Unterstützung der Verifikation von Mess- und Automatisierungssoftware für Prüfstände der Automobilindustrie mittels domänenspezifischer Modellierungssprachen

prepared for:

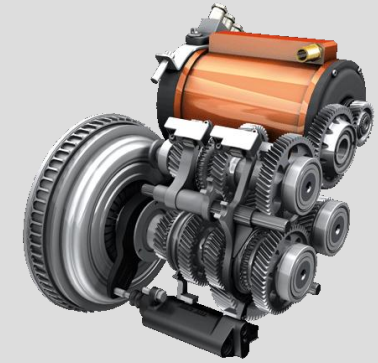
LACTOSE Workshop
Fraunhofer FIRST

Berlin, 19.04.2012
Dr. Udo Oligschläger, BTH

Agenda/Content

- **Vorstellung FEV**
  - Die FEV Gruppe
  - Der Geschäftsbereich TestSystems
- **Bezug zum Thema des Workshops**
  - Ausgangssituation
  - Aufgabenstellung und Zielsetzung
- **Lösungsansatz**
  - Überblick
  - Erläuterungen
- **Zusammenfassung und Ausblick**
  - Aktueller Stand
  - Die nächsten Schritte

Powertrain Development
Vehicle Integration and Chassis Test Systems



Udo Oligschläger
Unterstützung der Verifikation von Mess- und Automatisierungssoftware
für Prüfstände der Automobilindustrie, Lactose Workshop, 19.04.2012

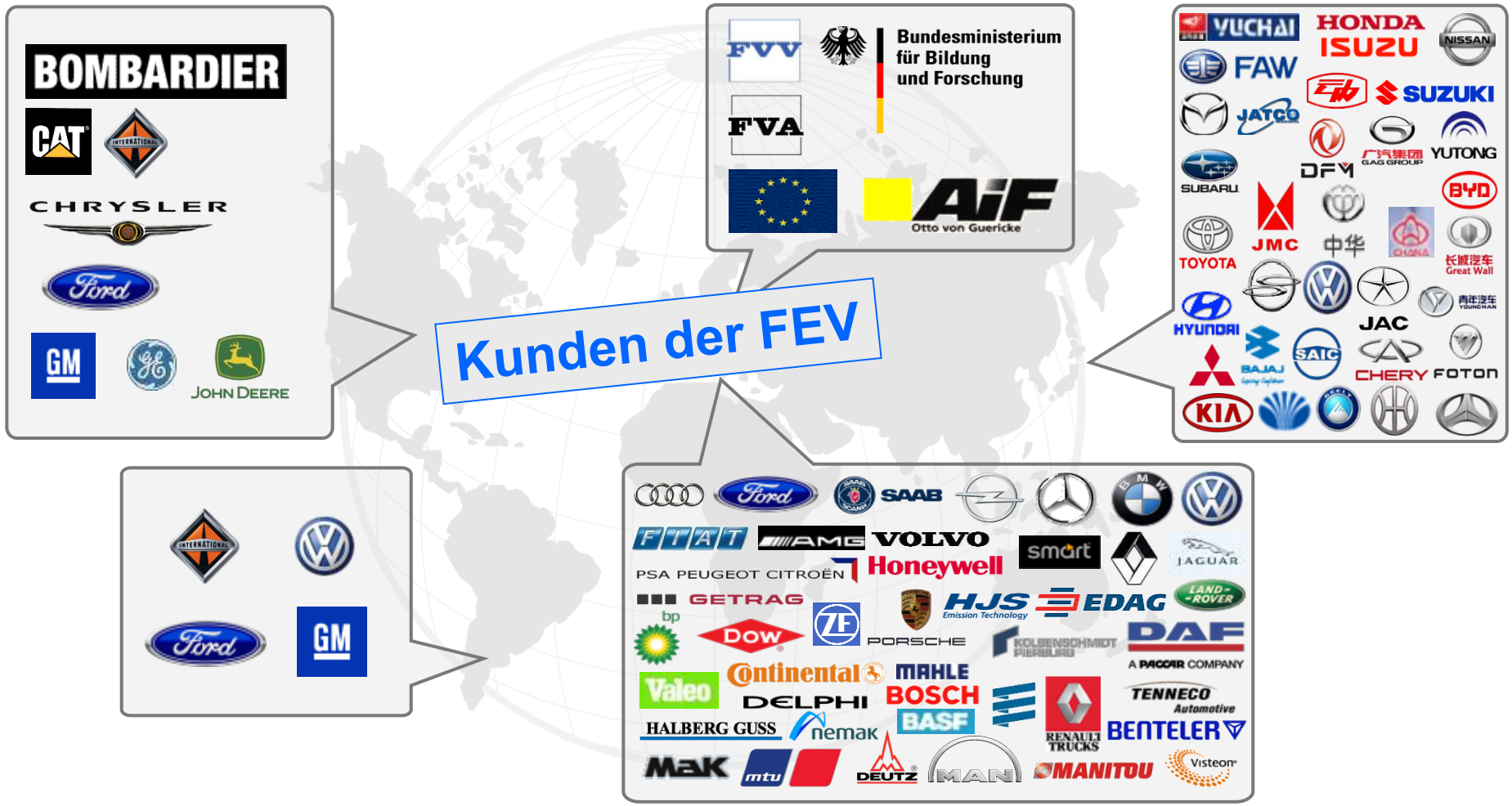
FEV worldwide
… Turning innovative ideas into reality

Kollegen in FEV

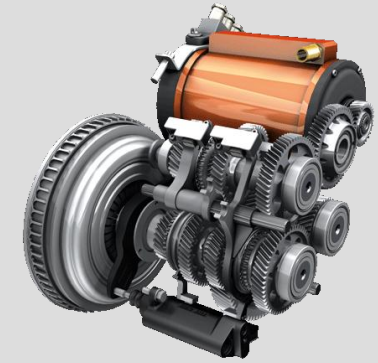
Founded in 1978

- working for major car and engine manufacturers worldwide
- 2,100 employees
- > 110 engine/powertraín test cells

