FirstSpirit™

Your Content Integration Platform

Qualitätssicherung für komplexe Software in der Praxis

Jörn Bodemann, Vorstand e-Spirit AG

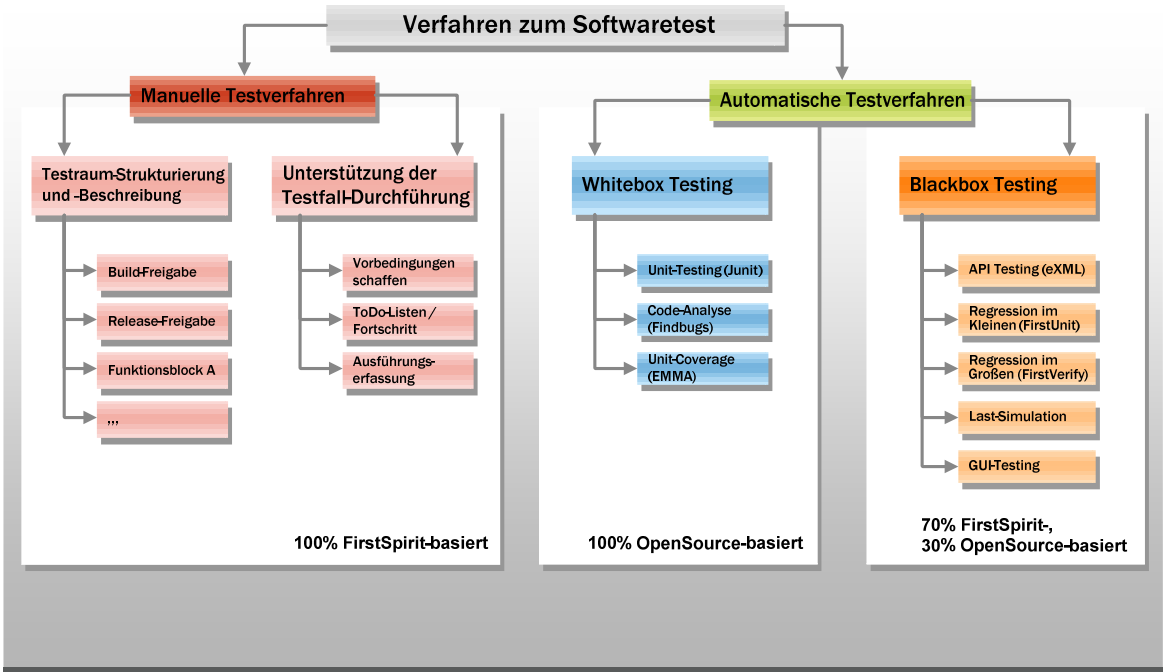
FirstSpirit{QA

Your Content Integration Platform

Das Produkt FirstSpirit

- **Enterprise Content Management mit Schwerpunkt im HighEnd-Segment**
- **Ein paar Kennzahlen (Stand 08-2008):**
  - 100% Java, Entwicklung seit 2001, ~ **45-50 Personenjahre** Implementierung
  - **Lines of code: 563.940** (Core-Product), Number of classes: 12.049
  - **Lines of test code: 90.346**
  - Zwei-Client-System (Java-Client und Web-Client)
  - Erfolgreiche Generalüberholung des Backends (2006/2007) ==> V4.0
  - Generalüberholung der Frontends (2008/2009) ==> V5.0 (in Arbeit)
- **Testraum:**
  - JDKs: SUN 1.5_*, 1.6_*, IBM 1.5_* und 1.6_* (etwa 20 relevante)
  - Betriebssysteme: Windows 2003 / Linux / Solaris Sprac & x86 / AIX (je 32- und 64-Bit)
  - Datenbanken: Oracle / MS-SQL 2005 / DB2 / PostgreSQL / …

