



# logi.DIAG Forschungsprojekt

Test Driven Automation und Condition  
Monitoring in der Systemumgebung logi.cals

logi.cals®  
by kirchner SOFT

Mehr als 20 Jahre Erfahrung in der Automatisierung

[www.logicals.com](http://www.logicals.com)

## Zur Person:

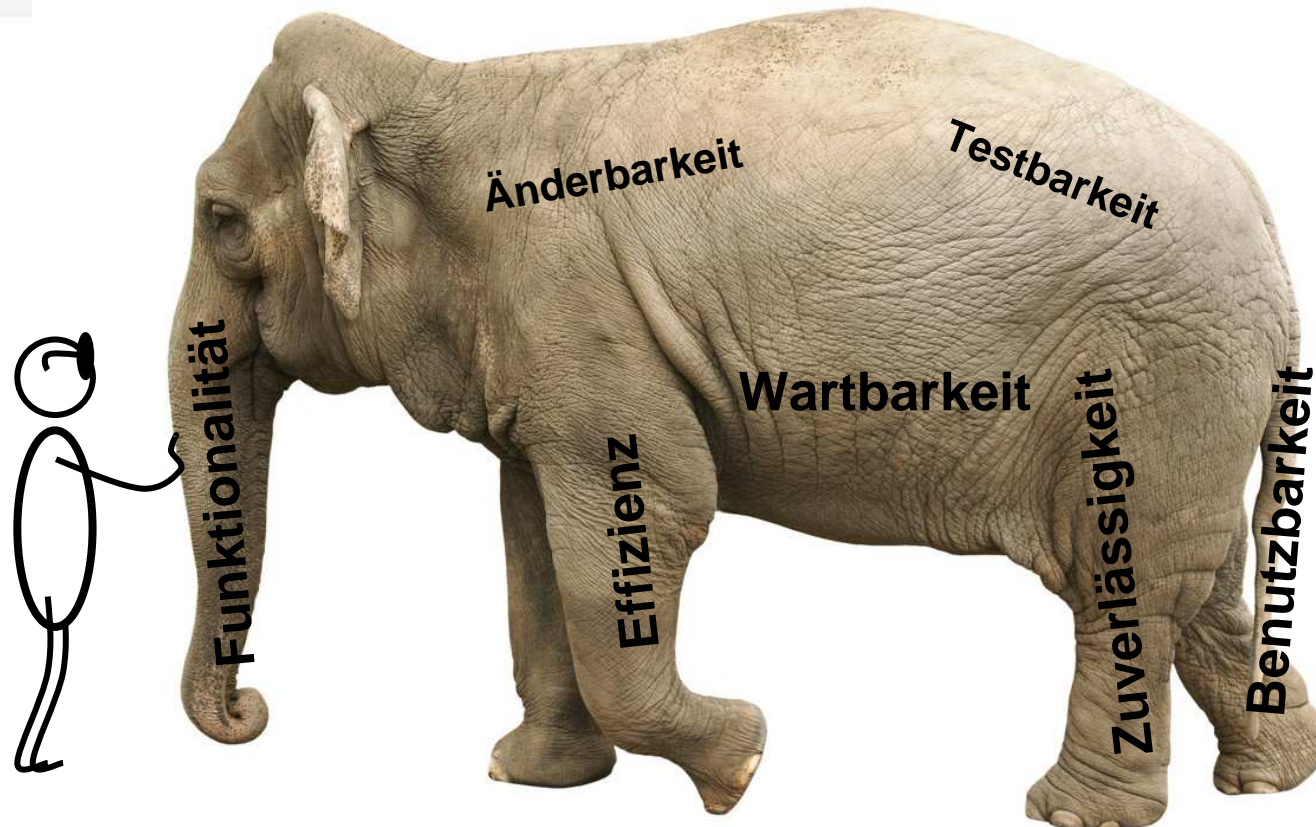
- Harald Nistelberger
- Seit 1992 bei logi.cals / kirchnerSOFT
- Bis Anfang 2009 Softwareentwickler
- Seitdem Leiter Qualitätssicherung

## Schlüsselerlebnis:

Austrian Software Quality Days, Keynote von Isabel Evans:

The Blind Men and the Quality Elephant

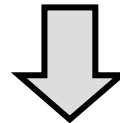
Isabel Evans:  
The Blind Men and the Quality Elephant



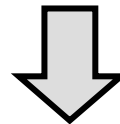
# Agenda

- Vorstellung
- logi.DIAG Forschungsprojekt
- Statischer Test durch Automatische Codeanalyse
- Testautomatisierung mit Keyword Driven Test

steigende Anforderungen



Erhöhung der Komplexität



umfangreichere Softwarefunktionalitäten

- 2 Schwerpunkte:
  - Test Driven Automation
  - Condition Monitoring

## logi.DIAG: Test Driven Automation

- Übernahme von Praktiken aus der betrieblichen IT in die Automatisierungswelt
  - Testgetriebener Entwicklungsprozess
  - Testbarere Architektur
  - Testpraktiken
- Entwicklung von Werkzeugen zur Prozess- und Testunterstützung
  - Anbindung an Lifecycle-Management  
(Anforderungs-, Test-, Fehler-, Konfigurationsmanagement)
  - Werkzeuge zur Testautomatisierung

