Qualitätssicherung bei Open Source Projekten

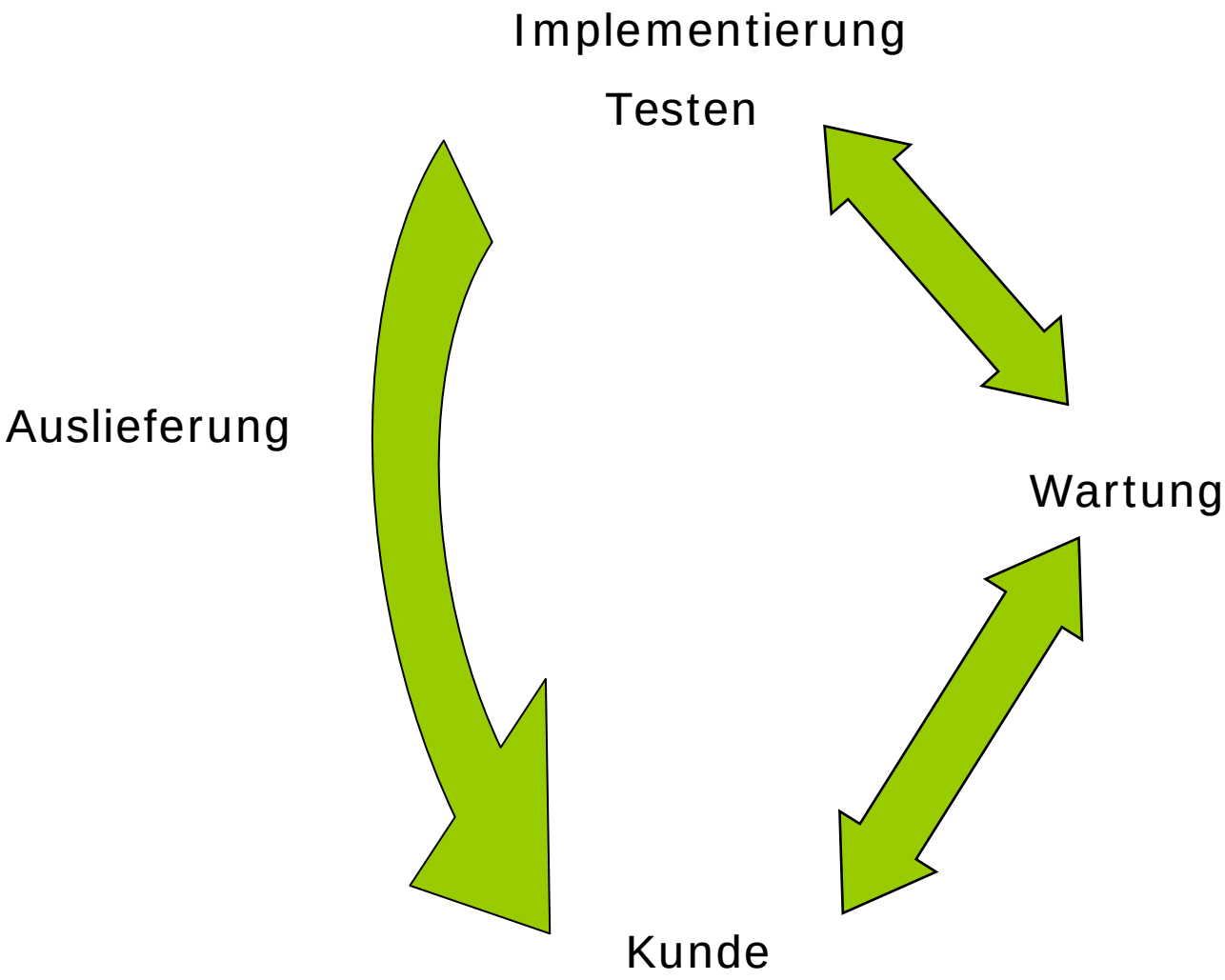
Stephanie Wilke
Institut für Informatik
FU Berlin
20.07.2006

Was ist Qualitätssicherung?