  ➔ **20 x 10 x 7 = 1400 Kombinationen (grobe Abschätzung)**



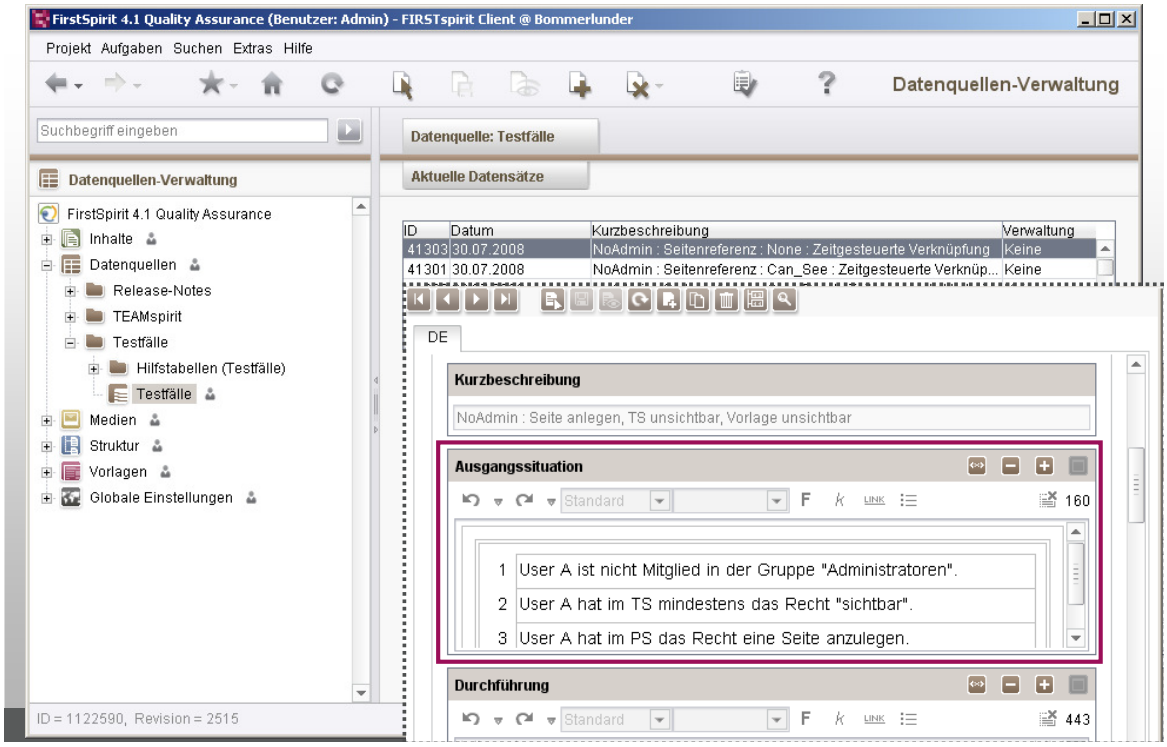
Manuelle Testverfahren

- **In vielen Bereichen leider immer noch unumgänglich**
  - Testen von Installations-Routinen (unterschiedliche Benutzerrechte, Lokalisation)
  - Grafische Darstellung komplexer GUIs (z.B. Redraw- / Antwort-Verhalten)
  - Lokalisation == Layoutänderungen durch unterschiedliche Lauflänge