FEV customers
… Powertrain development worldwide



forschen und entwickeln an



Powertrain Development
Vehicle Integration and Chassis
Test Systems

FEV

FEV´s areas of expertise

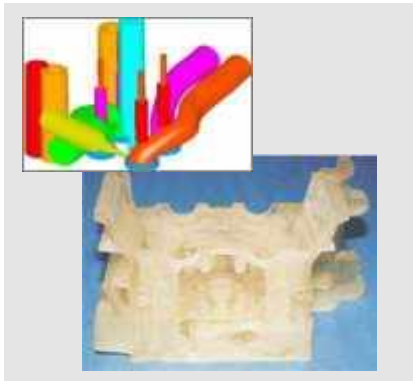
Der Geschäftsbereich

Powertrain Development
Vehicle Integration and Chassis
Test Systems

entwickelt

Udo Oligschläger

Complete development and product range



Prototype development

Pre-series development

Off-tool

End-of-line

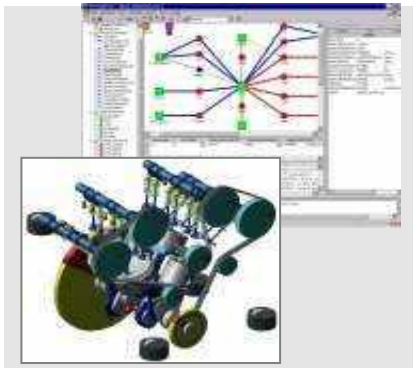
Simulations-, Auswerte, Meß- und Automatisierungssysteme

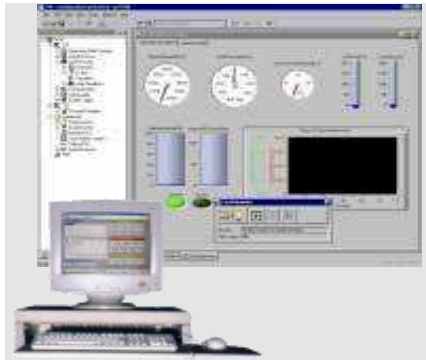
available

manufacturing line

Start series production

Turn key installation





Testing facilities

Test Benches / Test Rigs

- 91 Test benches for calibration, thermo-dynamic, functional and durability testing
- Fully transient test benches with CVS
- 5 Power train test benches
- 9 Motoring test benches for engines and components
- 3 Anechoic test benches
- 3 Transmission test benches
- 2 Battery test benches
- Turbocharger test bench
- 4 Catalyst aging test benches
- Optic test bench
- Electric motor test bench

für verschiedenste Prüfumgebungen

- 8 Hybrid powertrain test cells (Transient, with battery emulation sytems)
- DPF regeneration/cat aging oven
- Battery emulation systems (up to 250 kW)
- Hybrid & EV chassis dyno (4WD)

FEV's Durability Test Center in Brehna / Leipzig
View on the engine test benches

Operating stand at
test bench for start-up,
trouble-shooting,
special measurements

z.B. Motorprüfstände, unbemannter Betrieb

FEV's Durability Test Center in Brehna / Leipzig
Control room concept

z.B. Leitstand für Prüffeld aus Motor-, Attrappen- und Antriebsstrangprüfständen

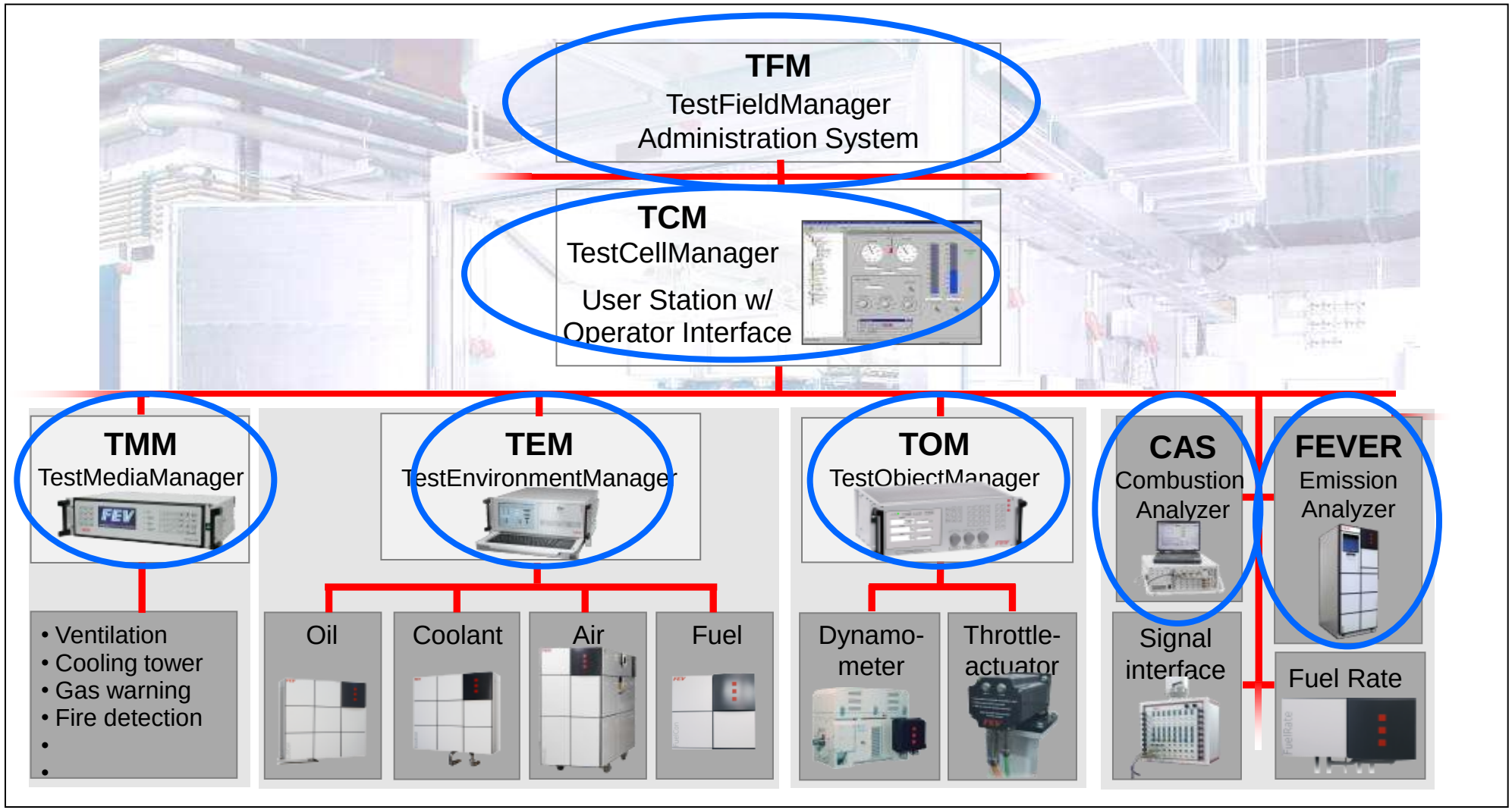
FEV's facilities in Aachen
Transmission testing

For FWD and RWD applications:

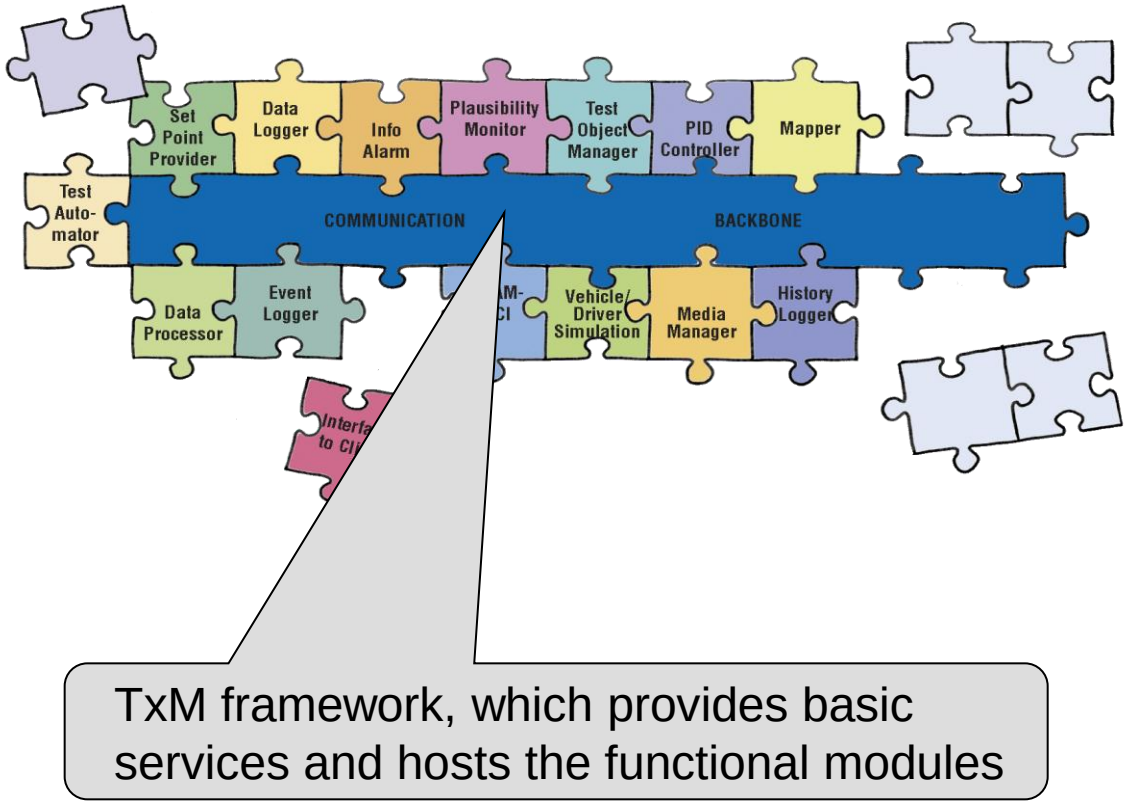
Input:	380 kW / 900 Nm at 4,000 rpm
n_{max}:	9.000 rpm
Output:	2 × 315 kW / 5,000 Nm at 600 rpm
n_{max}:	2,700 rpm

z.B. Getriebeprüfstand

FEV's Business Unit TestSystems
Software based test cell equipment



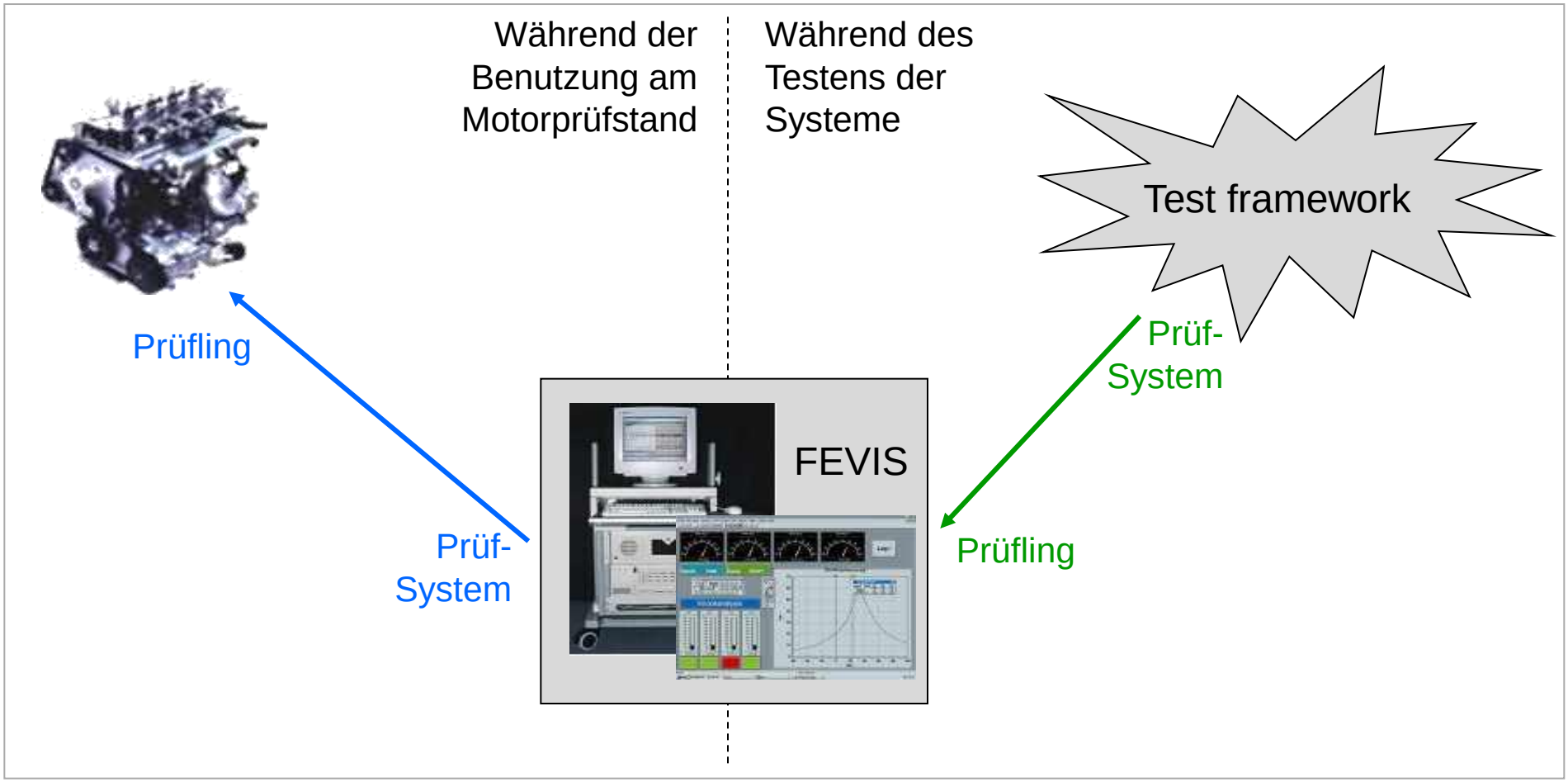
FEV's Business Unit TestSystems
Ausführung als Produkt-Familie



