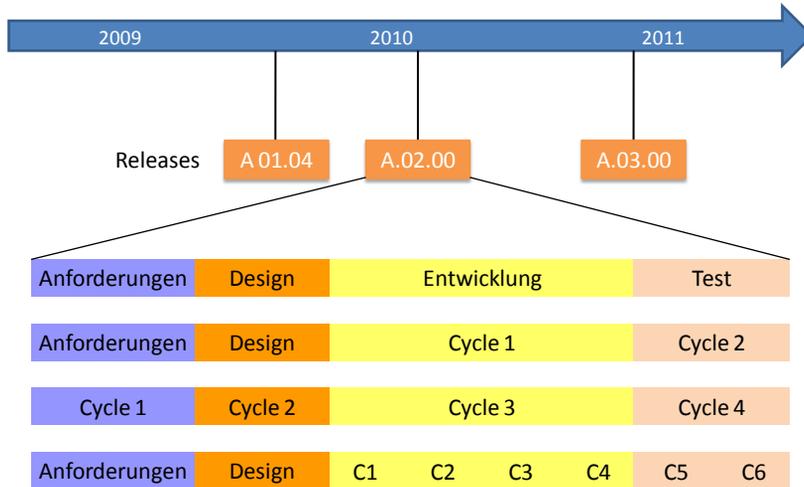


Aus Qualitätsperspektive ist es wichtig zu wissen, welche Qualitätsmaßnahmen für einen spezifischen Softwarestand vorgenommen wurden

# Beispiel: Ein Releasebaum in QC



# Projektlebenszyklen



19. März 2009

Copyright © 2009 Gerald Heller

9

# Quality Center Cycle

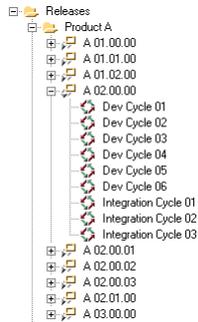
- Ein **Cycle** in Quality Center bezeichnet einen Zeitraum mit definiertem Anfang und Ende
- Einem Cycle können Quality Center Artefakte zugeordnet werden
  - Requirement
  - Testset
  - Defect

19. März 2009

Copyright © 2009 Gerald Heller

10

# Cycle Struktur

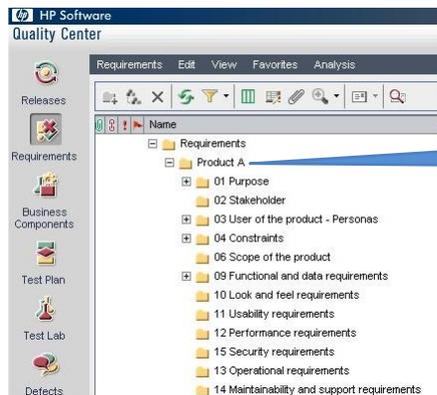


Je nach Einsatz können Cycles ausschließlich für Tests genutzt werden oder aber für alle Entwicklungsabschnitte

In diesem Beispiel wird das gesamte Produkt über mehrere Releases repräsentiert

Nicht jeder Cycle muss auch für QA relevant sein! (z.B. Dev Cycle 01)

# Anforderungsmodul



# Anforderungen sortiert nach cycle

Name	Status	Target Release	Target Cycle	Direct Cover Status
Requirements				-----
Product A				-----
User Stories				-----
User Story 1	07 Delivered	A.02.00.00	Dev Cycle 05	Failed
User Story 2	06 Validated	A.02.00.00	Dev Cycle 05	Failed
User Story 3	07 Delivered	A.02.00.00	Dev Cycle 05	Not Covered
User Story 4	06 Validated	A.02.00.00	Dev Cycle 05	Passed
User Story 5	07 Delivered	A.02.00.00	Dev Cycle 06	N/A
User Story 6	05 Implement...			Not Covered

- Anforderungen werden gemäß ihrer Businessfunktionalität im Baum angeordnet.
- Filtern erfolgt gemäß Cycle Wert
- Agile Teams bevorzugen Ordnerstrukturen gemäß den Cycles

# Testplanstruktur

Die Testplanstruktur orientiert sich an der Business- oder Architekturstruktur

**Test User Story 1**

Creation Date: 11/7/2007

Designer: gerald

Quality Aspects: Performance, Supp

Regression: Y

Status: Ready

Test ID: S

**Description**

This is a description of User story 1:  
 Joe the operator likes to get a bonus for excellent execution. He would like to perform the number one task in less than 5 minutes.