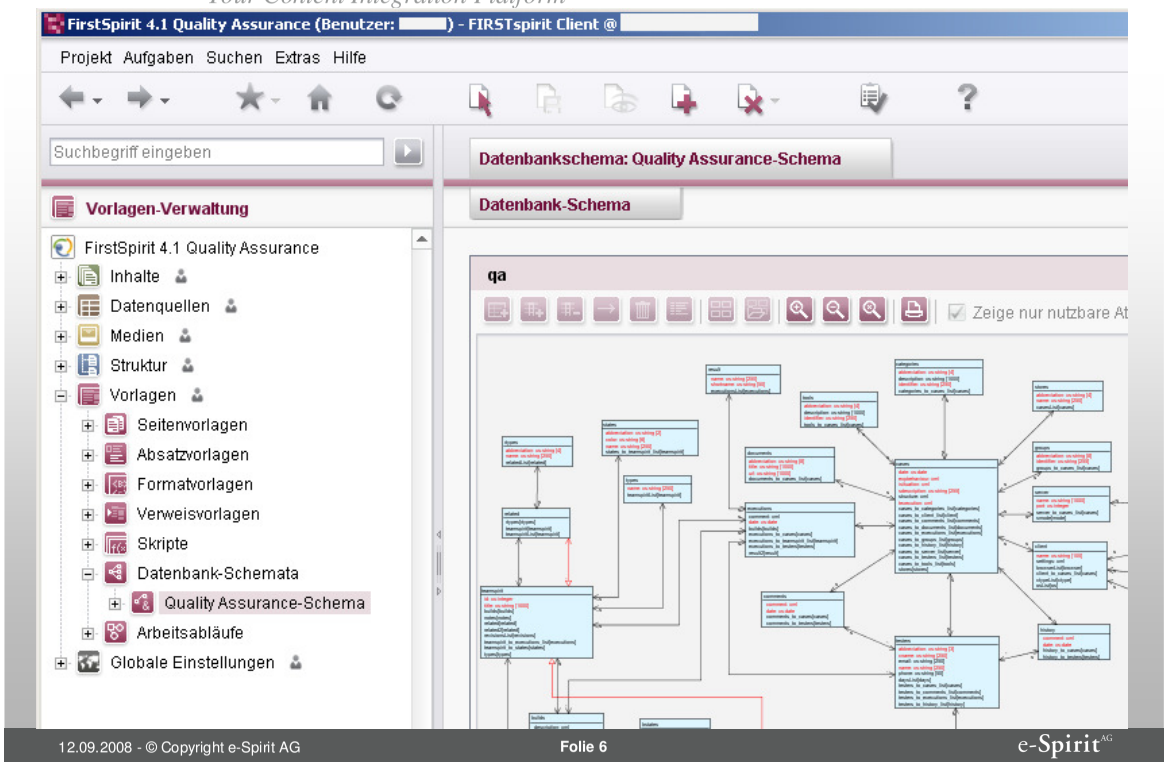
- **WICHTIG: Manuell != Unstrukturiert => Werkzeugunterstützung ist notwendig!**
  - Erfassung: Vorbedingungen, Aktion(en), Ergebnisse
  - Ausführung: Wann? / Wer? / Womit? (Softwarestand)
  - Unterstützung bei Vorbedingung: z.B. VM-System „Windows XP SP3 in RU“ starten
  - Gruppierung: z.B. „Release-Wechsel“ := Testfälle A, B, C sind auszuführen
  - Reporting: In Build X wurden die Testfälle E, F und G ausgeführt
  - Empfehlungen: Was wurde lange nicht getestet?

FirstSpirit wird zur Beschreibung und Ausführung von FirstSpirit-Tests genutzt!

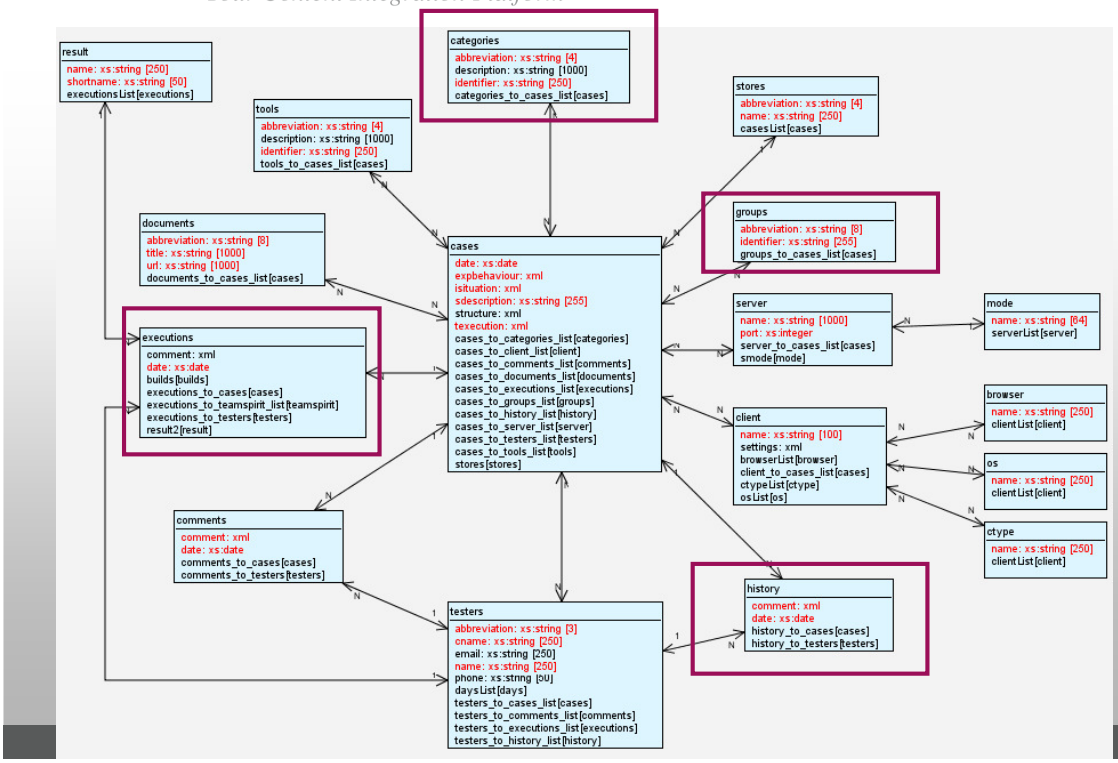
FirstSpirit{QA
Your Content Integration Platform