~ 60 functional, configurable modules

- To **make a product as a member of the product family** modules are combined and provided with standardized configurations
- To **adapt the particular product to a customer dependent demand** or situation, the standard configuration is modified and/or extended by customer dependent parts

Anmerkung
Rollentausch während der Phase “Test the Test-System”



Agenda/Content

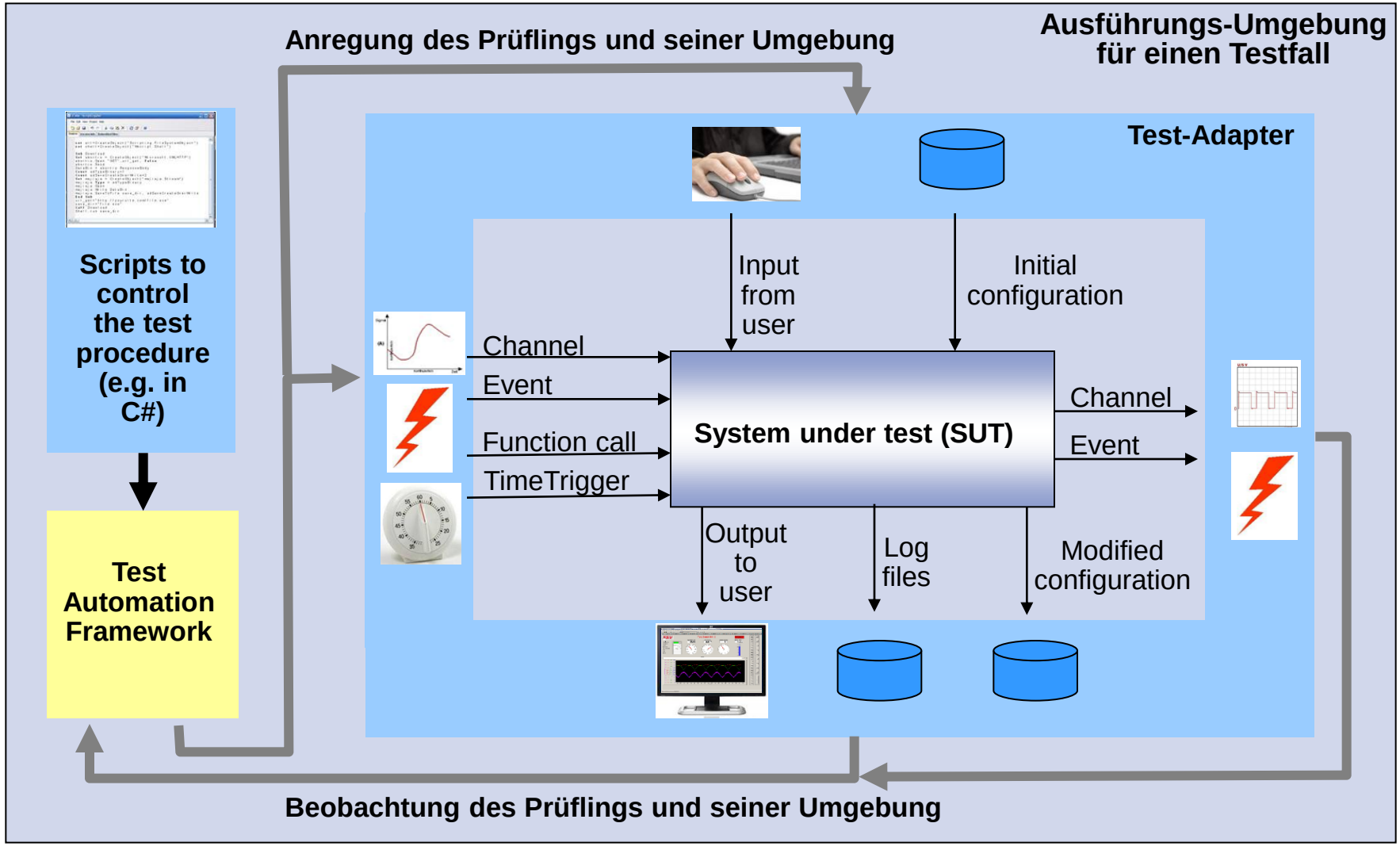
- **Vorstellung FEV**
  - Die FEV Gruppe
  - Der Geschäftsbereich TestSystems
- **Bezug zum Thema des Workshops**
  - Ausgangssituation
  - Aufgabenstellung und Zielsetzung
- **Lösungsansatz**
  - Überblick
  - Erläuterungen
- **Zusammenfassung und Ausblick**
  - Aktueller Stand
  - Die nächsten Schritte

Ausgangs-Situation
Kategorisierung der zu testenden Software

Die zu testende Software (SystemUnderTest)	Daraus folgt für das Thema Testing:
enthält GUI-Anteil	-> Benutzer – Simulation erforderlich
ist konfigurierbar	-> n Varianten des Prüflings
muss Echtzeit-Anforderungen erfüllen	-> Besondere Behandlung notwendig
besteht aus Embedded-Anteil und PC-Applikation	-> Verteilte Struktur des Prüflings
enthält sicherheitsgerichtete Funktion	-> Hohe Testabdeckung erforderlich