**Acceptance:**

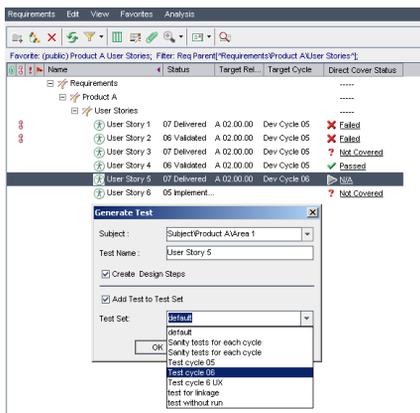
- Joe can do the job
- It takes him less than 5 minutes.
- The system is still in operation

# Tests und Cycles

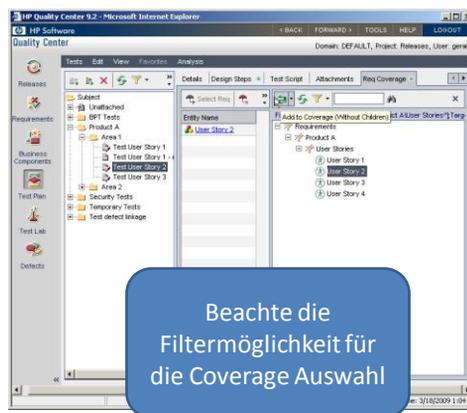
- Tests im Testplanmodul werden zu Anforderungen im Requirementsmodul verknüpft
- Damit erhalten sie automatisch eine Beziehung zu dem Cycle, welcher zur Anforderung verknüpft wurde

# Verknüpfung von Anforderung zu Test

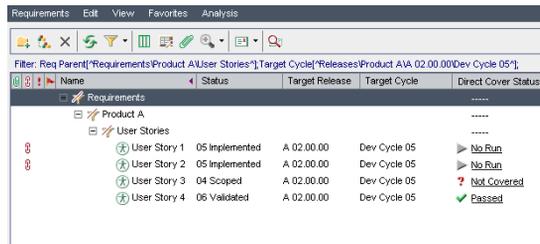
## A) Erzeuge Test aus Anforderung



## B) Erzeuge Test, verlinke zu Anforderung



# Coverage Berichte



The screenshot shows a software interface with a table of requirements. The table has columns for Name, Status, Target Release, Target Cycle, and Direct Cover Status. The data is as follows:

Name	Status	Target Release	Target Cycle	Direct Cover Status
Requirements				-----
Product A				-----
User Stories				-----
User Story 1	05 Implemented	A.02.00.00	Dev Cycle 05	No Run
User Story 2	05 Implemented	A.02.00.00	Dev Cycle 05	No Run
User Story 3	04 Scoped	A.02.00.00	Dev Cycle 05	Not Covered
User Story 4	06 Validated	A.02.00.00	Dev Cycle 05	Passed

- Stand der Testentwicklung
- Stand des Testens

# Nutzung von Cycles bei Fehlern

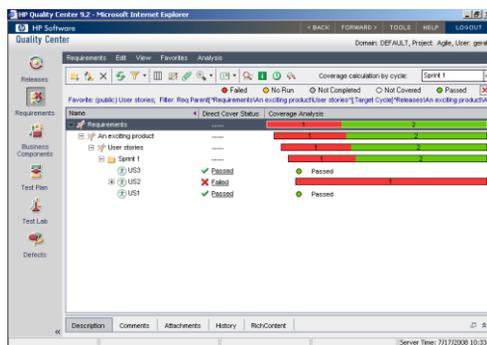
- Wenn während der Tests ein Fehler auftritt wird die Zuordnung zu dem Cycle automatisch eingefügt
- Zusätzlich empfiehlt es sich die Buildversion hinzuzufügen

# Die Kunst des effektiven Filterns

- Um qualitative Aussagen über Cycle coverage zu bekommen, ist es notwendig die Filter synchronisiert zu nutzen

# Cycles: Der erste Cycle

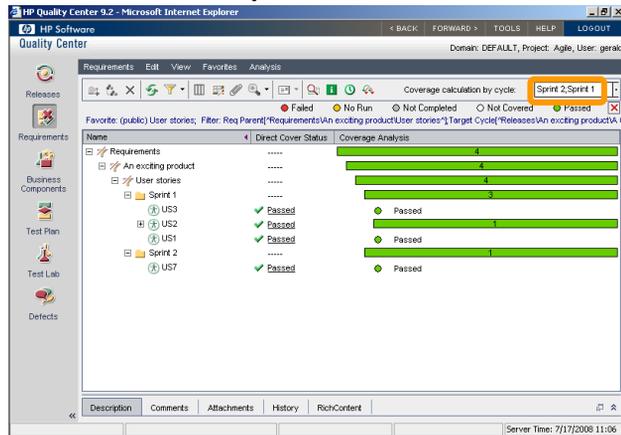
- Ergebnis nach dem ersten Cycle: Test für US2 nicht erfolgreich



