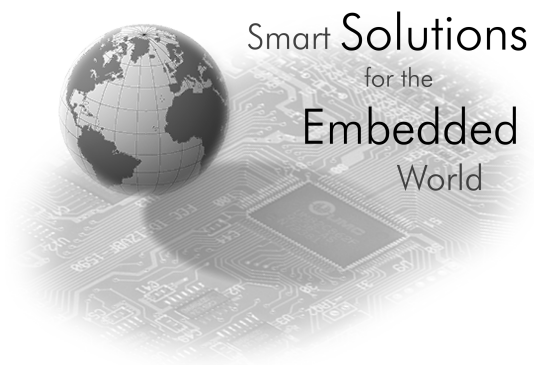


## Test von OSEK aus Sicht eines Betriebssystemherstellers

Dr. Jörg Nilson

**3SOFT** GmbH Erlangen

*(Joerg.Nilson@3SOFT.de)*

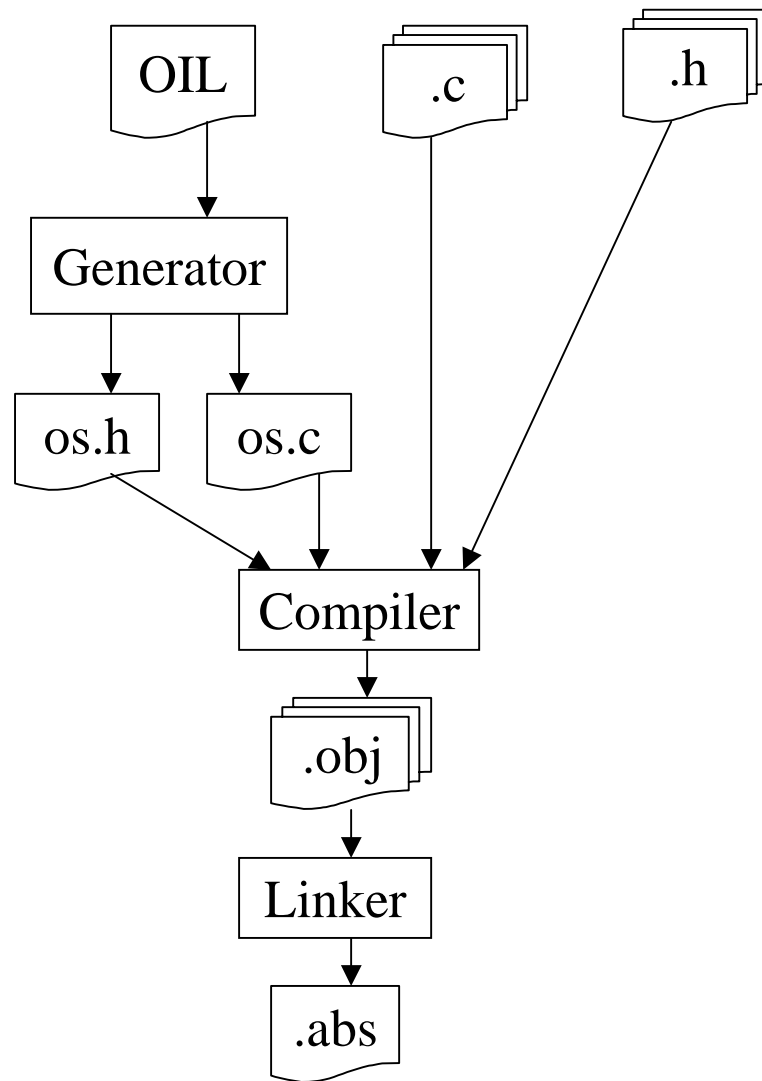


8. Juni 2001

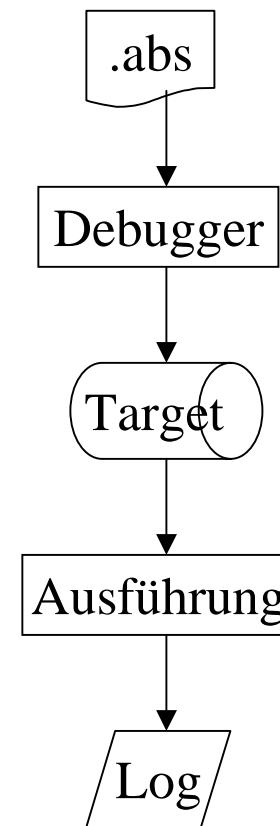
- ***Einführung***
- Automatisiertes Testen
- GUI
- Ausblick

- Konfigurationsvielfalt des Betriebssystems
  - OS: 8 Attribute  $\Rightarrow$  768 verschiedene Konfigurationen
  - Außerdem: TASK, COUNTER, ISR, ALARM, EVENT, RESOURCE mit jeweils weiteren Attributen
- Notwendigkeit von Regressionstests auf Systemebene
  - Betriebssystemkern ist zwar skalierbar, aber nicht modular!