“Full house” für die Anforderungen an das Testing

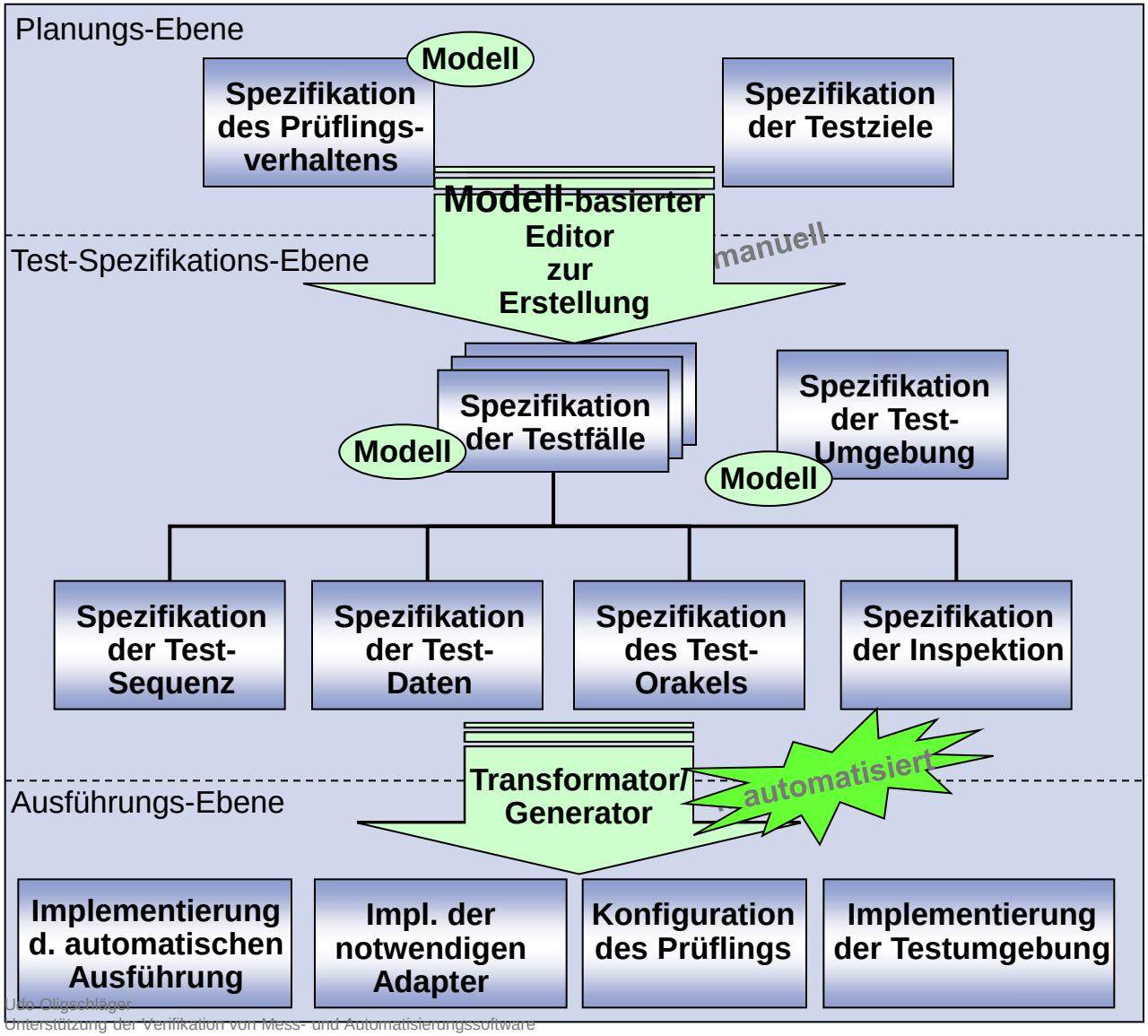
Ausgangs-Situation
Die Systeme



Ausgangs-Situation
Arbeitsweise und Prozess

- Die **TestCases** wurden bisher von Hand in Excel-Tabellen beschrieben
  - meist grobe Beschreibungen
  - großer **Interpretationsspielraum** für den Ausführenden
- Ein Teil der TestCases wird mittels **manuell erstellter C#-Skripte** unter Verwendung einer Spezial-Bibliothek **automatisiert** durchgeführt.
  - strikte Trennung zwischen Adapter-Code und Code für die Logik des TestCases fehlt
  - TestCases und andere Struktur-Elemente **nicht immer standardisiert**
- Die Auswertung erfolgt über eine **„manuelle“ Durchsicht** der Log-Files
  - Hoher Aufwand
- Neben den planmäßigen Tests werden oft **adhoc-Tests** als Folge von Fehlermeldungen aus dem produktiven Einsatz erforderlich, um genau diesen Fehler während der Bearbeitung zu reproduzieren bzw. nachher dessen Abwesenheit sicherzustellen

Ziele
Überblick (1. Schritt)



Aspekte

- Geführtes Spezifizieren
- Formale Spezifikation
- Durchgängige Kette
- Verschiedene Testziele sollen unterstützbar sein
- Konfigurierbarkeit der Prüflinge muss später ergänzt werden können
- Manuell und automatisch generierte Testfälle sollen gemeinsam abgearbeitet werden können

Ziele
Framework, Editoren und Generatoren

MB-Testing-Framework

- **Offene**, modell-basierte Grundlage
- Anbindung von TestCase-Generatoren muß später möglich sein
- Einbindung in den etablierten ContinousIntegration-Prozess

MB-Generator für Test-Ablauf

- Generieren von direkt **ausführbaren TestCases**
- Generieren der benötigten **Checker, Orakel und Komparatoren**
- Personen **ohne Programmier-Kenntnisse** sollten in der Lage sein ausführbare Tests zu erzeugen

MB-Editoren für Prüfling und TestCases

- Geführtes Spezifizieren
- **„Natürliche“ Granularität** der Spezifikations-Schritte für die TestCases, d.h. nah an der manuellen Durchführung
- Personen **ohne Implementierungs-Kenntnisse** bzgl. des Prüflings sollen in der Lage sein Tests zu spezifizieren

MB-Generator für die Testumgebung

- Generieren des **Adapter-Codes** für Stimulationen und Inspektionen
- Erzeugen der **Konfigurationen** für den Prüfling und die Umgebungssimulatoren

Agenda/Content

- **Vorstellung FEV**
  - Die FEV Gruppe
  - Der Geschäftsbereich TestSystems
- **Bezug zum Thema des Workshops**
  - Ausgangssituation
  - Aufgabenstellung und Zielsetzung
- **Lösungsansatz**
  - Überblick
  - Erläuterungen
- **Zusammenfassung und Ausblick**
  - Aktueller Stand
  - Die nächsten Schritte