FirstSpirit{QA
Your Content Integration Platform



FirstSpirit{QA
Your Content Integration Platform



FirstSpirit{QA
Your Content Integration Platform

White-Box Testing in der Praxis

- JUnit: wie konnte man „früher“ überhaupt Software entwickeln?
- Coverage: Indikator für die Testfreudigkeit des Entwicklungsteams und „weiße Flecke“
  Overall JUnit/eXML: Class-Coverage 30%, LOC 25% => noch keine GUI-Tests

name	class, %	method, %	block, %	line, %
de.espirit.firstspirit.access [API]	75%	48%	49%	48%
de.espirit.firstspirit.server [SERVER]	68%	55%	42%	43%

- Code-Analyse: diagnostiziert „reale“ Bugs! (sehr oft Java-Sprachdefizite)
  - Null Pointer (NPE): Correctness / Null pointer dereference
  - Class Cast Exception (CCE): Correctness / Impossible cast ("use 'final' where possible")
- WICHTIG: Tägliche Routine!
  Testprotokolle und FindBugs-Analyse des neu eingecheckten Codes liegen jeden Morgen im Postfach aller Entwickler / QSler und werden dann auch bearbeitet!

eXML: eine XML-basierte Testsprache für die FS

- **Wieso nicht direkt testen? (direkte API-Aufrufe)**
  - Tests nicht vom Entwickler, sondern von QS
  - aufwändig: Herstellen der Vorbedingungen == redundanter Code
  - komplex: nicht nur Input/Output prüfen, sondern auch Concurrency und Multi-User-Simulation!

```
<ConnectServer host="${host}" port="${port}" _threads="${user}„ _epeatsPerThread="${userRepeats}"> ...
        <CreatePage pageFolder="${pageFolder}" templateName="standard">
            <Lock>
                <CreateSection name="Text" templateName="textAndPicture">
                <UploadMedia _id="pic" mediaFolder="${mediaFolder}" type="picture"
                             files="../verify/testcontent/paketiki.png"/>
                </CreateSection>
            </Lock>…
            <AssertReferences message="post conditions (current)" broken="0"/>
```