# Cycles: Der zweite cycle

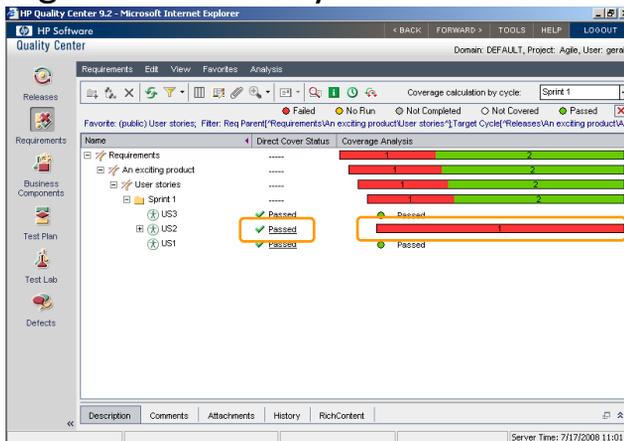
Ergebnis nach dem zweiten Cycle: Test für US2 ist nun o.k.

Die zwei Filter sind nun aufeinander abgestimmt: **Coverage und View Filter**



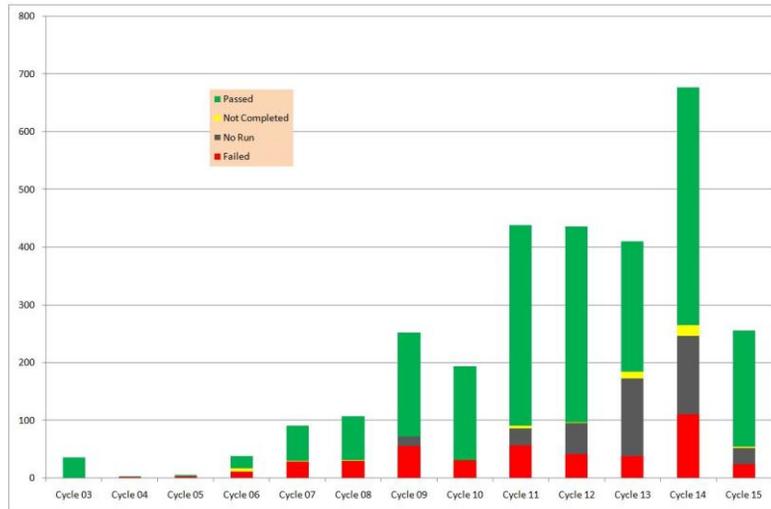
# Cycles: Rückblick auf cycle 1

Ergebnis nach 2. cycle: Ansehen von cycle 1



2 Filter müssen koordiniert sein: **Coverage und View Filter**

# Cycle Reporting



19. März 2009

Copyright © 2009 Gerald Heller

23

## QC 10 und pinned Tests

- Ausgangsszenario
  - Fixieren des Entwicklungsstand X samt Tests Auslieferung an Kunde
  - Weiterentwicklung des Produktes auf Stand Y
  - Problem bei X gefunden, Patch wird gebaut
  - Erneutes Testen von Entwicklungsstand X
  - Tests sollen nur einen Effekt auf Stand X haben

19. März 2009

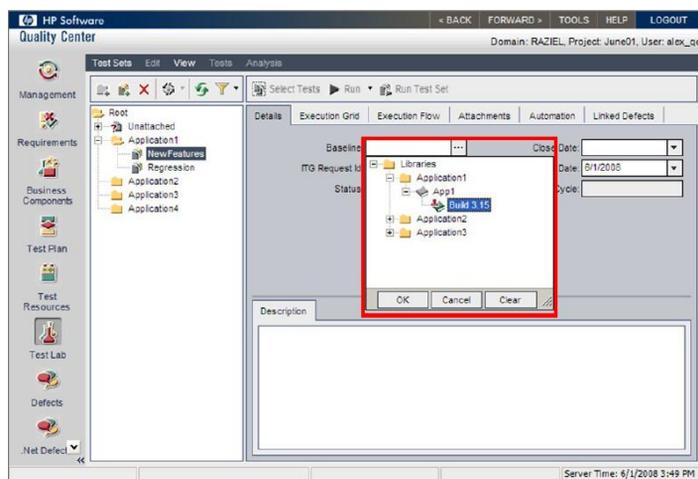
Copyright © 2009 Gerald Heller

24

## Pinned tests

- Ein Test-Set kann zu einer Baseline gebunden werden
- Beim erneuten Ausführen des Tests wird dann exakt die Version genommen, die zum Zeitpunkt der Baselineerstellung aktuell war
- Testläufe von pinned tests beeinflussen den aktuellen Stand nicht

## Pinned test GUI



Danke!

Fragen, Kommentare?



[gerald.heller@swq4all.de](mailto:gerald.heller@swq4all.de)