- Notwendigkeit automatischer Tests
  - höhere Testhäufigkeit
  - Arbeitsentlastung  $\Rightarrow$  mehr Zeit, um neue Testfälle zu kreieren
  
- Vorteile einer Testsuite mit einer GUI können sein:
  - geringere Wahrscheinlichkeit der Fehlbedienung
  - höhere Akzeptanz



## Ablauf der Softwareerstellung mit ProOSEK



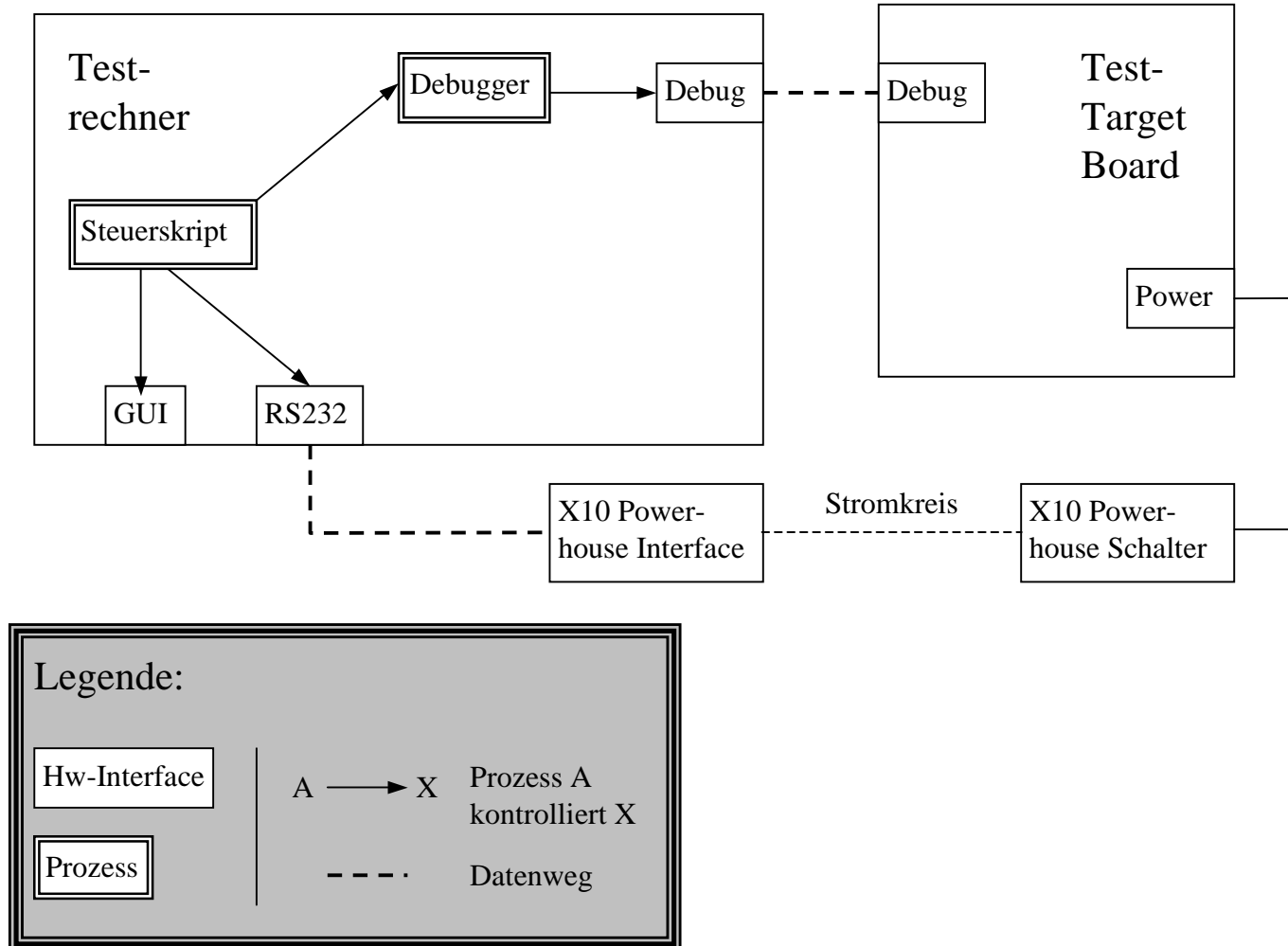
# Überblick

---

**3SOFT**

- Einführung
- ***Automatisiertes Testen***
- GUI
- Ausblick

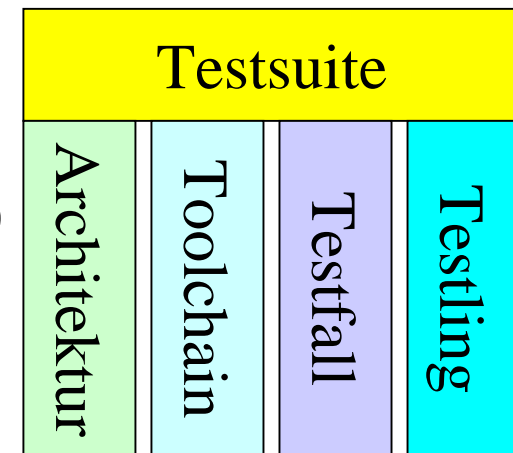
- Alle Tests für eine bestimmte Hardware (Targetboard) können ohne zwischenzeitliche Benutzerinteraktion durchgeführt werden (4-16 Std.).
- Sind zwei Targetboards unabhängig voneinander gleichzeitig an den Testrechner anschließbar, so können auch mehrere Architekturen hintereinander ohne zwischenzeitliche Benutzerinteraktion getestet werden.



- Steuerskript vollständig in Perl implementiert
- heute insgesamt ca. 10000 LOC verteilt über 35 Module (Packages)

- Abstraktionsschichten für

- Architektur/Derivat
- Toolchain (Compiler/Linker/Debugger)
- Testfall
- Testling, bzw. Betriebssystem (-variante)



- Blackbox- und Whitebox-Tests
  - Syntaktische Korrektheit
  - Testabdeckung des Generators
  - Tests bezüglich spezieller OS-Aspekte (Events, Ressourcen, Hooks, usw.)
  - Performanztests
- Ca. 200 Testfälle
- Ca. 800-2000 Einzeltests je Derivat und Toolchain

- 2 verschiedene Formen der Testabdeckung:
  - Generator-basiert (98 %, 2% dead code)
    - ohne Instrumentierung
    - in die Testsuite integriert
  - Block-basiert
    - mit skalierbarer Basisinstrumentierung
    - in die Testsuite integriert
    - zukünftig fester Bestandteil der Testprotokolle