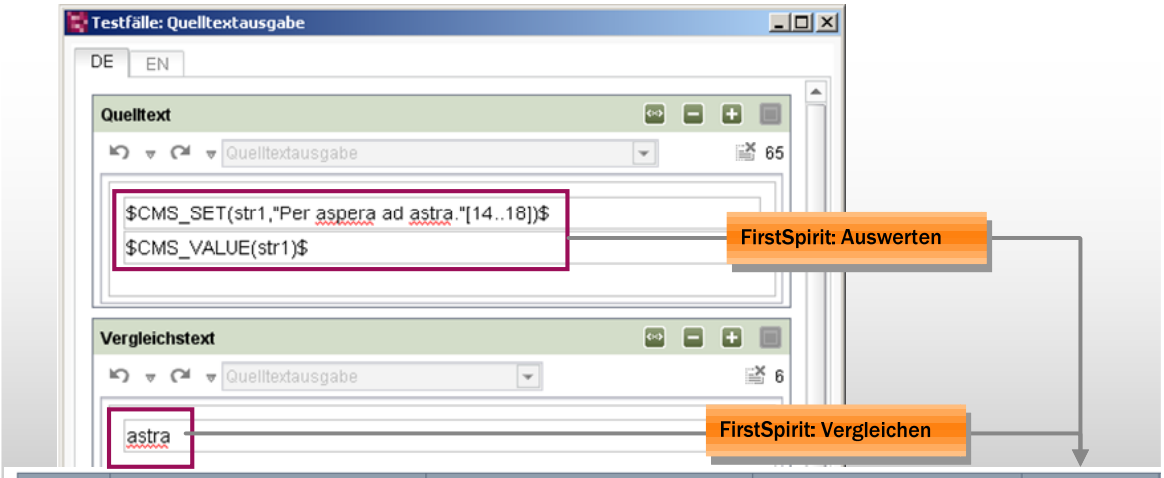

Regression-Test „im Kleinen“: FirstSpirit testet sich selbst!

- **Kernprinzip: FirstSpirit kann zur Laufzeit berechnete Ausdrücke auswerten und einen SOLL / IST-Vergleich der Ergebnisse durchführen.**
  Qualitätssicherung: Definition des Ausdrucks und des Erwartungswerts
  FirstSpirit: vollautomatische Auswertung des Ausdrucks und Prüfung gegen den Erwartungswert

- **Was ist „im Kleinen“ testbar?**
  - Ausdrücke der Template-Syntax
  - Eingabekomponenten
  - API-Funktionen
  - Systemvariable

- **Vorteil: Sehr systematisch und komplett**

- **Nachteil: Aufwändig in der Erstellung und Pflege**



FirstUnit ID	Eingabe	Ausgabe	Vergleichswert	Vergleichsergebnis
1121969 (#5)	$CMS_SET(str1,"Per aspera")$ $CMS_SET(str2," ad astra.")$ $CMS_SET(str3, str1+str2)$ $CMS_VALUE(str3)$	Per aspera ad astra.	Per aspera ad astra.	Erfolgreich
1121969 (#6)	$CMS_SET(str1,"Per aspera"[6])$ $CMS_VALUE(str1)$	p	p	Erfolgreich
1121969 (#7)	$CMS_SET(str1,"Per aspera ad astra."[14..18])$ $CMS_VALUE(str1)$	astra	astra	Erfolgreich
1121969 (#4)	$CMS_SET(str1,"Per aspera"+" ad astra.")$	Per aspera ad astra.	Per aspera ad astra.	Erfolgreich

FirstSpirit{QA
Your Content Integration Platform

Regression-Test „im Großen“: Vergleich mit „Referenz“ (1/2)

- Im Gegensatz zur „Regression im Kleinen“ werden komplette FirstSpirit-Projekte generiert (inkl. z.B. PDF) und Datei für Datei mit einem „Referenz-Generat“ verglichen
- „Leichte“ Abweichungen durch Systemspezifika, z.B. bei der Erzeugung von Grafiken aus Fonts, werden durch „unscharfe“ Vergleichsstrategien abgefedert
- Vorteil: es können „reale“ Projekte genutzt werden (kein Zusatzaufwand)
- Nachteil: es gibt keinen explizit definierten Testraum, d.h. Funktionen werden nur „zufällig“ getestet

Regression-Test „im Großen“: Vergleich mit „Referenz“ (2/2)