Qualitätssicherung ist der unternehmensinterne Prozess, der sicherstellen soll, dass ein hergestelltes Produkt ein festgelegtes Qualitätsniveau erreicht und behält.

- Überblick
  - Theorie
    - Sommerville
    - Balzert
  - Qualitätssicherung in Open Source Projekten
    - Komponententests
    - Integrationstests
    - Wie bekommt man Leute zum Testen
    - Tools

Qualitätssicherung im Softwareentwicklungszyklus



Wie Sommerville das Testen definiert

Freie Universität Berlin

- Verifikation „Erstellen wir das Produkt richtig?“

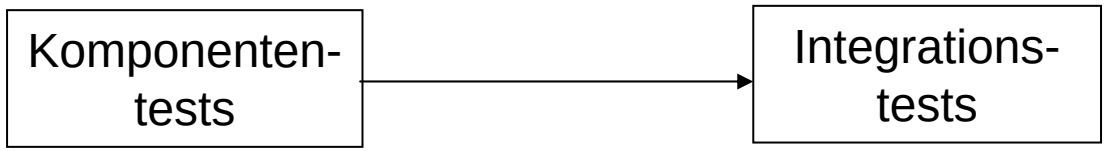
  Ziel: Ob die Software die Kundenerwartung erfüllt

- Validierung „Erstellen wir das richtige Produkt?“

  Ziel: Ob die Software ihre Spezifikation erfüllt

Freie Universität Berlin

- Testphasen:



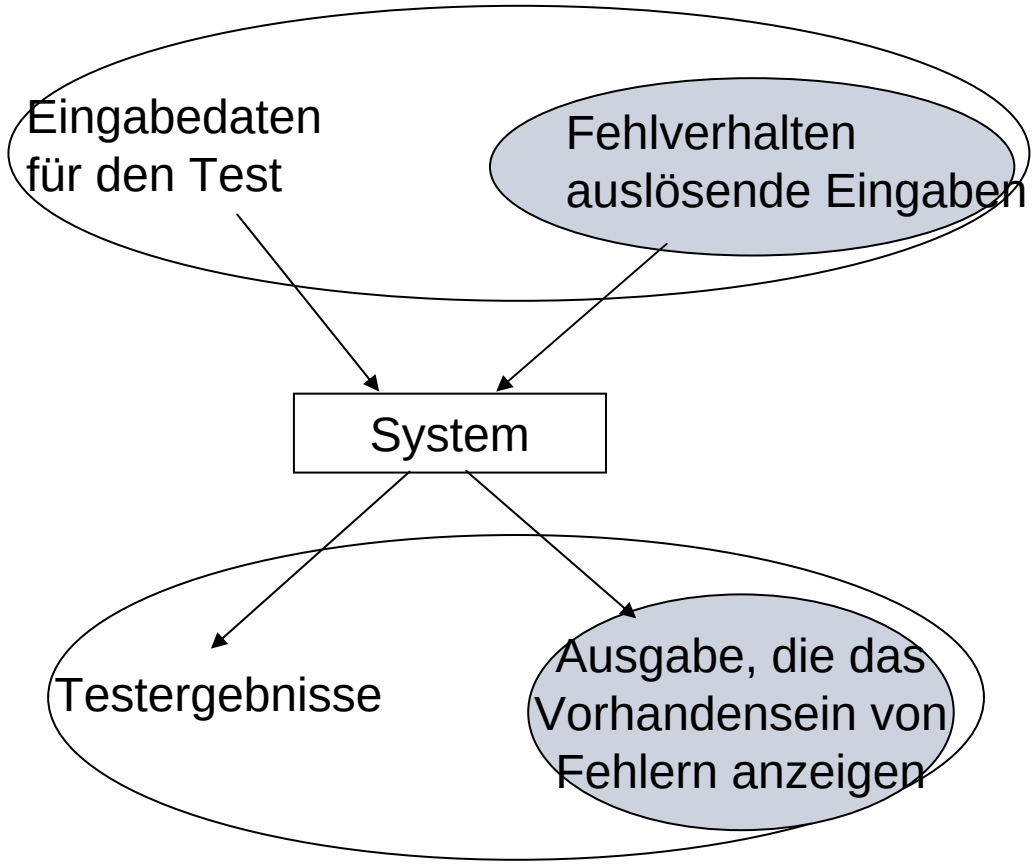
Softwareentwickler

- Blackbox-Test
- Whitebox-Test

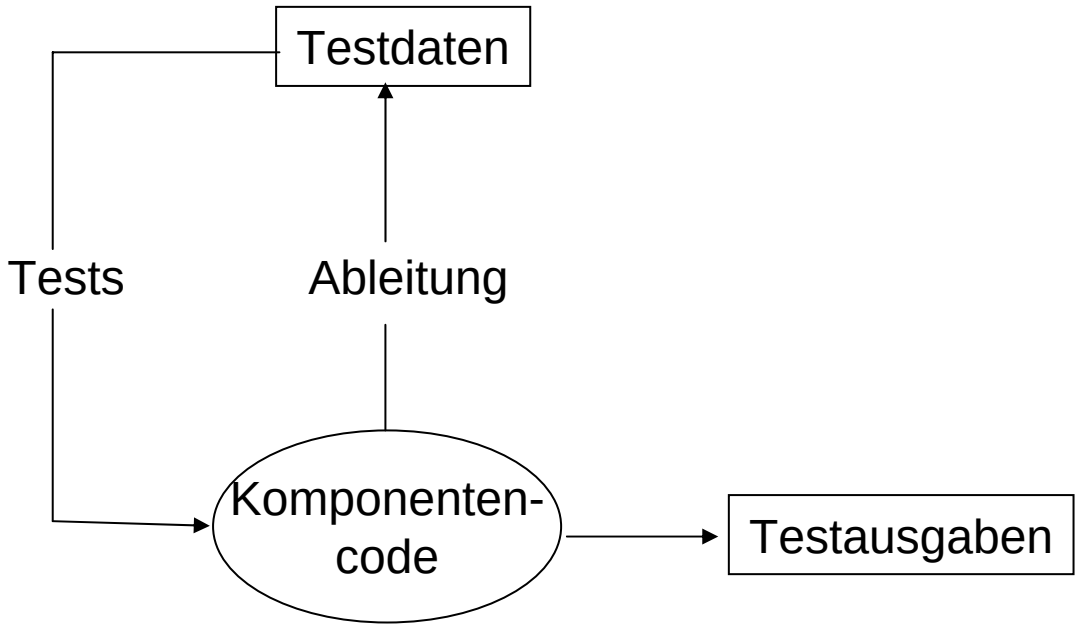
Unabhängiges Testteam

- Top-Down-Test
- Bottom-Up-Test

- Black-Box-Test (funktionale Tests)
  - Das Verhalten wird nur über die Eingabe und Ausgabe untersucht
  - Man beschäftigt sich nur mit der Funktionalität, nicht mit der Implementierung



- White-Box-Test (strukturelle Tests)
  - Die Tests werden aus der Struktur und Implementierung abgeleitet
  - Der Test liefert eine Analyse, die dazu dient, herauszufinden, welche und wie viele Testfälle man benötigt



Freie Universität Berlin

- Top-Down-Test
  - Erst werden die übergeordneten Komponenten integriert getestet, bevor ihre Implementierung beendet ist

- Bottom-Up-Test
  - Erst die Komponenten integrieren und testen
  - Dann übergeordnete Komponenten entwickeln

