Testen von Embedded Systemen mit Robot Framework

Embedded Testing 2018

Thomas Maierhofer Consulting

www.maierhofer.de

Agenda

- Vorstellung Thomas Maierhofer
- Vorstellung Robot Framework
- Aufbau von Tests mit Robot Framework
- Erweiterbarkeit / Remote Keyword Server
- HIL Tests
- Fragen

Thomas Maierhofer

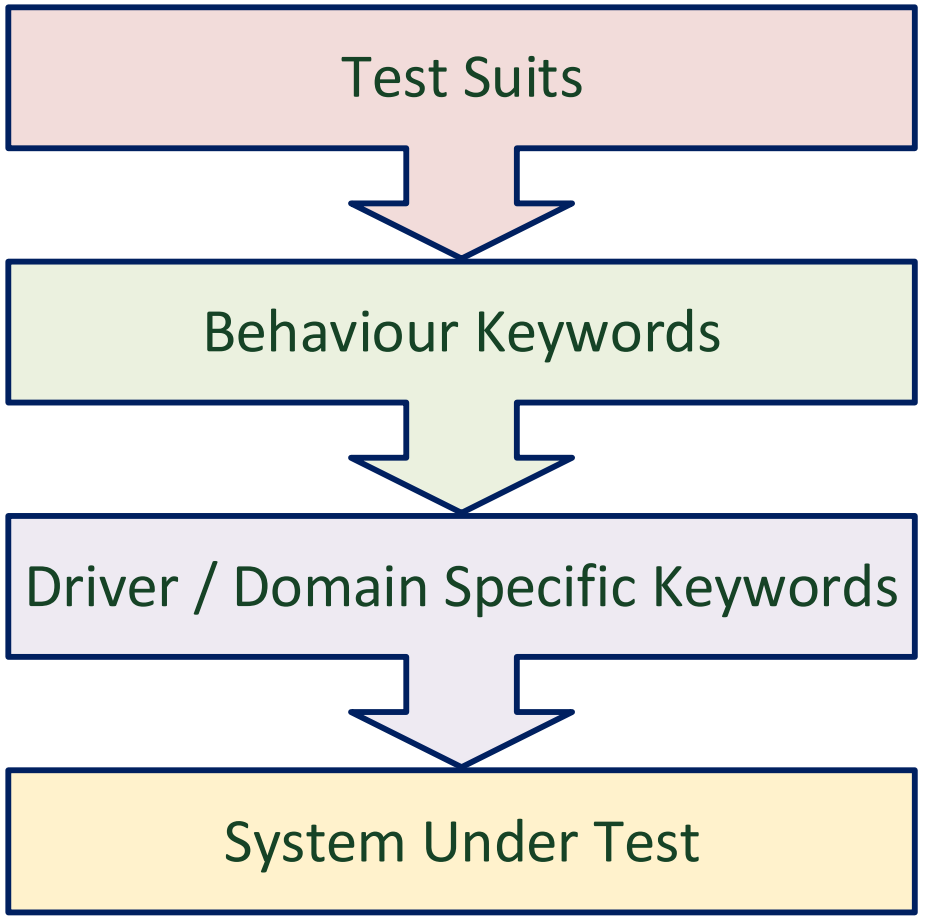
- Senior Software Consultant
- Seit 3 Jahren Berater bei Atlas Copco IAS
  Lead Developer „Nächste Steuerungsplattform“
- Embedded RT Linux
- .NET Core Framework on Embedded Linux / Browser HMI
- Industrie 4.0 / OPC-UA
- Feldbusse, insbesondere EtherCAT; TwinSafe; Beckhoff Komponenten
- Zertifizierter Scrum Master / Product Owner

Was ist Robot Framework?