# Überblick

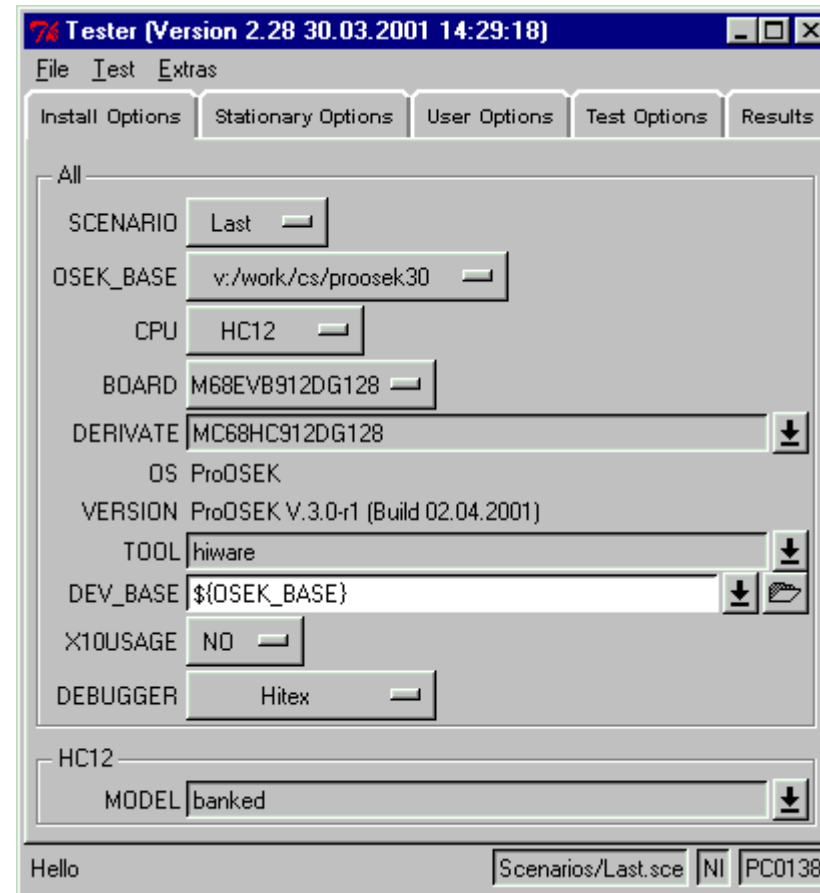
---

**3SOFT**

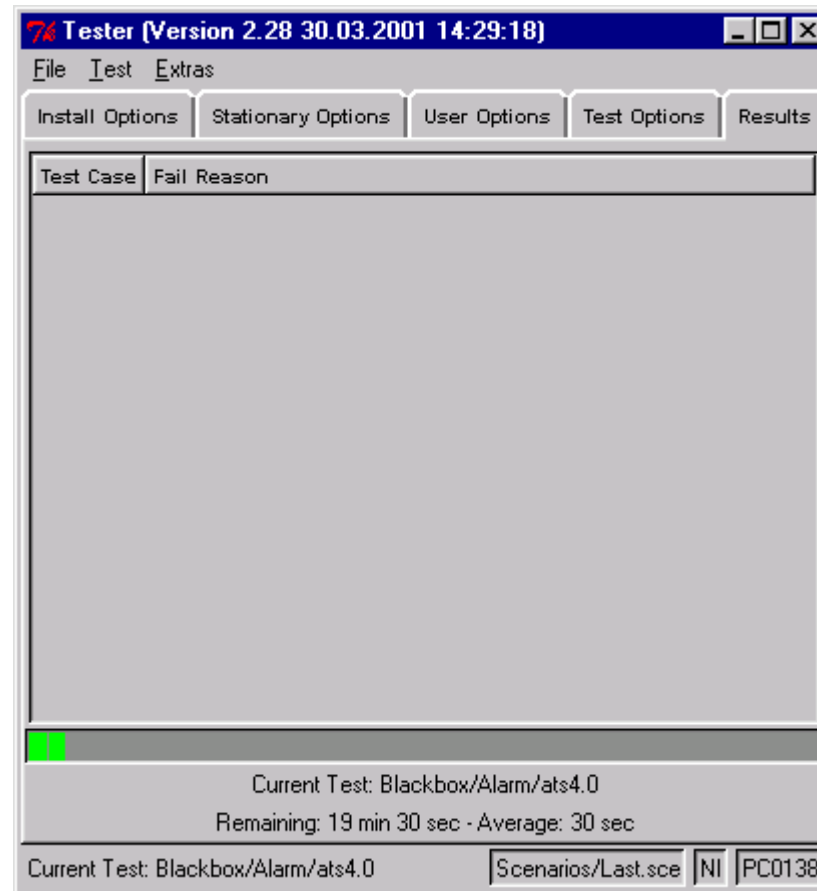
- Einführung
- Automatisiertes Testen
- **GUI**
- Ausblick

- Perl/Tk
- Constraint-gesteuert
- Tool-Tips
- Log-Window
- Plugins
- Unterteilung der Parameter nach ihrer Bindung
  - Workstation
  - Benutzer
  - Szenario
  - Sitzung

## Hauptfenster mit Sitzungsparametern



# Hauptfenster: Ergebnis



# Überblick

---

**3SOFT**

- Einführung
- Automatisiertes Testen
- GUI
- ***Ausblick***

- Integration der block-basierten Testabdeckung
  - auch verlässliche Aussagen über die Testabdeckung hinsichtlich tatsächlich ausgeführten Codes!
  
- Spezifikationssprache für Testfälle
  - geringerer Aufwand für
    - Wartung und
    - Erstellung von Testfällen

Vielen Dank für Ihre Aufmerksamkeit