## logi.DIAG: Condition Monitoring

- Kopplung zwischen Automatisierung und Condition Monitoring
  - Berücksichtigung der Diagnose in der Automatisierungsarchitektur
  - Nutzung der für den Test geschaffenen Diagnoseschnittstellen für Condition Monitoring Applikationen
- Entwicklung von neuen Algorithmen zur Datenweiterverarbeitung

## Partner

- TU-Wien
  - Institut für Automatisierungs- und Regelungstechnik (ACIN)  
([www.acin.tuwien.ac.at](http://www.acin.tuwien.ac.at))
  - Institut für Softwaretechnik und Interaktive Systeme (ISIS)  
([www.isis.tuwien.ac.at](http://www.isis.tuwien.ac.at))
- Universität Wien
  - Institut für Scientific Computing (ISC)  
([www.dac.univie.ac.at](http://www.dac.univie.ac.at))
- Messfeld GmbH  
([www.messfeld.com](http://www.messfeld.com))
- logi.cals, kirchner SOFT GmbH  
([www.logicals.com](http://www.logicals.com))
- Österreichische Forschungsförderungsgesellschaft mbH (FFG)



# Agenda

- Vorstellung
- logi.DIAG Forschungsprojekt
- Statischer Test durch Automatische Codeanalyse
- Testautomatisierung mit Keyword Driven Test

- Werkzeug für den Statischen Test
- Code-Analyse von IEC 61131 Quellcode
- Konfigurier- und Erweiterbares Regelwerk
- Unterstützt Kennzeichnung akzeptierter Abweichungen
- Anwendbar auf einzelne Bausteine oder ganze Applikationen
- In Codegenerierung integriert oder getrennt aufrufbar

# Was ist Statischer Test bzw. Statische Code Analyse

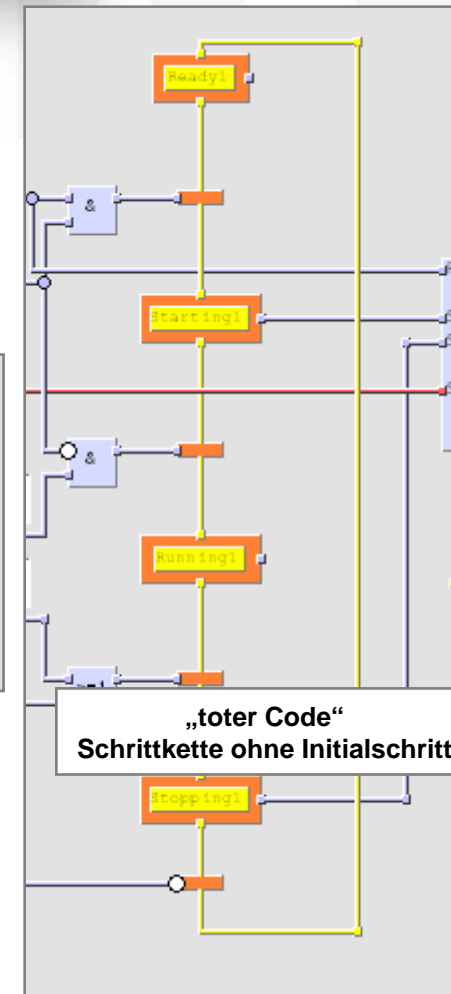
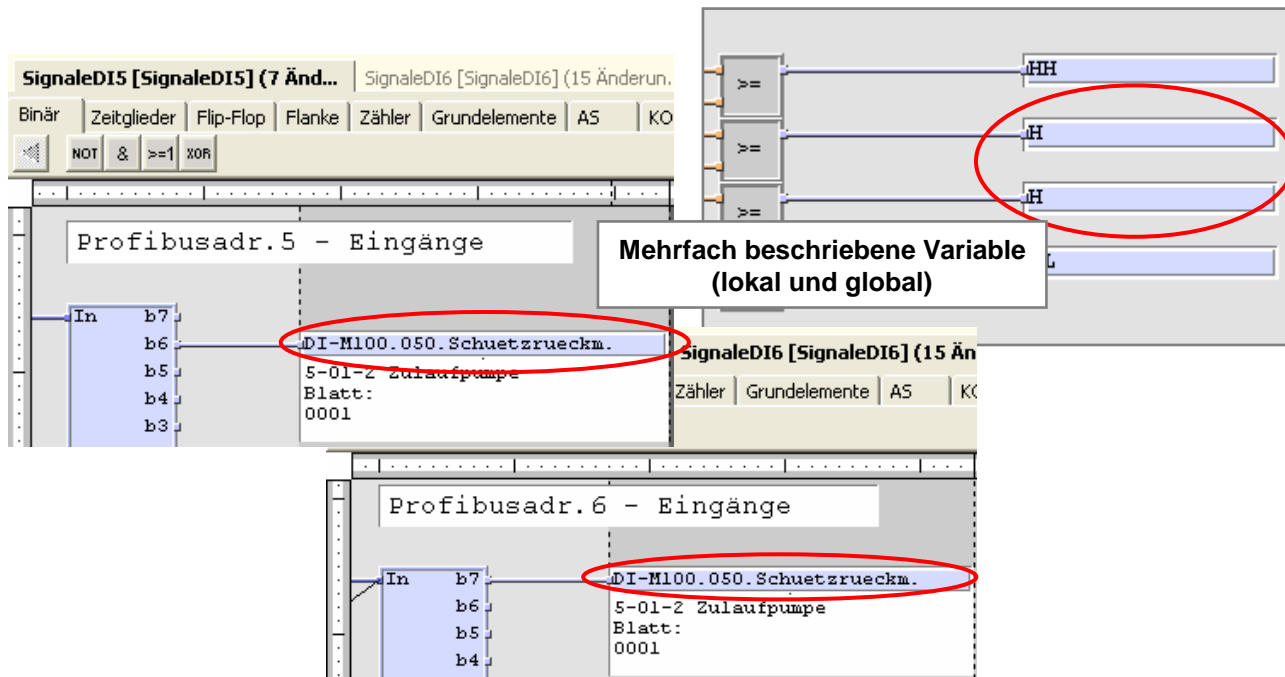
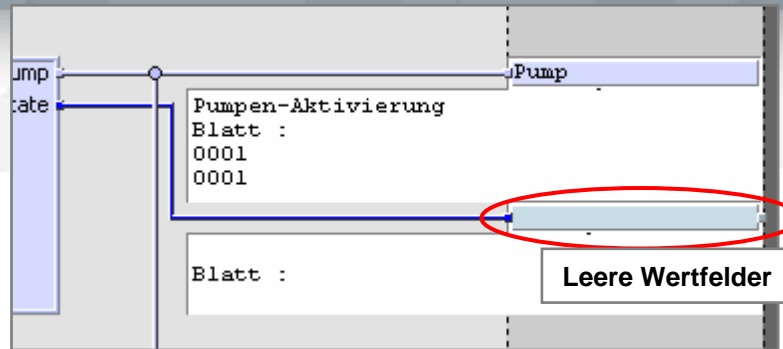
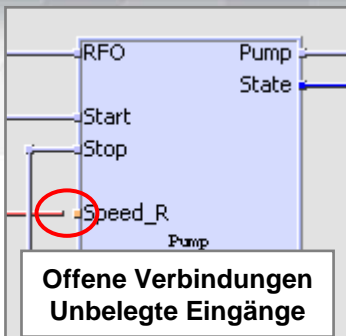
Eine oft unterschätzte Prüfmethode ist der sogenannte **statische Test** [...]. Im Gegensatz zum dynamischen Test wird das Testobjekt nicht mit Testdaten versehen und ausgeführt, sondern einer Analyse unterzogen. Diese kann in Form einer intensiven Betrachtung durch mehrere Personen erfolgen oder durch entsprechende Werkzeuge.