- Generisches Open Source Test Framework
  - Entwickelt von Nokia Solutions – Hauptsponsor des Projekts
  - Apache Lizenz 2.0
  - Programmiersprache Python
  - Remote Keyword Server – in vielen Programmiersprachen verfügbar
- Keyword Driven Testing ohne strenge Methodische Bindung
  - Freestyle Integration Test / System Test
  - Acceptance Test Driven Development (ATDD)
  - Klassischer Testaufbau (AAA) oder Gherkin Variante (Given…,When…, Then…)

Keyword Driven Testing

- Testmethode nach ISO/IEC/IEEE 29119-5
- Abstrahierung des Verhaltens durch Schichten
- Komplexe Keywords werden über einfache Keywords aufgebaut
- Keywords lösen Aktionen aus



Beispiel „Benutzer Login“ – Web Demo

Given **browser is opened to login page**
When user "demo" logs in with password "mode"
Then welcome page should be open

Browser is opened to login page
Open browser to login page

Open Browser To Login Page
Open Browser ${LOGIN URL} ${BROWSER}
Maximize Browser Window
Set Selenium Speed ${DELAY}
Login Page Should Be Open

Selenium ist ein Framework für automatisierte Softwaretests von Webanwendungen

Open Browser
<Keyword Implemented by Selenium Driver>

Web API of Device
<System Under Test>

Testaufbau mit Robot Framework

```
*** Settings ***
Documentation   A Gherkin Style Test!
Resource        resource.robot
Test Teardown   Close Browser

*** Test Cases ***
Valid Login
    Given browser is opened to login page
    When user "demo" logs in with password "mode"
    Then welcome page should be open

*** Keywords ***
Browser is opened to login page
    Open browser to login page
...
```

Robot Framework Tests werden textuell beschrieben

Die Testdateien sind in feste Abschnitte unterteilt

Keywords sind nicht auf einzelne Worte beschränkt

Der Test enthält keinen Programmcode

Welche Typen von Tests werden unterstützt?

- Workflow Tests
  - Integrations-, System und Akzeptanztests
- Datengetriebene Tests
  - Testdaten als Bestandteil des Tests
  - Externe Testdaten; z.B. aus Modellen generiert
- Unterstützung der Extreme Programming (XP) Praktiken
  - Testgetriebene Entwicklung in unterschiedlichen Ausprägungen
    - Behavior Driven Development (BDD)
    - Acceptance Test Driven Development (ATDD)
  - Continuous Integration (CI)

Test (Keyword) Bibliotheken

- Erweiterung von Robot Framework
- Neue Keywords für spezifische Funktionen
- Anbindung des Tests an die „Reale Welt“
- Domänenspezifischer Treiber für Tests
- Eine große Zahl von Bibliotheken existiert
- Bibliotheken können in unterschiedlichen Sprachen erstellt werden
- Bibliotheken sind auch „Remote“ anbindbar