Wie Balzert das Testen definiert

- Drei Klassen von analytischen Verfahren zum Testen

1. Testende Verfahren
   - dynamische Tests, statische Tests
   - **Ziel**: Fehler zu erkennen

2. Verifizierende Verfahren
   - Verifikation, Symbolische Ausführung
   - **Ziel**: Korrektheit beweisen

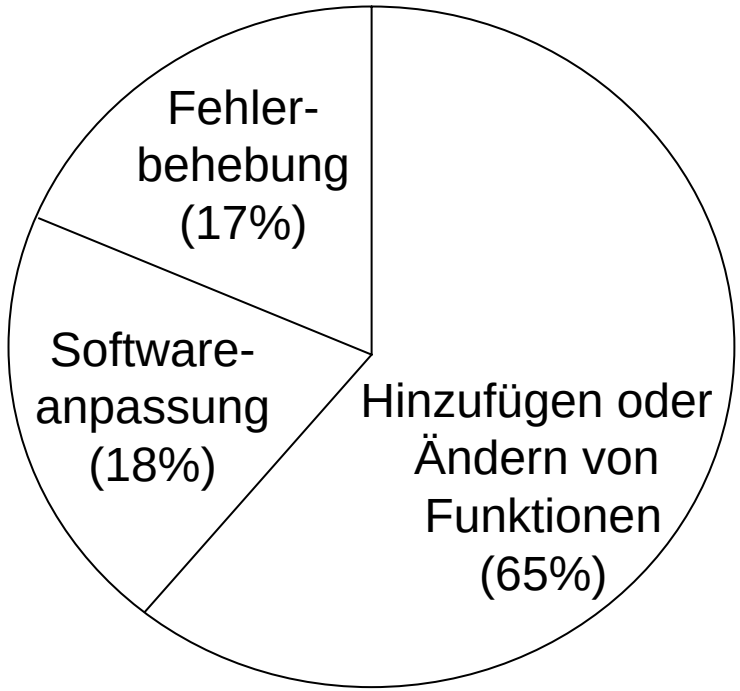
3. Analysierende Verfahren
   - Grafiken und Tabellen
   - **Ziel**: Darstellen der Eigenschaften von Komponenten

- Dynamische Tests
  - Wird mit konkreten Eingaben geteste
    - Black-Box-Test
    - White-Box-Test
    - Diversifizierender Test (vergleicht die Ergebnisse von verschiedenen Versionen)

- Statische Tests
  - Komponenten werden nicht ausgeführt, sondern analysiert
    - Inspektion
      - D.h. Personen durchsuchen ein Dokument um Anomalien und Fehlern zu finden ohne Programmausführung
    - Reviews (Bewertung)
    - Walkthroughs

Wie Sommerville die Wartung definiert

- Drei Arten der Softwarewartung
  1. Wartung zur Reparatur von Softwarefehlern
  2. Wartung zur Anpassung der Software an eine andere Betriebsumgebung
  3. Wartung zum Hinzufügen oder Ändern von Systemfunktionen



- Problem:
  - Wartung kostet viel Zeit (zwischen 50% und 75% der Zeit)
  - Wartung wird als zweitklassige Aufgabe angesehen
  - Keine Motivation der Programmierer, die Fehler beim Entwickeln zu finden, da die Entwickler meistens unter Zeitdruck arbeiten

- Zeitaufwändig
  - Änderungen werden vom Benutzer vorgeschlagen bzw moniert
  - Neue Version wird geplant
    - Anforderungsanalyse
    - Entscheiden, ob sich die Veränderung lohnt
    - Implementierung
  - Idealerweise sollte die Spezifikation angepasst werden

Wie Balzert die Wartung definiert

- Wartung

  beschäftigt sich mit der Lokalisierung und Behebung von Fehlerursachen, wenn Fehler bekannt sind