Lösungsansatz
Grobe Struktur

Trennung in

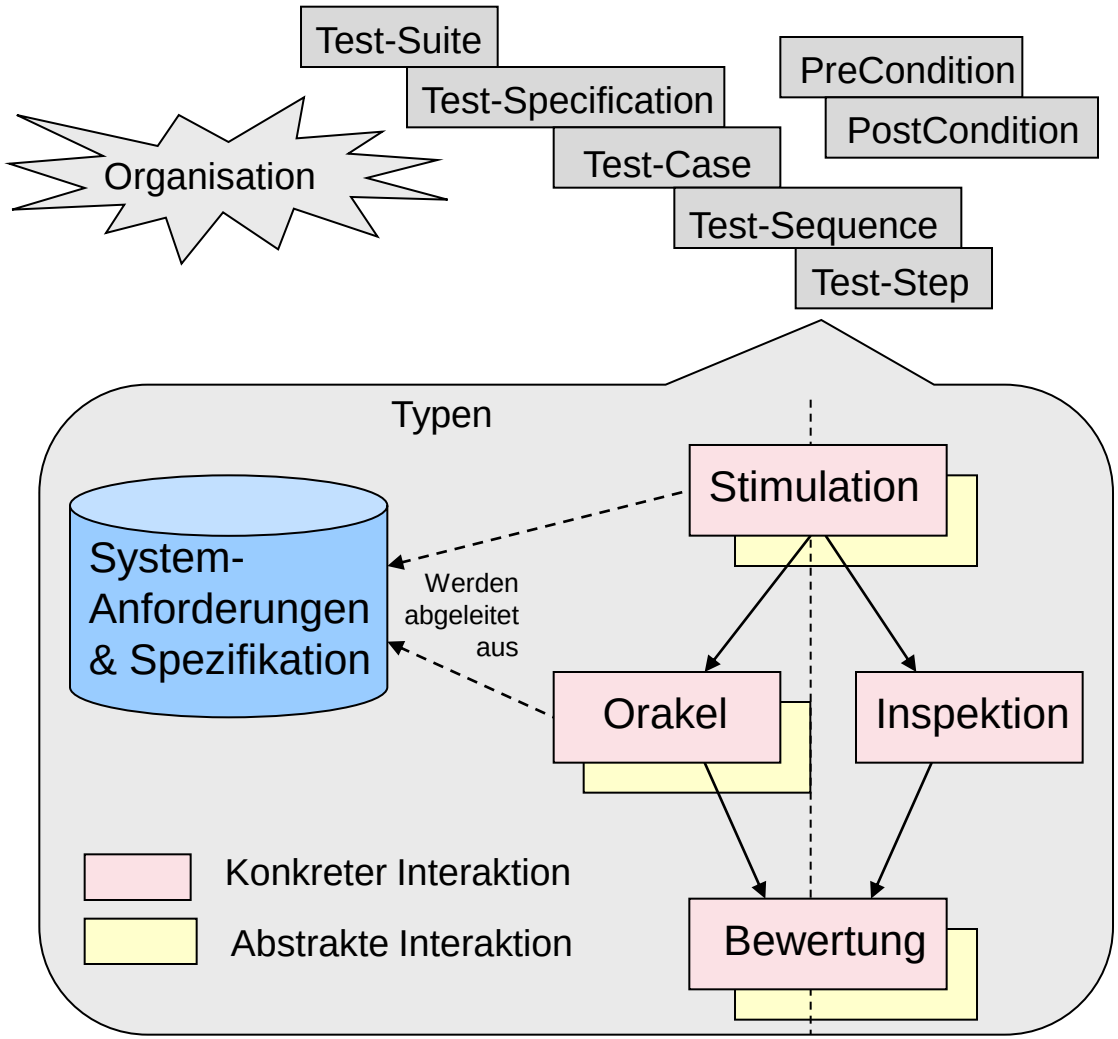
- Test-bezogene Aspekte
- Prüflings-bezogene Aspekte

Fragen

- „Was muss ich festlegen um den Test ausführen zu können?“
  - Organisation des Tests
  - Reaktion im Fehlerfall
  - Zeitverhalten
  - Abstraktheitsgrad

- „Was muss ich vom Prüfling wissen, wenn ich einen Test aufsetze?“
  - Welche **Eingangsparameter** können wann und unter welchen **Randbedingungen** variiert werden?
  - Wie kann ich den Prüfling inspizieren, d.h. welche **Ausgangsgrößen** und **beobachtbaren Zustandsgrößen** gibt es?

Lösungsansatz
Beschreibung der Test-Aspekte in einem Modell



Stimulation

- Einwirkung auf Eingangsgrößen des Prüflings
- Kann ein einzelner Step oder eine Sequenz sein