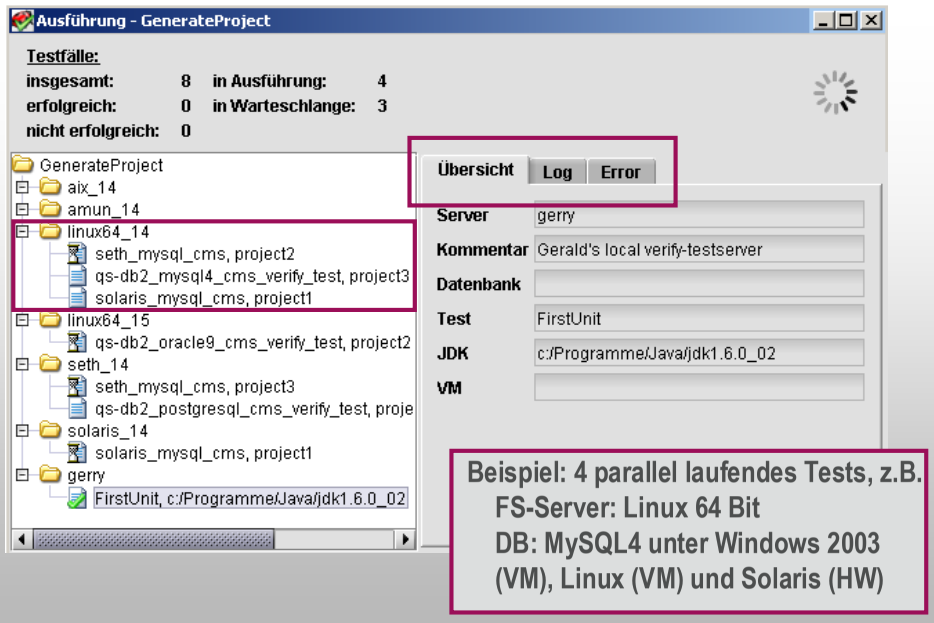
- **Problem: Testraum!**
  ~ 20 JDKs x 10 DB x 7 OS = 1400 Kombinationen (grobe Abschätzung)

  Lösung: VM-Ware mit Integration in die Regression-Test-Infrastruktur!

  - Installation der Referenz-OS (32/64-Bit) inkl. JDKs (1.5 und 1.6) und FirstSpirit mit Auto-Update-Funktion
  - Installation Referenz-DBs (32/64-Bit) auf separaten VMs
  - Testablauf:
    - Starten FirstServer- und Datenbank-VM
    - Software-Update FirstSpirit, Konfiguration JDK (vollautomatisch)
    - Projekt-Import und Generierung
    - Ergebnisse und Logfiles einsammeln und mit Referenz vergleichen

    ➔ parallele Ausführung soweit möglich

FirstSpirit{QA
Your Content Integration Platform



Beispiel: 4 parallel laufendes Tests, z.B.
FS-Server: Linux 64 Bit
DB: MySQL4 unter Windows 2003
(VM), Linux (VM) und Solaris (HW)

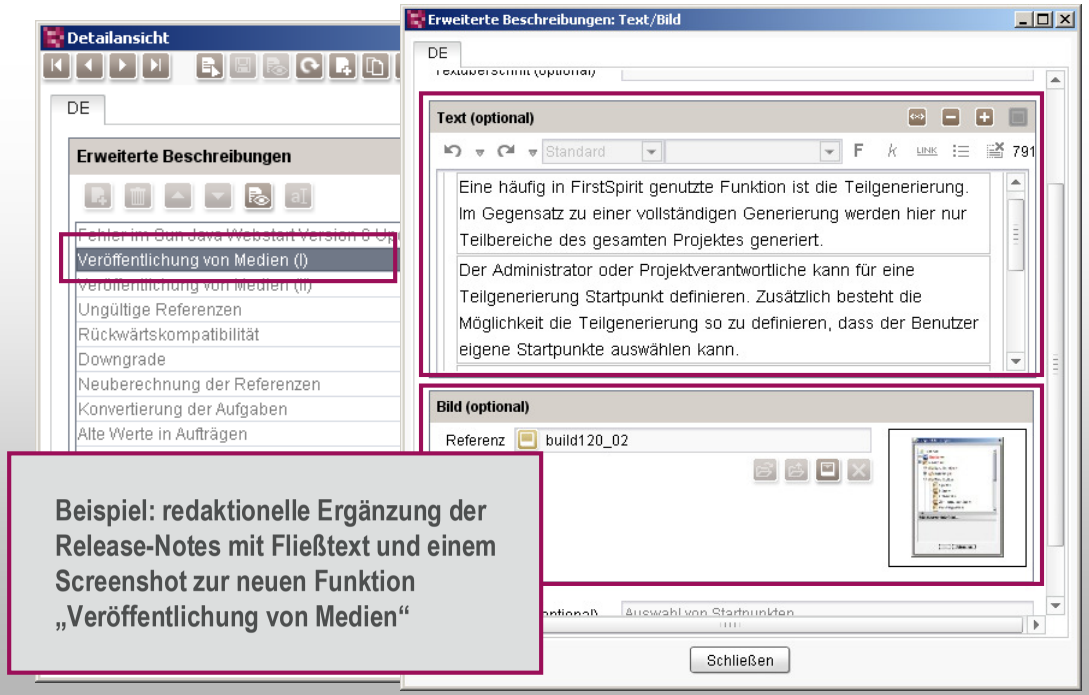
FirstSpirit{QA
Your Content Integration Platform