- Pflege

  beschäftigt sich mit der Lokalisierung und Durchführung von Änderungen und Erweiterungen

Freie Universität Berlin

- Drei Kategorien zur Wartung und Pflege

1. Korrigierende Aktivität
2. Anpassende Aktivität
3. Perfektionierende Aktivität

Qualitätssicherung in Open Source Projekten

Freie Universität Berlin

- Komponententests
- Integrationstests
- Wie bekommt man Leute zum Testen
- Tools

Freie Universität Berlin

- Bei klassischer Software Entwicklung hat Testen eine wichtige Rolle

- Bei Open Source Projekten spielt **formales** Testen nicht so eine große Rolle

- Validierung wird auf den Benutzer übertragen
  - Tester == Benutzer

[Q1]

Open Source und QM

Freie Universität Berlin

- **Komponententests**
- Integrationstests
- Zufriedenheit der Benutzer
- Wie bekommt man Leute zum Testen
- Tools

Freie Universität Berlin

- In einer Studie von Zhao und Elbaum wurde herausgefunden, dass
  - 68% der Entwickler macht Eingabe-Versuche, um Benutzer zu simulieren (Black Box Test)
  - 25% extreme Eingabewerte (Black Box Test)
  - 26% andere Validierungsmethoden
- Keine Aussage darüber, wer und wie viel getestet wurde

[Q1]

- Wichtige Säulen von Open Source Software und QM:

Peer Review

- Der Code wird über Mailingliste veröffentlicht
- So können die anderen Entwickler den Code bewerten
- Es können Vetos eingelegt werden
- Informelle Diskussion oder
- Manche Projekte haben formalisierte Vetos (z.B. Apache)

Beispiel: Apache Webserver
Open Source

- Der Entwickler testet seinen Code selber (Komponententests, ad-hoc testen)
- Veröffentlichung über Mailinglisten mit Peer Review
- Hier kann man um eine positive Bewertung bitten, d.h. drei Personen müssen mindestens den Code befürworten (hierbei sind keine Vetos erlaubt)

[Q2]

- **Verteiltes Debugging**
  - Voraussetzung
    - Viele Benutzer entwickeln selber
    - der Quellcode ist frei verfügbar
  - So tragen die Benutzer zur Fehlersuche, Fehlerbehebung sowie Programmverbesserung bei
  - Vorteile
    - Probleme können schneller als bei klassischer Software Entwicklung behoben werden
      - Der Benutzer kann das Problem selber lösen
      - Es gibt mehr Entwickler, die das Problem interessiert, da sie die Software selber benutzen. ***Die Stärke*** von Open Source Projekten

- **Verteiltes Debugging**

  - Probleme:
    - Es gibt keine Daten darüber, wie viele den Code schon bearbeitet haben
    - Der Benutzer muss sich den Code runterladen
    - Den Code richtig verstehen
    - Das Problem lösen können
    - Der Benutzer muss vertrauenswürdig erscheinen, so dass seine Lösung übernommen wird

Open Source und QM

Freie Universität Berlin

- Komponententests
- **Integrationstests**
- Wie bekommt man Leute zum Testen
- Tools

- **Power-User**, die Integrationstests durchführen
  - Power-User verwenden die neuste Version
  - Geben gerne und häufig Feedback
  - wie z.B. bei
    - Linux Kernel
    - Mozilla Webbrowser

[Q5]

- Linux Kernel
  - Wird meistens nach Implementierung in einer Mailingliste für die Besprechung veröffentlicht
  - Meinungen dazu werden ebenfalls über die Mailingliste verschickt
    - Also werden Verifikation und Validierung von der Gemeinschaft durchgeführt

- Mozilla Webbrowser
  - Es gibt ein Test-Team, das Integrationstests durchführt
  - Dieses Team ist noch aus der Zeit von Netscape
    - Wie und ob das Team bezahlt wird ist nicht klar

Open Source und QM

Freie Universität Berlin

- Komponententests
- Integrationstests
- **Wie bekommt man Leute zum Testen**
- Tools

Wie bekommt man die Leute zum Testen?