„Basiswissen Softwaretest“, Spillner & Linz 2003

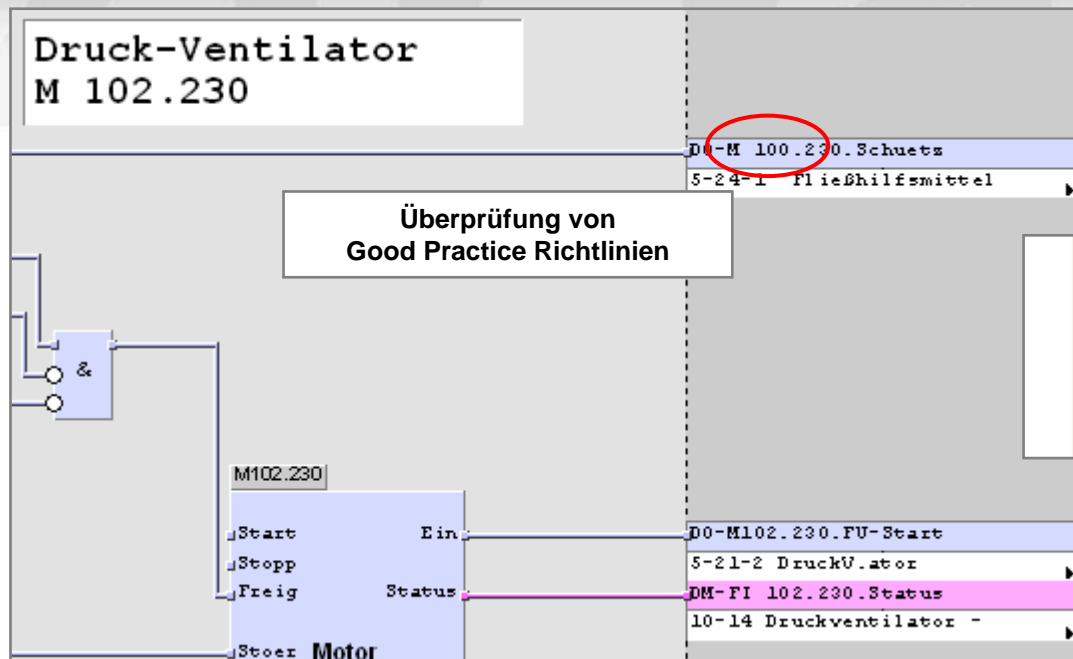
## Statische Testmethoden:

- Strukturierte Gruppenprüfungen (Reviews)
  - Eine Gruppe von Personen unterzieht ein Dokument einer mehr oder weniger formalen Prüfung
- Werkzeugunterstützte Statische Analyse
  - Analyse des Quellcodes durch ein Werkzeug
  - Nachweis von Fehlern oder fehlerträchtigen Situationen („Bad Smells“)
    - Syntaxüberprüfung
    - Konventions- oder Standardabweichungen (z.B. Namenskonventionen)
    - Kontrollflussanomalien (z.B. „toter Code“)
    - Datenflussanomalien (z.B. Mehrfachzuweisungen)
  - Ermittlung von Metriken
    - Geben Auskunft über Innere Qualitätsmerkmale wie Wartbarkeit oder Testbarkeit

# Welche Fehler können gefunden werden (1)



# Welche Fehler können gefunden werden (2)



**Einhaltung von Namenskonventionen**

DM-Zulaufpumpe.RmFe...	BOOL
DO- M101.230.FU-Start	BOOL
DO- M101.230.Schuetz	BOOL
DO-Fließhilfsm	
DO-M 100.230	
DO-M 100.230	
DO-M100.150.Schuetz	BOOL

6-09-4 Steuerpult-Taster  
freig

VAR	VAR INPUT	VAR OUTPUT
Name	Deklaration	Initialisieru
5sek	TIME	T#5s
freig	BOOL	1

**Nicht verwendete Variable**

## Vorteile Statischer Test durch Werkzeuggestützte Codeanalyse

- Kostengünstige Möglichkeit zur Qualitätsverbesserung
  - Automatischer Test – keine zusätzlichen Ressourcen notwendig
- Test sucht gezielt Fehlerquellen
  - Findet auch schwer zu provozierende Fehler
  - Aufwand zum Lokalisieren der Fehlerquelle aufgrund der Fehlerwirkung entfällt
- Verbesserung der inneren Qualität wie
  - Wartbarkeit/Änderbarkeit
  - Testbarkeit
  - Lesbarkeit

- Nicht alle Fehler sind durch statische Analyse auffindbar
  - Dynamischer Test trotzdem notwendig
- Zeigt **potentielle** Fehler (Nichteinhaltung von Regeln) auf
- Gefahr des „Der Junge, der Wolf schrie“ – Effekts
  - Echte Fehler gehen in Flut von akzeptierten Abweichungen unter

## Warum akzeptierte Abweichungen

- Abweichungen vom Regelwerk müssen zulässig sein
  - Code würde ineffizienter
  - Aufgabe nicht anders lösbar
  - Codierungsrichtlinien sind nicht überall gleich anwendbar (Firmenstandards)
  - Verwendung von „historischem“ Code

- Regelwerk muss konfigurierbar sein
  - Welche Regeln sollen geprüft werden, welche nicht
    - Verschiedene Ebenen, z.B. Firmenrichtlinien, Projektrichtlinien
  - Schwere der Abweichungen pro Regel konfigurierbar
    - Schwere Fehler, bricht weitere Verarbeitung ab
    - Warnung, Information
- Abweichungen müssen als „in Ordnung“ markiert werden können
  - Durch einfache Änderung
  - Ausnahmemarkierung im Code
  - Durch Bestätigen der Fehlermeldung

## Agenda

- ☑ Vorstellung
- ☑ logi.DIAG Forschungsprojekt
- ☑ Statischer Test durch Automatische Codeanalyse
- ☐ Testautomatisierung mit Keyword Driven Test

# Warum Testautomatisierung?

„Testautomatisierung verringert automatisch den Testaufwand“

## Testautomatisierung: Aufwand und Amortisierung

- Aufwand Testautomatisierung gegenüber manuellen Test
  - ↑ Erstellung der Testfälle
  - ↓ Testdurchführung
  - ↑ Anpassung bei Änderungen
- Voraussetzung für Effiziente Testautomatisierung
  - Einfache Testfallerstellung
  - Hohe Wiederverwertung
    - Für verschiedenen Testumgebungen
    - Für verschiedene Komponenten
    - Durch häufige Testwiederholung
  - Verringerung des Anpassungsaufwandes

# Automatisierbare Testfall-Spezifikation: Vom Prosatext zu Aktionswörtern

- Testanweisungen in Prosatext – für Tester verständlich, aber nicht automatisierbar
- „Programmierte“ Testanweisungen – Automatisierbar, setzen Programmierkenntnisse voraus
- Idee hinter Testfallbeschreibung mit Aktionswörtern:
  - So formal wie nötig, so verständlich wie möglich
  - Automatische Abarbeitung der Testfälle möglich
  - Testfall bleibt für den Tester lesbar