Release-Management

- **Issue-Tracking:**
  - In welchen Builds / Versionen ist ein Bugfix enthalten?
  - Welcher Build ist freigegeben / welcher nicht / wieso?
- **Redaktionelle Arbeiten:**
  - Kundenfreundliche Beschreibung von neuen Funktionen und beseitigten Fehlern
  - Einstellen von Bildern (Screenshots) zu neuen Funktionen
  - Übersetzung der Release-Notes, Erzeugung von PDF-Dokumenten
- **Interne Dokumente:**
  - Test-Protokoll: welche Tests wurden im Rahmen der Freigabetests durchgeführt?
  - Changelist: welche Dateien wurden zur Beseitigung von Fehlern verändert? (Forensik)

➔ **FirstSpirit wird zum Release-Management und zur Erstellung der Release-Notes genutzt!**

FirstSpirit{QA
Your Content Integration Platform



Beispiel: redaktionelle Ergänzung der Release-Notes mit Fließtext und einem Screenshot zur neuen Funktion „Veröffentlichung von Medien“

ID	Status	Titel
47781	Zu Qualitätsichern	Too many open files http://ts/47781

Gefixt in folgenden Branches:

STABLE_40_120 - **Letzte Änderung ab Build 4.0.132 verfügbar**

STABLE_40_90 - **Letzte Änderung ab Build 4.0.106 verfügbar**

trunk - **Letzte Änderung ab BUILD_4_1_DEV_7 verfügbar - Nach dies**

```
261                 _outNetBuffers.addLast(buffer);
262                 flushOutNet();
263               } finally {
264                 try {
                    -  getKey().interestOps(0);
265                 »  super.close();
266               } finally {
267                 _closed = true;
268                 _shutdown = false;
269               }
270             }
```

Changelog:

Revision: **24376** by **winkler** from **01.08.2008 08:52:22**

Bug #47781
Too many open files

merged with trunk

branches/STABLE_40_120/firstspirit/fs/src/impl/de/espirit/firstspirit/client/io/HttpServerCaller.java (diff, tslog, changelog)
branches/STABLE_40_120/firstspirit/fs/src/impl/de/espirit/firstspirit/client/io/SocketServerCaller.java (diff, tslog, changelog)
branches/STABLE_40_120/firstspirit/fs/src/impl/de/espirit/firstspirit/server/io/ChannelIO.java (diff, tslog, changelog)
branches/STABLE_40_120/firstspirit/fs/src/impl/de/espirit/firstspirit/server/io/ClientRequestHandler.java (diff, tslog, changelog)
branches/STABLE_40_120/firstspirit/fs/src/impl/de/espirit/firstspirit/server/io/DhArc4ChannelIO.java (diff, tslog, changelog)
branches/STABLE_40_120/firstspirit/fs/src/impl/de/espirit/firstspirit/server/io/NIOSocketServer.java (diff, tslog, changelog)
branches/STABLE_40_120/firstspirit/fs/src/impl/de/espirit/firstspirit/server/io/SSLChannelIO.java (diff, tslog, changelog)

Revision: **24131** by **winkler** from **24.07.2008 16:27:45**

FirstSpirit{QA
Your Content Integration Platform

Grenzen der Qualitätssicherung (im kommerziellen Umfeld)

- **Software ist komplex, FirstSpirit ganz besonders, weil:**
  - Entwicklungs- und Ausführungsumgebung in einem (z.B. Workflow- und Schema-Editor + Render-Engine für HTML / PDF)
  - erweiterbares System: Module und Scripte
  - komplexes System: es gibt „zufällige Funktionen“ (so nicht „geplante“ Nutzungen)