Freie Universität Berlin

- **Zusammengehörigkeitsgefühl**
  - Linux z.B.
    - belohnt seine Benutzer mit Gewissheit, einen Teil einer lohnenden Sache zu sein und
    - regelmäßig mit Verbesserungen versorgt zu werden, an denen man seinen Anteil hat
  - Linux verhält sich nach folgenden Grundsatz:

    Kann man auf genügend große Anzahl von Mitentwicklern zählen, lässt sich praktisch jedes Problem schnell charakterisieren und die Lösung ist einen der Beteiligten offensichtlich. (Linus Law)
  - Vermutung: nur 10% - 20% der Benutzer testen und geben Feedback

[Q7]

Open Source und QM

Freie Universität Berlin

- Komponententests
- Integrationstests
- Zufriedenheit der Benutzer
- Wie bekommt man Leute zum Testen
- **Tools**

Freie Universität Berlin

- Fehler werden zum Teil mit **Bug-Tracker** behoben
  - Benutzer suchen nach interessanten Problemen
  - Finden Code, der bearbeitet werden muss
  - Bearbeiten den Code
  - Veröffentlichen den Code
  - Und warten auf Feedback

Beispiel: Bugzilla bei Mozilla Webbrowser

Open Source

Freie Universität Berlin

- Bugzilla ist eine Datenbank, in der alle veröffentlichten noch vorhandenen Fehler aufgelistet sind
- Jeder Fehler bekommt eine Nummer und eine Kurzbeschriftung des Fehlers und Fehlerszenarien
  - Kurzbeschriftung und Szenarien müssen nicht geschrieben werden
- Jeder Fehler hat einen Status der Bearbeitung
- Zusätzlich hat Bugzilla die Möglichkeit, Wünsche für zukünftige Funktionen anzubringen

[Q5]

Bug List

Bugs on this list are sorted by relevance, with the most relevant bugs at the top. Only the 200 most relevant bugs are shown.

Mon Jul 17 2006 14:32:23 PDT

200 bugs found.

ID	Sev	Pri	Plt	Assignee	Status	Resolution	Summary
278260	nor	--	PC	sspitzer@mozilla.org	NEW		HTML signatures can add HTML headers in the middle of bod...
227872	min	--	PC	sspitzer@mozilla.org	NEW		Save message with HTML attachment as HTML results in inva...
232358	nor	--	PC	english-us@evangelism.bugs	NEW		matcheroo.com displays HTML code instead of proper HTML l...
214091	nor	--	PC	english-us@evangelism.bugs	NEW		npaci.edu - Fails to displasy proper html - IE 5/6 Opera ...
260078	nor	--	PC	mscott@mozilla.org	NEW		Escaped HTML brackets in Atom feeds display as raw HTML i...
344665	nor	--	PC	morgamic@gmail.com	NEW		Admintree html output is a bit quirky <HTML>\n<HTML>
206522	nor	--	PC	mail@seamonkey.bugs	NEW		html message shows as source, multiple content types (one...
157152	nor	--	All	dwitte@stanford.edu	ASSI		read-only/hidden bookmarks.html (or bookmarks.html.moztmp...
282476	nor	--	PC	mail@seamonkey.bugs	NEW		Simple HTML content moved out of the div.moz-text-html co...
142149	enh	--	PC	ducarroz@ducarroz.org	NEW		Simple HTML mode should give option to view current mail ...
229528	nor	--	PC	bugs@bengoodger.com	NEW		html URL (html:a) in <description> tags do not work corr...
306303	enh	--	All	mscott@mozilla.org	UNCO		Automatically use / allow HTML formatting when replying t...
222746	enh	--	All	sspitzer@mozilla.org	NEW		Change "HTML" to "HTML (rich-text)"
241191	nor	--	PC	mozeditor@floppymoose.com	UNCO		midas - insert html command (cmd_insertHTML) should sele...
240848	enh	--	All	parser@content.bugs	NEW		Option to use HTML Sanitizer (Simple HTML) in browser
189311	nor	--	All	ben.bucksch@beonex.com	NEW		HTML Sanitizer (Simple HTML) breaks Mailnews' own stylesheet
132421	nor	--	All	cmanske@jivamedia.com	ASSI		publish with a filename w/o the .html extension throws HT...
246668	nor	--	PC	composer@editor.bugs	NEW		HTML Source mode blows away lang attribute on root element
324954	nor	--	PC	nobody@mozilla.org	NEW		HTML Reply to .EML/rfc822 attachment without Message-ID r...
202437	nor	--	All	mozeditor@floppymoose.com	NEW		When opening HTML documents in editor, do a (potentially ...
247313	nor	--	PC	sspitzer@mozilla.org	NEW		HTML being changed when replying to Outlook HTML email co...
341869	nor	--	All	ui@bugzilla.bugs	NEW		Fix usage of HTML header tags in template/en/default/glob...
154379	enh	--	All	composer@editor.bugs	NEW		RFE: customize the HTML formatting
258541	nor	--	All	bug@annevankesteren.nl	NEW		news.html is not valid html 4.01 strict