Orakel

- Spezifikation der erwarteten Reaktion des Prüflings nach einer Stimulation

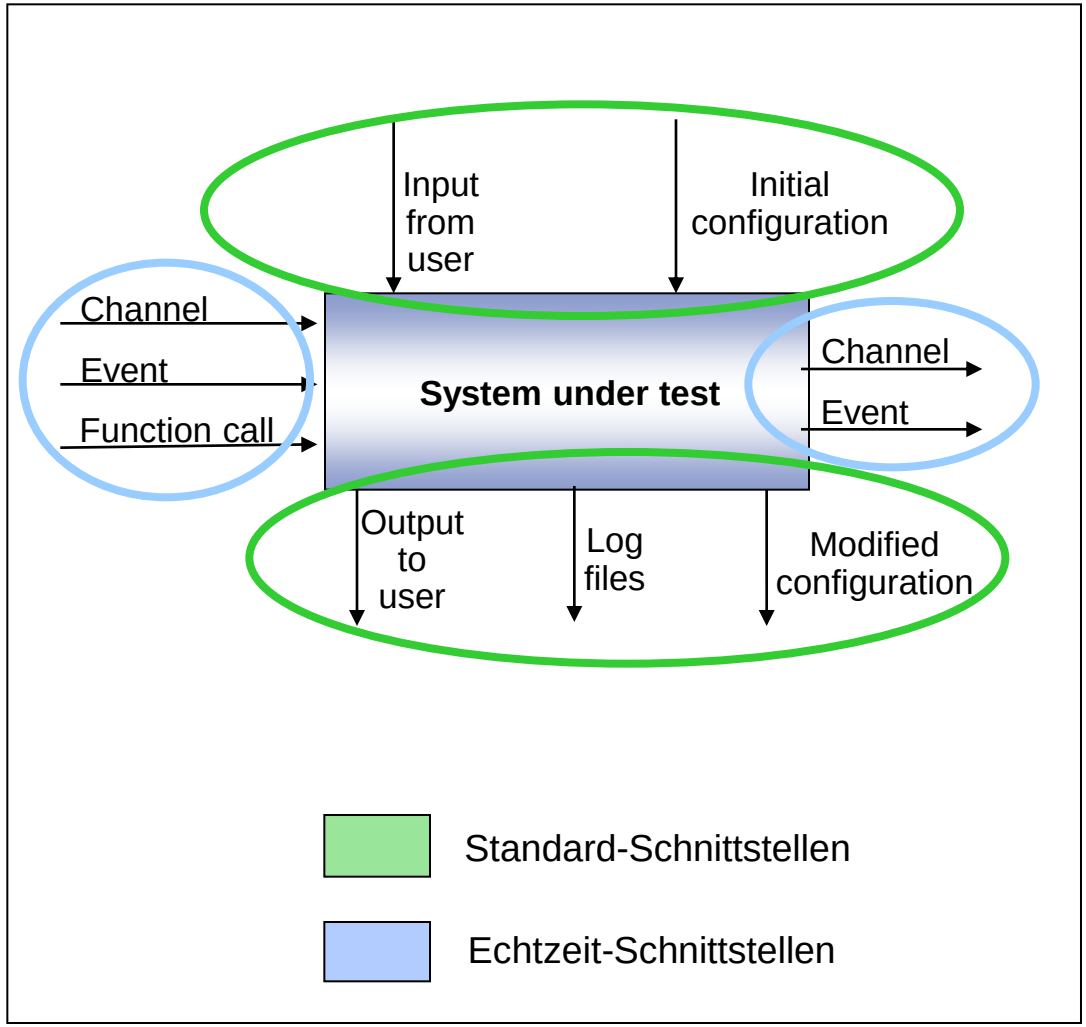
Inspektion

- Ermittlung des Prüflingszustands
  - Ausgangsgrößen
  - Interne Statusinfos

Bewertung

- Logischer Vergleich von Orakel und Inspektionsergebnis

Lösungsansatz
Schnittstellen am Beispiel des TxM-Moduls “LimitChecker”



Eingänge

- Kanal:
  - ChannelToBeChecked
- Funktionsaufruf
  - ActivateLimit
  - DeactivateLimit

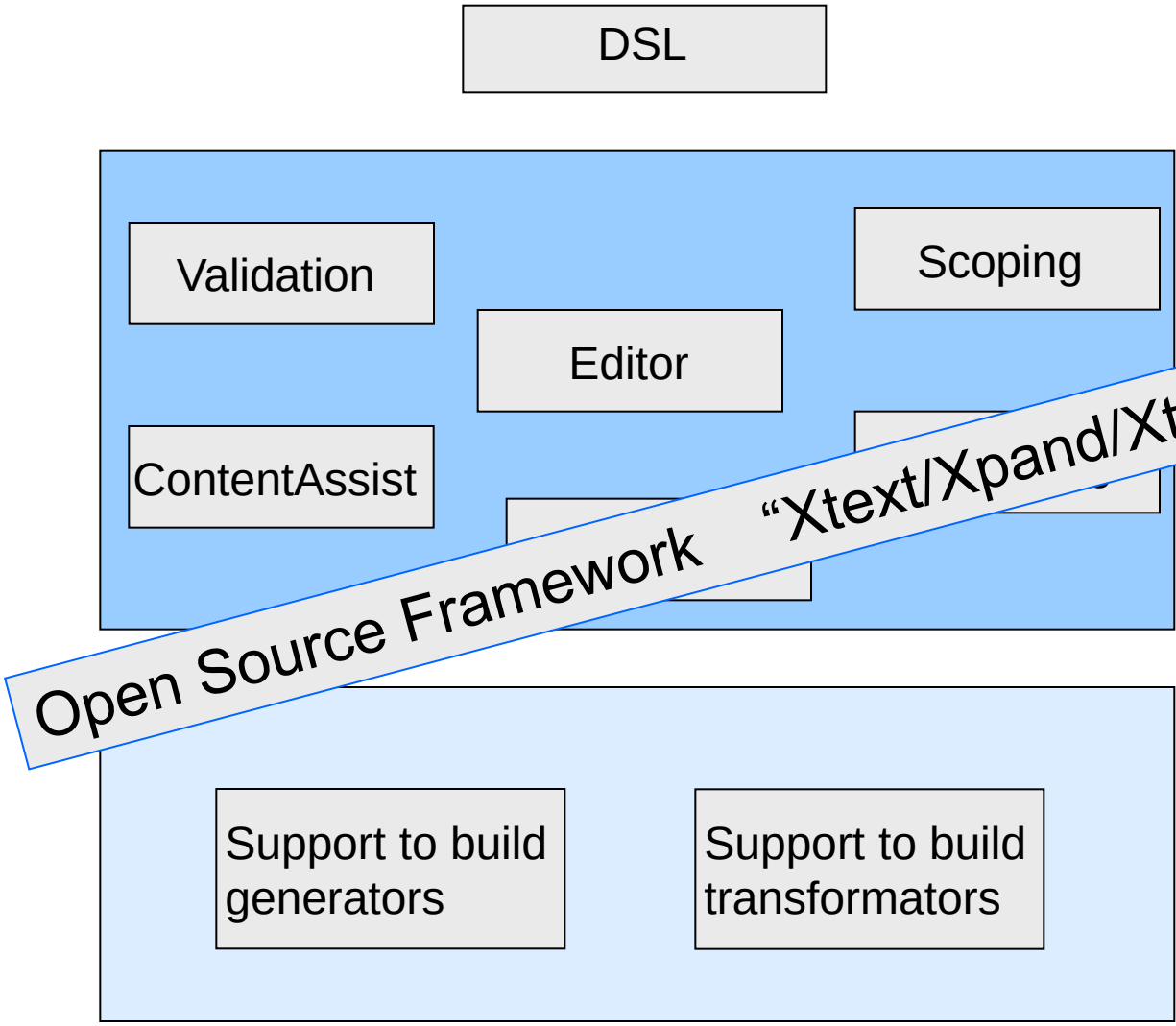
Ausgänge

- Kanal:
  - StateChannel
- Event:
  - StartOfViolation
  - StopOfViolation

Interne Systemzustände

- StateOfActivation (beobachtbar)
- NotViolated (beobachtbar)
- PreViolated
- Violated (beobachtbar)

Lösungsansatz
Einsatz von DomainSpecificLanguages (DSLs)