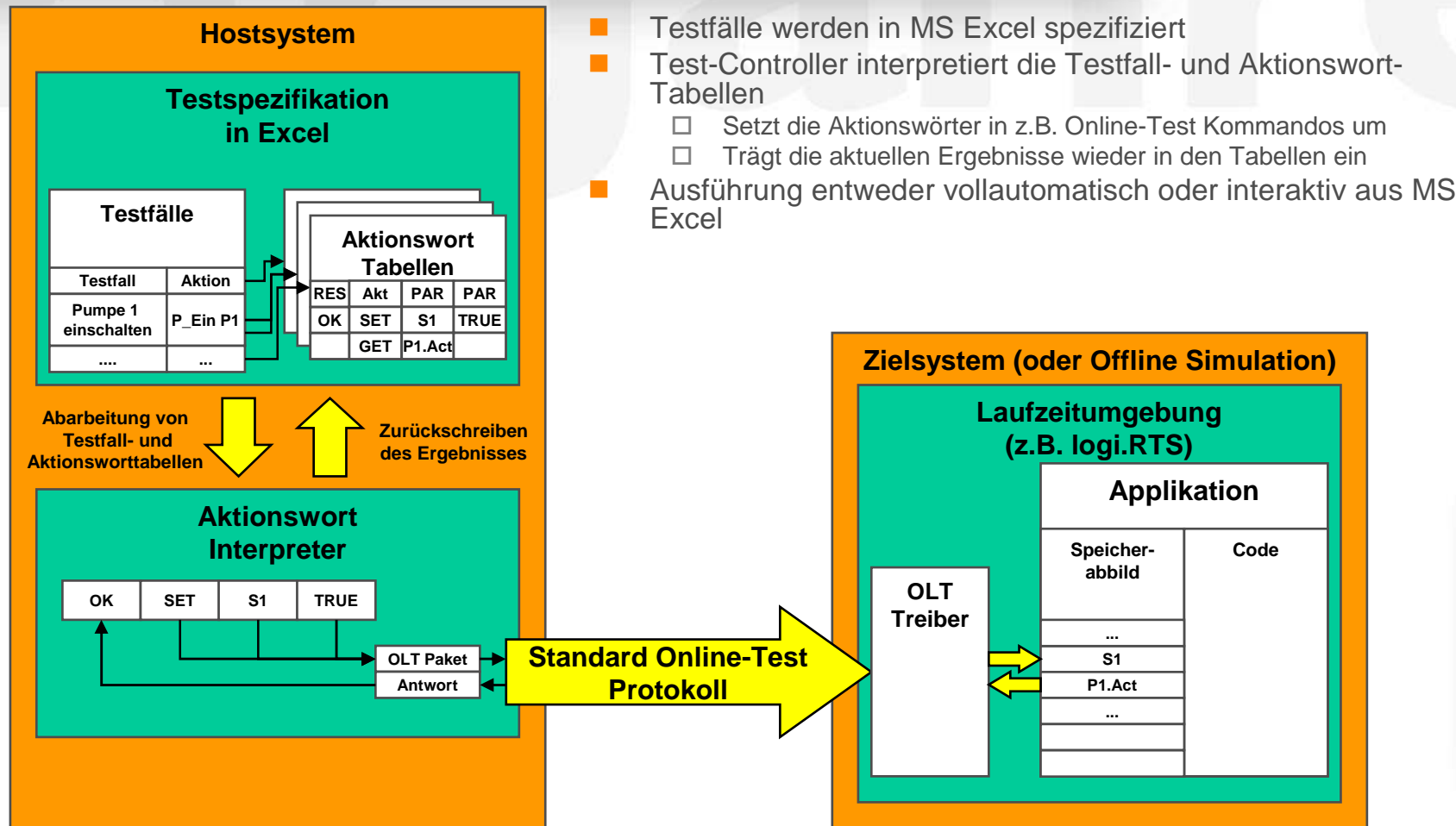
Prosa Testfallbeschreibung	Testfallbeschreibung mit Aktionswörtern (Schlüsselwörtern)
1. L1000_H und L1000_HH zurücksetzen	Setze Signal: „L1000_H“ = „FALSE“ Setze Signal: „L1000_HH“ = „FALSE“
2. Einschaltbereitschaft (RFOAuto) aktivieren	Setze Signal: „RFOAuto“ = „TRUE“
3. Überprüfen ob Pumpe P1000 läuft	Prüfe: „P1000“ = „TRUE“

## Automatisierbare Testfall-Spezifikation: Ausnutzung der Wiederverwendung

- Umfangreichere Aktionen können in mächtigere Aktionswörter gekapselt werden
  - Mehrmaliges Beschreiben eines Ablaufs nicht mehr notwendig
  - Komplexität kann vor dem Tester verborgen werden
  - Bei Änderung muss nur ein Aktionswort angepasst werden
- Durch entsprechende Parametrierung kann ein Test für mehrere Komponenten verwendet werden

Prosa Testfallbeschreibung	Testfallbeschreibung mit Aktionswörtern (Schlüsselwörtern)
1. Pumpe P1000 einschalten (siehe auch Testfall „Pumpe P1000 einschalten“)	Pumpe ein: „P1000“
2. Innerhalb von 10 sec wieder ausschalten	Warte: „2s“ Pumpe aus: „P1000“
3. Pumpe muss weiterlaufen	Prüfe: „P1000“ = „TRUE“

# Testautomatisierung mit logi.DIAG

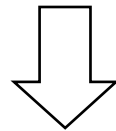


- Testfälle werden in MS Excel spezifiziert
- Test-Controller interpretiert die Testfall- und Aktionswort-Tabellen
  - Setzt die Aktionswörter in z.B. Online-Test Kommandos um
  - Trägt die aktuellen Ergebnisse wieder in den Tabellen ein
- Ausführung entweder vollautomatisch oder interaktiv aus MS Excel