Freie Universität Berlin

- **Stanford Checker**
  - Systematischer Fehlerscanner für Open Source Quellcode
  - Findet Fehler wie z.B. Pufferüberläufe und andere kritische Bugs
  - In einer Datenbank werden alle gefundene Fehler aufgelistet

  - Auftraggeber: Departement for Homeland Security (DHS)
  - Zusammenarbeit von Standford Universität und Sicherheitsfirma Coverity
  - Ist eine frei verfügbare Software
  - DHS bezahlte 1,24 Mio $ für die Arbeit der nächsten drei Jahre

[Q10]

Standford Checker
Open Source Tool

- Bis jetzt wird das Tool unter anderem von Linux Kernel, Mozilla Webbrowser, MySQL und Apache Webserver benutzt

- Beispiel von Linux-Kernel:
  - In 5,7 Millionen Programmzeilen
  - 985 Bugs gefunden

  - In kommerziellen Programmen werden durchschnittlich 5.000 Bugs gefunden

[Q10]

Zusammenfassung I

- Validierung wird auf den Benutzer übertragen

- Komponententests
  - Black Box Test, den der Entwickler durchführt
  - Peer Review mit Veto einlegen
  - Verteiltes Debugging, Benutzer finden Fehler und können sie lösen

- Integrationstests
  - Power-User, die Feedback geben
  - Test-Teams

Zusammenfassung II

- Durch das Gemeinschaftsgefühl bekommt man die Leute zum testen

- Tools
  - Bug-Tracker, Benutzer suchen interessante Probleme
  - Standford Checker, eine Software die Fehler findet

=> Die Qualitätssicherung von Open Source Projekten kommt dem Ansatz von Sommerville am nächsten

Vielen Dank!

Quellen

- [Q1] „Quality assurance under the open source development model“ von Zhao und Elbaum
- [Q2] „Two Case Studies of Open Source Software Development: Apache and Mozilla“ von Mockus
- [Q3] „Open Source Jahrbuch 2006“
- [Q4] „Open Source Jahrbuch 2005“
- [Q5] „Prozesse in Open Source Projekten“ von Sebastian Charlet
- [Q6] „Qualitätssicherung bei Open Source Projekten“ von Istvan Bartkowiak
- [Q7] „The Cathedral and the Bazaar“ von Eric Steven Raymond
- [Q8] „Software Engineering“ von Ian Sommerville
- [Q9] „Lehrbuch der Software-Technik“ von Helmut Balzert
- [Q10] Pro-Linux