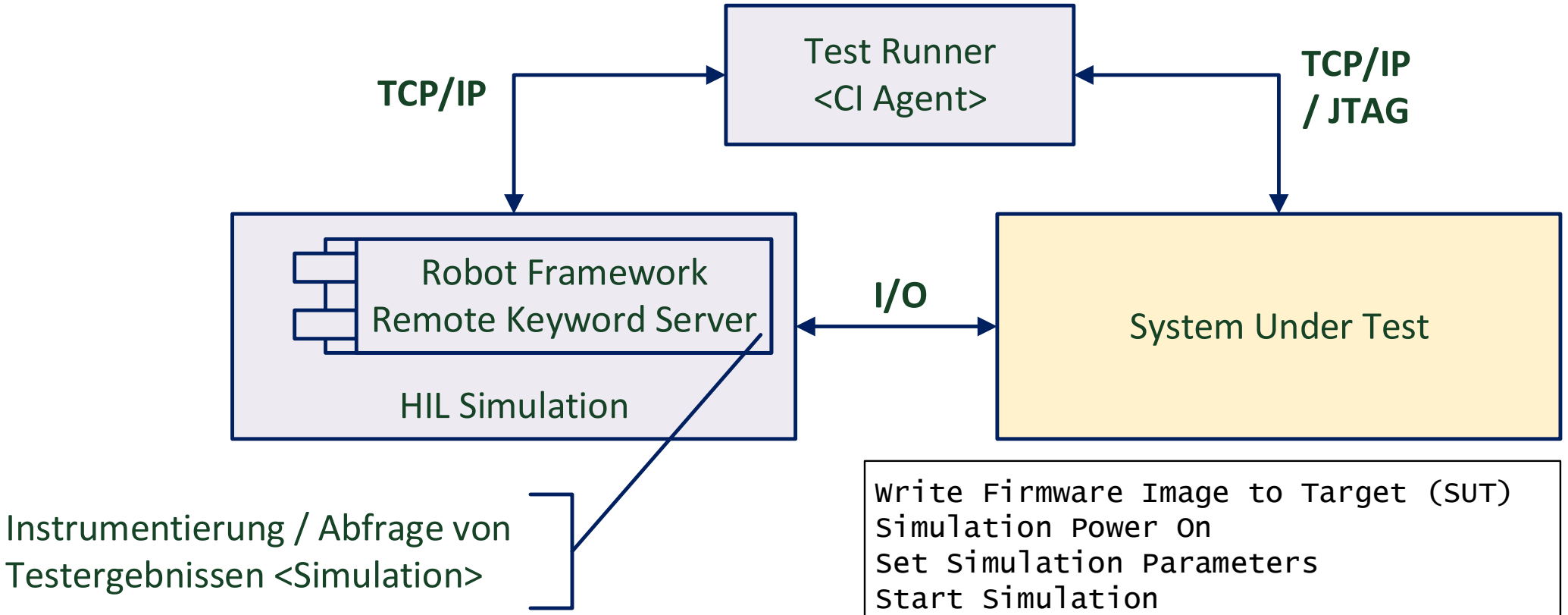
Remote Keyword Server

- Robot Framework definiert eine „Remote Library“ Schnittstelle
- Rein TCP basiert - mit HTTP und XML-RPC Interface
- Remote Keyword Server exportieren Keywords
- Auf der Server Plattform kein Python erforderlich
- Implementierung in beliebiger Programmiersprache möglich
- Kann auf dem Target zur Instrumentierung integriert werden
- Kann im Hintergrund von Echtzeitsystemen ausgeführt werden