# Einfache Aktionsworttabelle

Run	Stop	Reset				
Ergebnis	Aktion	P1	P2	P3	P4	P5
	START					
	SET	L1000_H	FALSCH			
	SET	L1000_HH	FALSCH			
	SET	RFOAuto	WAHR			
	WAIT	1000				
	ASSERT	P1000	WAHR			
	GET	P1000_Error				
	STOP					
Anweisungen vor dieser Zeile einfügen			Anweisungen vor dieser Zeile einfügen			

- Ausführung kann direkt in Excel gestartet werden
- End- und auch Zwischenergebnisse werden direkt in Arbeitsblatt eingetragen
- Detaillierte Protokollierung des Testlaufs
- Unterstützung bei der Fehlersuche
- Einfache Erstellung von Regressions- und Vergleichstests



Ergebnis	FALSCH					
Run	Stop	Reset				
Ergebnis	Aktion	P1	P2	P3	P4	P5
WAHR	START					
WAHR	SET	L1000_H	FALSCH			
WAHR	SET	L1000_HH	FALSCH			
WAHR	SET	RFOAuto	WAHR			
WAHR	WAIT	1000				
FALSCH	ASSERT	P1000	WAHR	FALSCH		
WAHR	GET	P1000_Error		0		
WAHR	STOP					
Anweisungen vor dieser Zeile einfügen			Anweisungen vor dieser Zeile einfügen			

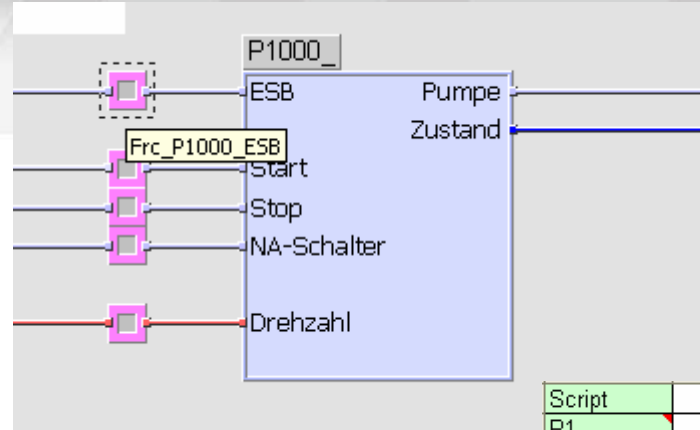
  

Ergebnis	WAHR					
Run	Stop	Reset				
Ergebnis	Aktion	P1	P2	P3	P4	P5
WAHR	START					
WAHR	SET	L1000_H	FALSCH			
WAHR	SET	L1000_HH	FALSCH			
WAHR	SET	RFOAuto	WAHR			
WAHR	WAIT	1000				
WAHR	ASSERT	P1000	WAHR	WAHR		
WAHR	GET	P1000_Error		0		
WAHR	STOP					
Anweisungen vor dieser Zeile einfügen			Anweisungen vor dieser Zeile einfügen			

Run Test								Tabellennamenname						
Komponente	Kompo	Test Id	Szenario	Subszenario	Vorbedingungen	Beschreibung	Erw. Ergebnis	Ergebnis	Script	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5
Zuflusspumpe 1	P1000	T1.1.1	Pumpe Einschalten		Pumpe Aus, ESB Ein		Pumpenschütz ein	WAHR	Pumpe_Ein	P1000	WAHR	WAHR		
Zuflusspumpe 1	P1000	T1.1.2	Pumpe Einschalten	Keine ESB	Pumpe Aus, ESB Aus		Pumpenschütz aus	WAHR	Pumpe_Ein	P1000	FALSCH	FALSCH		
Zuflusspumpe 1	P1000	T1.1.3	Pumpe Einschalten	Pumpe läuft schon	Pumpe Ein		Pumpenschütz ein	WAHR	Pumpe_Ein_Drehzahl	P1000	100	0		
Zuflusspumpe 1	P1000	T1.1.4	Pumpe Einschalten	Pumpe läuft nicht an	Pumpe Aus, ESB Ein	Pumpendrehzahl < 5% nach 5sec	Fehlerzustand 1	WAHR	Pumpe_Ein_Drehzahl	P1000	4	1		
Zuflusspumpe 1	P1000	T1.2.1	Pumpe Läuft		Pumpe > 10sec ein			WAHR	Pumpe_Lauft	P1000				
Zuflusspumpe 1	P1000	T1.2.2	Pumpe Läuft	Pumpe überdreht	Pumpe > 10sec ein	Pumpendrehzahl > 100%	Fehlerzustand 2	WAHR	Pumpe_Lauft_Drehzahl	P1000	110	2		
Zuflusspumpe 1	P1000	T1.2.3	Pumpe Läuft	Pumpe wird langsamer	Pumpe > 10sec ein	Pumpendrehzahl < 5%	Fehlerzustand 3	WAHR	Pumpe_Lauft_Drehzahl	P1000	4	1		
Zuflusspumpe 1	P1000	T1.3.1	Pumpe Abschalten		Pumpe > 10sec ein			WAHR	Pumpe_Lauft_Stop_Drehzahl	P1000	0	0		
Zuflusspumpe 1	P1000	T1.3.2	Pumpe Abschalten	Noch in Anlaufphase	Pumpe < 10sec ein		Pumpe läuft weiter	WAHR	Pumpe_Ein_Stop	P1000				
Zuflusspumpe 1	P1000	T1.3.3	Pumpe Abschalten	Schaltet nicht ab	Pumpe > 10sec ein	Pumpendrehzahl > 0% nach 5 sec	Fehlerzustand 4	WAHR	Pumpe_Lauft_Stop_Drehzahl	P1000	20	4		
Zuflusspumpe 1	P1000	T1.4.1	Pumpe Not-Aus		Pumpe > 10sec ein		Pumpenschütz aus	WAHR	Pumpe_Lauft_NotAus_Drehzahl	P1000	0	0		
Zuflusspumpe 1	P1000	T1.4.2	Pumpe Not-Aus	Noch in Anlaufphase	Pumpe < 10sec ein		Pumpenschütz aus	WAHR	Pumpe_Ein_NotAus_Drehzahl	P1000	0	0		
Zuflusspumpe 1	P1000	T1.4.3	Pumpe Not-Aus	Schaltet nicht ab	Pumpe > 10sec ein	Pumpendrehzahl > 0% nach 5 sec	Fehlerzustand 4	WAHR	Pumpe_Lauft_NotAus_Drehzahl	P1000	20	4		

