

# Reverse Testing

Ulrich Habel <[rhaen@cpan.org](mailto:rhaen@cpan.org)>

# Genesis

- Enterprise Content Management System

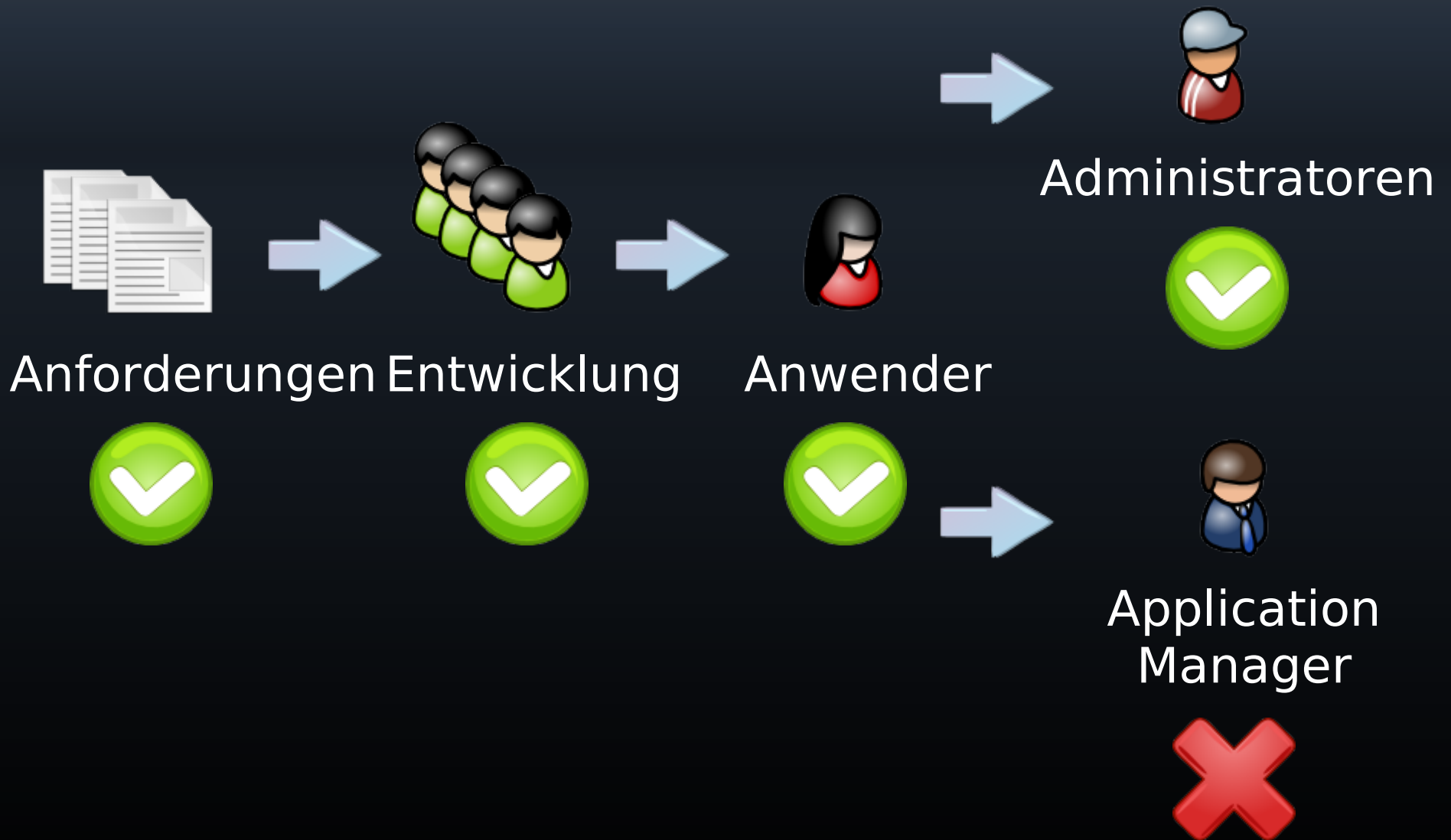
`iwserver: 6.7.1 Build 67998 Interwoven 20070828`

- Server ist nicht mehr startbar wenn XML Konfigurationsdateien Umlaute enthalten
- Nach Deployments endlose Tests...

# Wer testet eigentlich was?

- Es gibt für sämtliche Stufen eigene Testverfahren
- Testverfahren testen den Erfolg (Stabilität, Beständigkeit API/ABI)
- Wer hilft dem „Betreiber“ einer Applikation?

# Testphasen von Software



# Status „Reverse Testing“

- Projekt in der Designphase
- Projekt in der Zielorientierung
- Interesse?

...just starting...