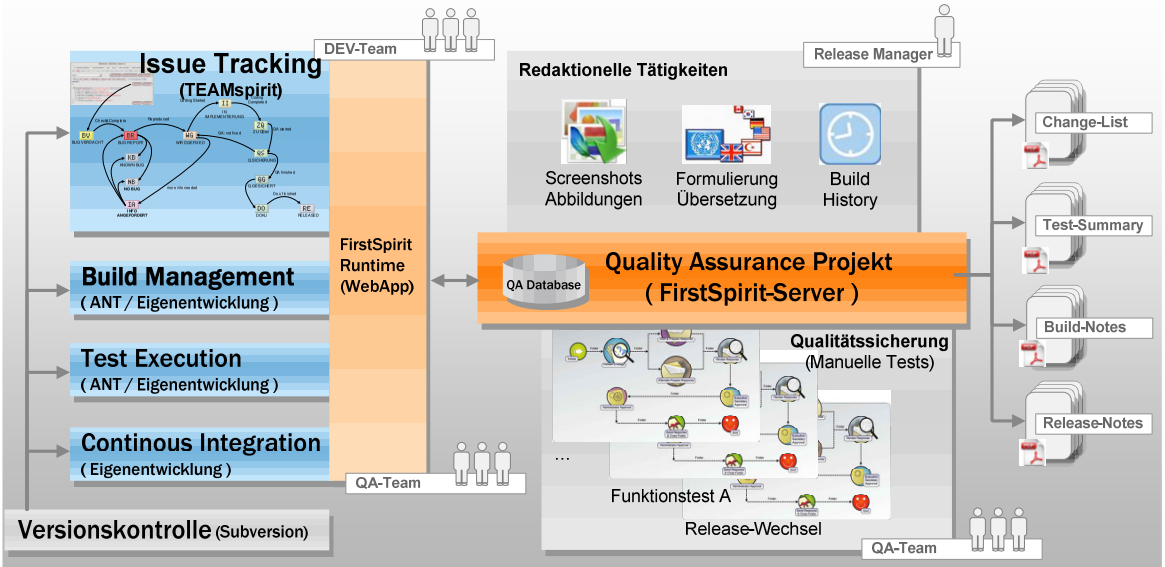
  ➔ **Einsatz-Szenarien sind nicht vorhersehbar**

- **„a-priori“ nicht effektiv testbar**
  - Zitat: „neue Funktionen sind erst mal nicht einsetzbar!“ OBWOHL die QS gelaufen ist
    - fehlende Funktionen (bei der Spezifikation übersehen)
    - „reale Situationen“ müssen den Testplan ergänzen!

  ➔ **Beta-Testing und Ramp-Up == unumgängliche Grundlagen für dauerhafte & effektive Qualitätssicherung!**

FirstSpirit{QA
Your Content Integration Platform

Zusammenfassung: Testen von und mit FirstSpirit (1/2)



FirstSpirit{QA
Your Content Integration Platform

Zusammenfassung: Testen von und mit FirstSpirit (2/2)

- **Komplexe Software & heterogene Umgebung:**
  - vollautomatische Tests = absolut zentrale Voraussetzung für langfristige Produktentwicklung
  - konsequenter Einsatz von Virtualisierungstechniken unumgänglich
- **Release-Management:**
  - enge (möglichst DB-basierte) Integration zwischen Build- / Test- und Release-Management mit Anbindung an das Versionskontrollsystem
- **Grenzen der Qualitätssicherung (im kommerziellen Umfeld):**
  - Einsatz-Szenarien nicht vorhersehbar → „a-priori“ nicht effektiv testbar
  - „reale Situationen“ müssen den Testplan ergänzen (Beta-Testing und Ramp-Up)
- **DogFood-Prinzip (wertvolle Ergänzung zur „regulären“ Qualitätssicherung):**
  - Pflicht: Einsatz der eigenen Software, wo immer möglich
  - Kür: das Softwareprodukt selbst ist Test-Infrastruktur und / oder testet sich selbst

Danke für Ihre Aufmerksamkeit!

Haben Sie noch Fragen?

e-Spirit AG
Barcelonaweg 14
44269 Dortmund
Germany

t +49 231 . 286 61-30
f +49 231 . 286 61-59

info@e-Spirit.de
www.e-Spirit.de