## Aktionsworttabellen Aufrufe

# Erweitert Testfall: Parameter, Verwendung von Forcemarker



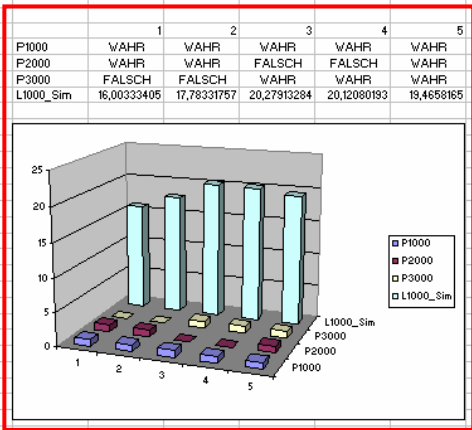
- Auftrennen der Logik durch Setzen von Forcemarkern
- Ermöglicht gezieltes Testen von Teilfunktionalitäten im Kontext der Gesamapplikation
- Vorgabewert setzen und Aktivieren mit einem Aktionswort

Script	Pumpe_Ein					
P1	P1000					
P2	FALSCH					
P3	FALSCH					
P4						
P5						
Res	WAHR					
<div style="display: flex; justify-content: space-around;"> <span>Run</span> <span>Stop</span> <span>Reset</span> </div>						
Ergebnis	Aktion	P1	P2	P3	P4	P5
WAHR	START					
WAHR	FORCE	PumpStation.Frc_%P1%_ESB	%P2%			
WAHR	FORCE	PumpStation.Frc_%P1%_Start	WAHR			
WAHR	WAIT	2000				
WAHR	ASSERT	%P1%	%P3%	FALSCH		
WAHR	ASSERT	%P1%_Error	0	0		
WAHR	STOP					
Anweisungen vor dieser Zeile einfügen		Anweisungen vor dieser Zeile einfügen				