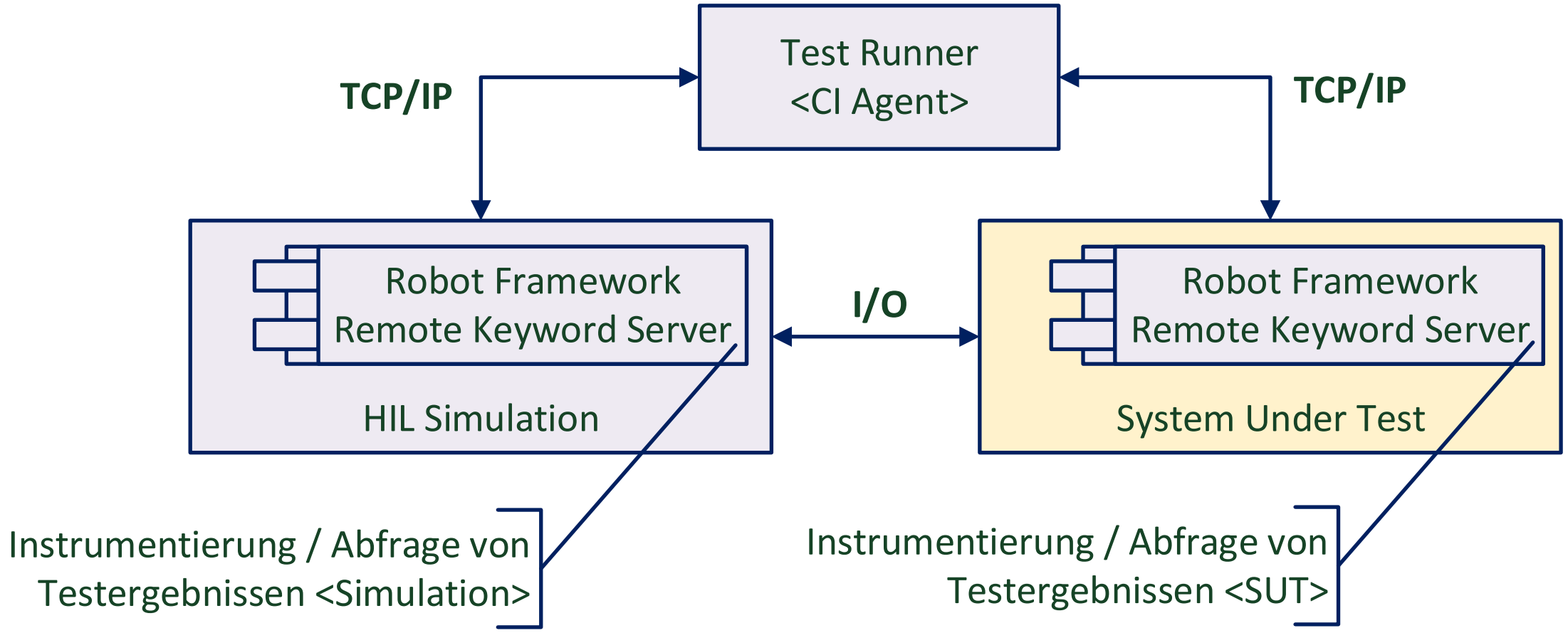
Embedded und Robot Framework

+ Instrumentierung von Testumgebungen, insbesondere HIL
+ Ausführen von Testsuits im Zuge des Buildprozesses
+ Optimale Unterstutzung von Embedded Linux (z.B. YOCTO)
+ Optimale Unterstützung zum Test von Web basierten HMIs
+ Einfache Erweiterbarkeit um eigene Infrastrukturen anzubinden
- Keine „out of the Box“ Lösung für Bare Metal Systeme

Schematischer Testaufbau Blackbox HIL Test



Schematischer Testaufbau Whitebox HIL Test



[Echtzeitfähige] Embedded Testumgebungen

- Simple Lösungen mit Einplatinencomputern und GPIO
  z.B. Raspberry PI mit Python GPIO Library + RT Linux
- Modulare Messsysteme / Laborstationen
  z.B. National Instruments ELVIS mit ELVIS API for Python
- Speicherprogrammierbare Steuerungen
  z.B. TwinCAT 3 mit ADS .NET API über IronPython
- Embedded Realtime Linux und Feldbusse
  z.B. YOCTO + ESD EtherCAT Master + Remote Keyword Server
- Instrumentierbare Modellbasierte Umgebungen
  z.B. ausführbare MatLab / SimuLink oder Colored Petrinet Modelle

HIL Test System – derzeit in Betrieb

- YOCTO Realtime Linux auf ARM basiertem System
- Robot Keyword Server auf .NET Core Runtime
- ESD EtherCAT Master Stack
- Beckhoff Buskoppler und Klemmen für I/O Anbindung
- C++ Simulation in Echtzeit – instrumentiert über Keyword Server
- SD Karten Switch – automatisches Erzeugen eines neuen Images
- „Power On“ des Targets über Beckhoff Klemmen
- Model Driven Testing mittels Colored Petrinet (CPN) Modellen

Continious Integration (CI) Pipeline

- Robot Framework Jenkins Plugin
- Auf dem Agent außer Python keine weiteren Systemanforderungen
- Auf dem HIL Simulator nur TCP/IP + Keyword Server nötig
- Auf dem Target keine Anforderungen für Black Box Tests
- Auf dem Target nur TCP/IP + Keyword Server nötig für White Box Tests

Erfahrungen aus dem Betrieb

- Tests werden strukturierter
  Entwickler sehen den Business Kontext durch den Testaufbau
- Tests sind für die Stakeholder Lesbar
  Kein Code in den Tests, Layer helfen Tests lesbar zu machen
- Standardisierte Umgebung
  Sowohl die Bestandsprodukte als auch die neu zu entwickelnde Steuerungsgeneration wird mit dem selben Framework getestet.
- Permanente Integration bringt Sicherheit
  Durch die CI Pipeline haben die Entwickler das Vertrauen den Code zu verbessern (Refactoring).

Weiterer (denkbarer) Ausbau

- Integration manueller Test (Test & Validation)
  - Nutzen von Robot Framework zur (Teil) Automatisierung
  - Permanente Erweiterung des HIL Tests
- Integration der direkt gesteuerten Peripherie ins HIL Test System
  - Z.B. Simulation des Antriebs über einen Bremsmotor (Erweiterung Motorenprüfstand zum HIL Test System)
  - Integration der Safety SPS und der Hardbuttons der HMI
  - Integration der Feldbusperipherie

Links

- Robot Framework
  http://robotframework.org/
- Robot Framework Source Code
  https://github.com/robotframework
- GPIO Python Library
  https://www.raspiprojekt.de/machen/basics/software/29-gpio-python-library.html
- MatLab und SimuLink
  https://de.mathworks.com/
- CPN Tools
  http://cpntools.org/