# Was ist Reverse Testing

- Reverse Testing testet einen definierten Fehlerzustand
- Reverse Testing hilft bei der Implementierung in eine vorhandene Infrastruktur
- Reverse Testing hilft dem Application Management proaktiv
- Reverse Testing ist aktuell ein Framework von Regeln und Practices

# Für wen/welche Aufgaben?

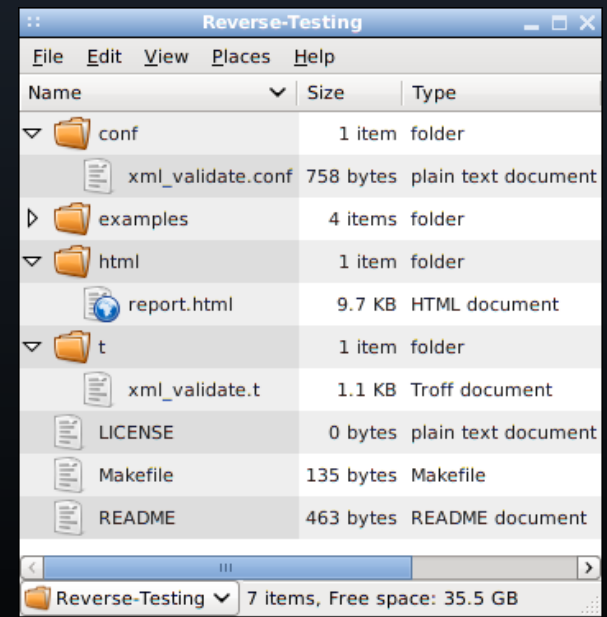
- Application Management (ITIL v3)
- Zustandstest für Dienstleistungen (Installationsservices, Konfigurationen)
- Jeder, der einen Service anbietet

# Lösung des XML Problems

- Verwendung von Test::Harness
- Validieren der XML Konfigdateien
- Ausgabe mittels TAP::Formatter::HTML

# Aufbau des Testframeworks

- Fester Aufbau in Form von Verzeichnissen, Dateien
- Konfigurierbar ausserhalb Perl Modulpfaden
- Dokumentation in den jeweiligen Dateien



# Was passiert wie?

- Tests werden im „t“ Verzeichnis abgelegt
- Konfigurationen werden im „conf“ Verzeichnis abgelegt
- Ein Test wird durchgeführt, wenn die Konfigurationsdatei dazu vorhanden ist
- Ausgabe im Verzeichnis „html“

# Wie passiert was?

- Shellzugriff erforderlich
- „make“ - verbesserungsfähig!
  
- Wunsch: Webserver mit `HTTP::Simple`

# Ausgabeformat TAP::Formatter::HTML

- Ausgabe eines Tests

Test Report - Minefield

file:///home/rhaen/Desktop/2009-PerlWorkshop/Reverse

[show failed](#)

## FAILED

Test file	Test results	Time	%
t/xml_validate	Useless use of a constant in void context at t/xml_validate.t line 21. <b>not ok 1 - Parsing Ein bewusster Fehler</b> # Failed test 'Parsing Ein bewusster Fehler' # at t/xml_validate.t line 24. ok 2 - Parsing Rollendefinitionen des TeamSite Servers ok 3 - Parsing Gruppendatei des TeamSite Servers ok 4 - Parsing Userdatei des TeamSite Servers 1.4 # Looks like you failed 1 test of 4. <b>exit status: 1, wait status: 256</b>	0.78s	75.0%
1 files	4 tests, 3 ok, 1 failed, 0 todo, 0 skipped, 0 parse errors exit status: 1, wait status: 256 elapsed time: 1 wallclock secs ( 0.05 usr 0.01 sys + 0.47 cusr 0.06 csys = 0.59 CPU)	0.78s	75.0%

Generated by TAP::Formatter::HTML v0.07 @ 19:32:15 24-Feb-2009

Done

# Mögliche Tests...

- Solaris pkg als Voraussetzung für eine Installation (Dienstleister)
- Locale Checks (Java auf Appservern)
- Application Connection Tracking (Firewalls)
- Überprüfung von Rechten (SELinux, policy-kit)
- Perl Modul Versionen

# Spielregeln

- Ein Test ergibt 0 oder 1 – keine Bereiche
- Test eines bekannten Fehlers, Fehlerquelle, Störung
- Der Test erzeugt Nachvollziehbarkeit
- Konfigurationsdatei enthält Beispiel (Bsp. Anwender ↔ Perl)
- Konfigurationsdateien werden validiert

# CPAN - Modul

- In Vorbereitung...
- Austausch von möglichen Tests
- Gemeinsam lassen sich Tests einfacher warten und pflegen

*...freue mich auf Gedankenaustausch...*

# Fragen?

- Fragen!

```
git clone git://pkgbox.org/reverse-testing.git
```

<http://www.pkgbox.org/Perl>