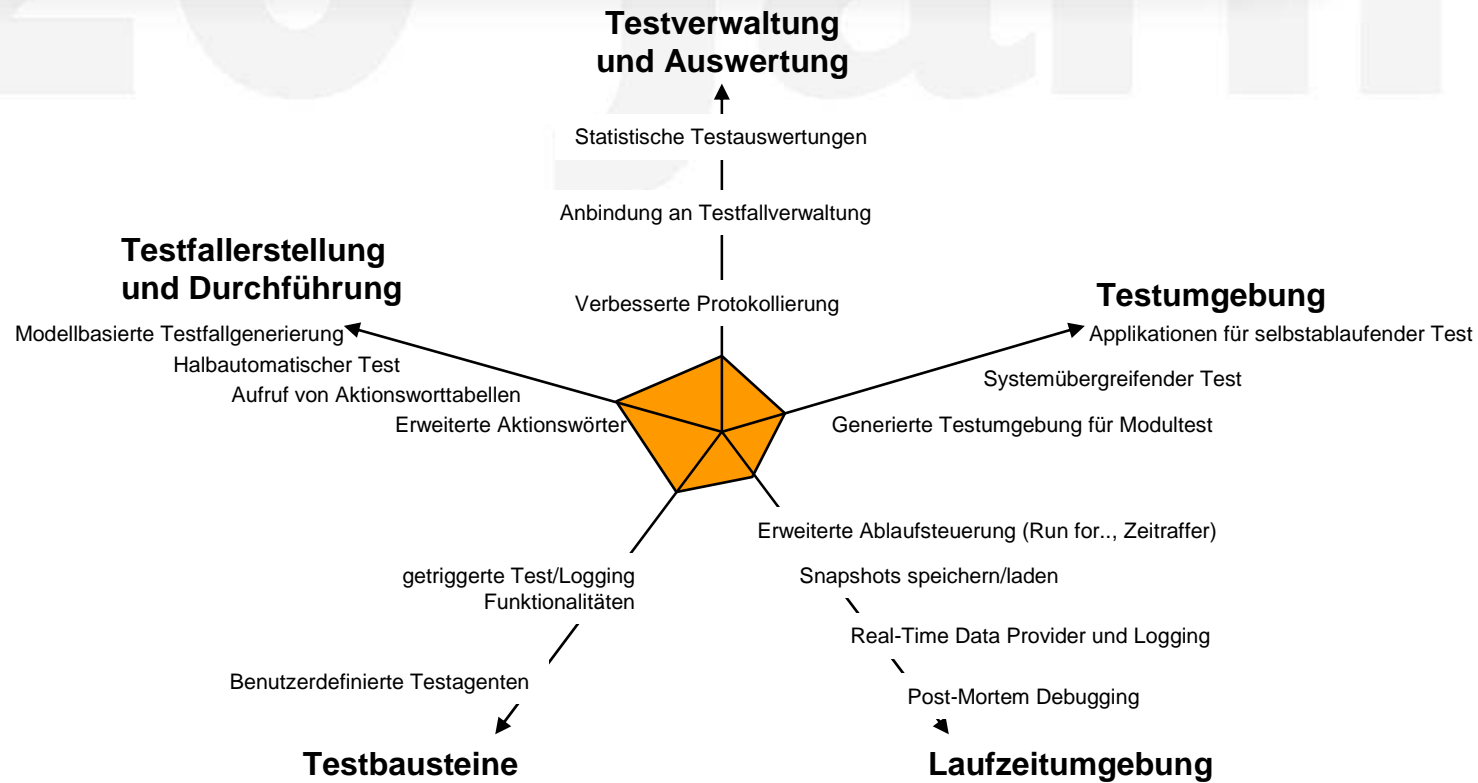
# Nutzung von Excel Funktionalitäten

Formeln

D30		=D26>10		D	E	F	G	H	I	J	K	L	M	N	O
7	Res	WAHR													
8															
9															
10	Run Test														
11															
12	Ergebnis	Command	P1	P2	P3	P4	P5								
13	WAHR	START													
14	WAHR	SET	RFOAuto	TRUE											
15	WAHR	SET	T_PumpStation.Sim_Tank.Level	16											
16	WAHR	WAIT	8000												
17	WAHR	GET	L1000_HH	FALSCH											
18	WAHR	GET	L1000_H	FALSCH											
19	WAHR	GET	L1000_L	FALSCH											
20	WAHR	GET	L1000_LL	WAHR											
21	WAHR	GET	P1000	WAHR											
22	WAHR	GET	P2000	WAHR											
23	WAHR	GET	P3000	FALSCH											
24	WAHR	GET	L1000_Sim	16,003334											
25	WAHR	WAIT	8000												
26	WAHR	GET	L1000_Sim	17,7833176											
27	WAHR	ASSERT	L1000_HH	FALSCH	FALSCH										
28	WAHR	ASSERT	L1000_H	FALSCH	FALSCH										
29	WAHR	ASSERT	L1000_L	FALSCH	FALSCH										
30	WAHR	ASSERT	L1000_LL	WAHR	WAHR										
31	WAHR	ASSERT	P1000	WAHR	WAHR										
32	WAHR	ASSERT	P2000	WAHR	WAHR										
33	WAHR	ASSERT	P3000	FALSCH	FALSCH										
34	WAHR	SET	S300	TRUE											
35	WAHR	WAIT	8000												
36	WAHR	GET	L1000_Sim	20,2791328											
37	WAHR	ASSERT	L1000_HH	FALSCH	FALSCH										
38	WAHR	ASSERT	L1000_H	FALSCH	FALSCH										
39	WAHR	ASSERT	L1000_L	WAHR	WAHR										
40	WAHR	ASSERT	L1000_LL	WAHR	WAHR										
41	WAHR	ASSERT	P1000	WAHR	WAHR										
42	WAHR	ASSERT	P2000	FALSCH	FALSCH										
43	WAHR	ASSERT	P3000	WAHR	WAHR										
44	WAHR	WAIT	8000												
45	WAHR	GET	L1000_Sim	20,1208019											
46	WAHR	ASSERT	L1000_HH	FALSCH	FALSCH										
47	WAHR	ASSERT	L1000_H	FALSCH	FALSCH										
48	WAHR	ASSERT	L1000_L	WAHR	WAHR										
49	WAHR	ASSERT	L1000_LL	WAHR	WAHR										
50	WAHR	ASSERT	P1000	WAHR	WAHR										
51	WAHR	ASSERT	P2000	FALSCH	FALSCH										
52	WAHR	ASSERT	P3000	WAHR	WAHR										
53	WAHR	WAIT	8000												
54	WAHR	GET	L1000_Sim	19,4658165											
55	WAHR	ASSERT	L1000_HH	FALSCH	FALSCH										
56	WAHR	ASSERT	L1000_H	FALSCH	FALSCH										
57	WAHR	ASSERT	L1000_L	FALSCH	FALSCH										
58	WAHR	ASSERT	L1000_LL	WAHR	WAHR										
59	WAHR	ASSERT	P1000	WAHR	WAHR										
60	WAHR	ASSERT	P2000	WAHR	WAHR										
61	WAHR	ASSERT	P3000	WAHR	WAHR										
62	WAHR	STOP													
63	Anweisungen vor dieser Zeile einfügen			Anweisungen vor dieser Zeile einfügen											



Diagramme



## Agenda

- ✓ Vorstellung
- ✓ logi.DIAG Forschungsprojekt
- ✓ Statischer Test durch Automatische Codeanalyse
- ✓ Testautomatisierung mit Keyword Driven Test

# Fragen?