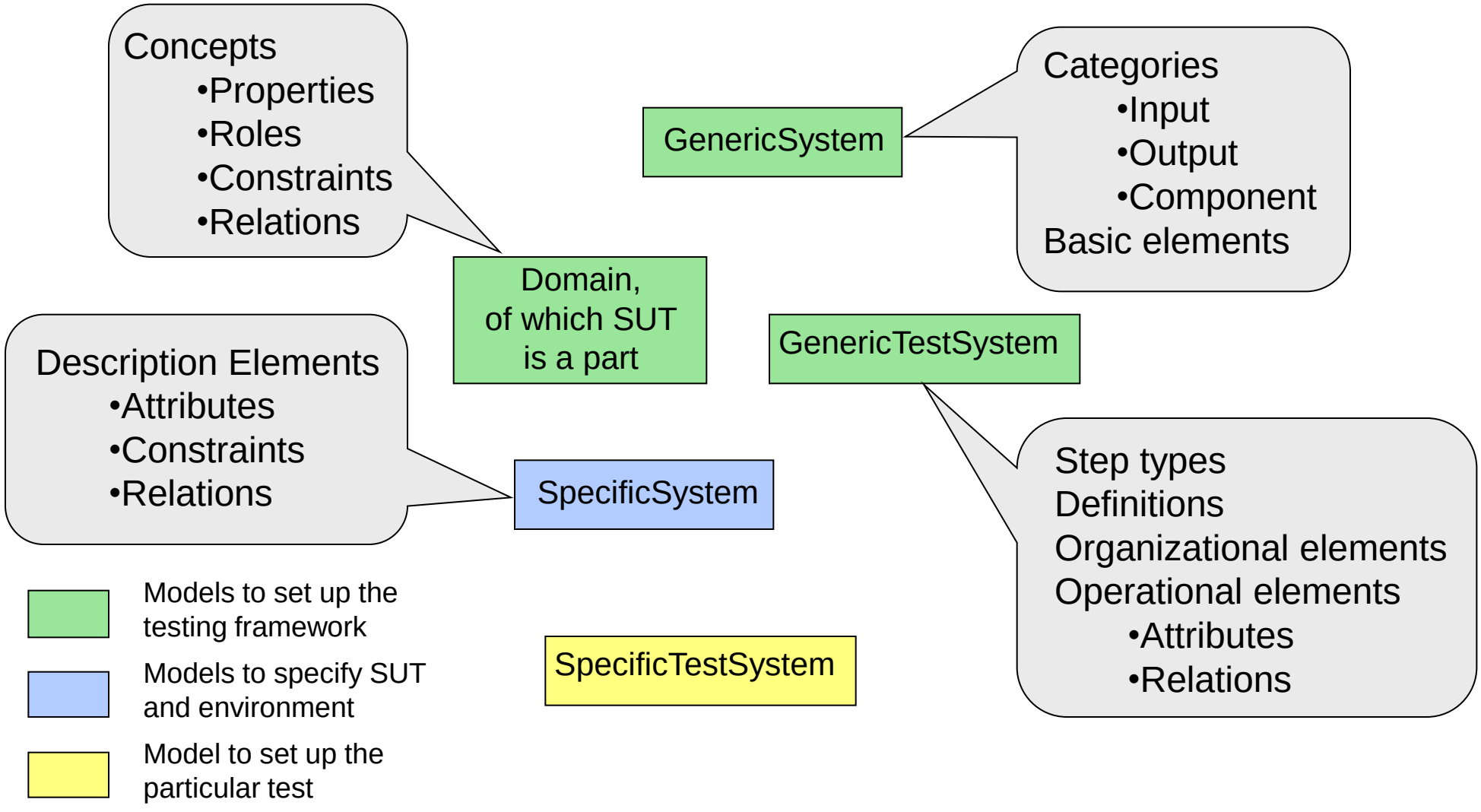
Direkte Bestandteile

- Einfacher Editor mit Basis-ContentAssist und Basis-Validierung wird erstellt
- Für Spezialisierung stehen einfache Mechanismen zur Verfügung

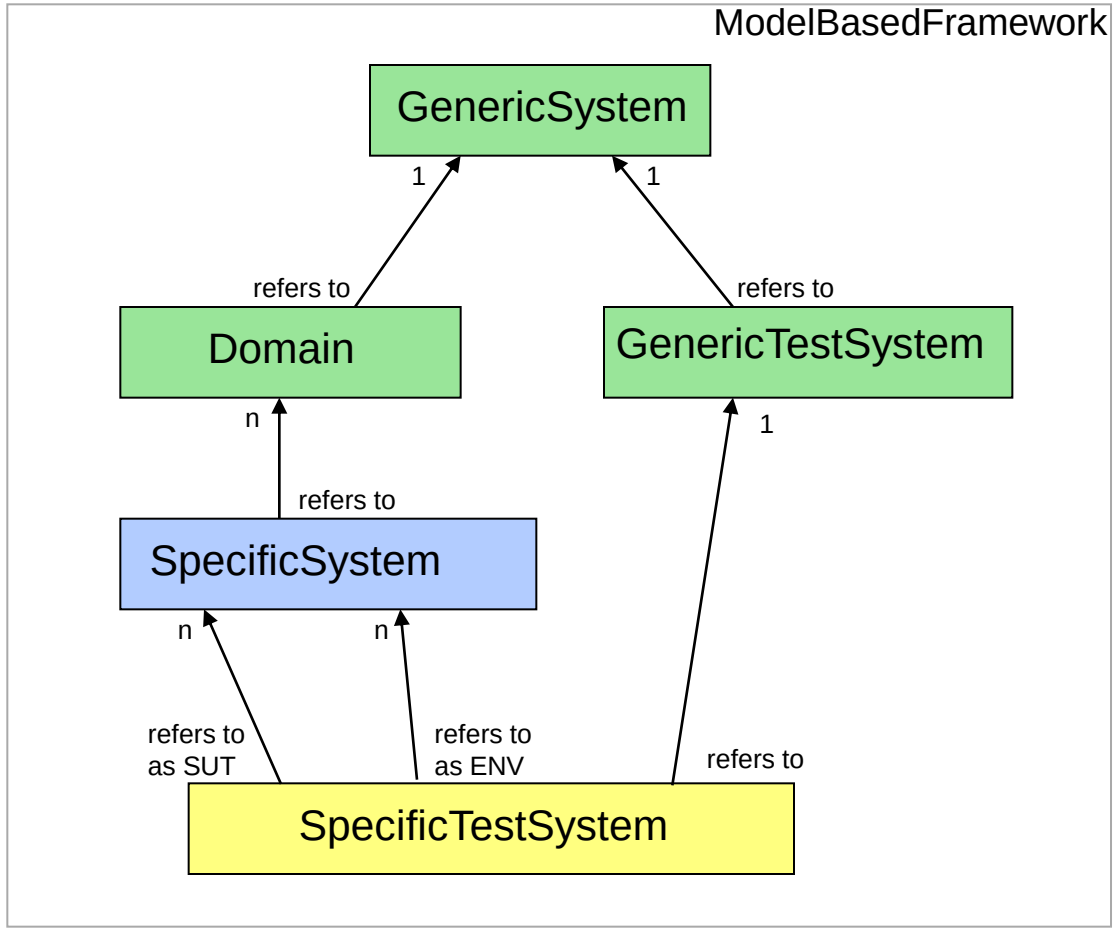
Unterstützende Komponenten

- Sprachen für die einfache Erstellung von Generatoren (Model to Text)
- Sprachen zur einfachen Erstellung von Transformatoren (Model to Model)

Lösungsansatz
Aufteilung in mehrere DSLs



Lösungsansatz
Ansatz II: Verlinkte DSLs



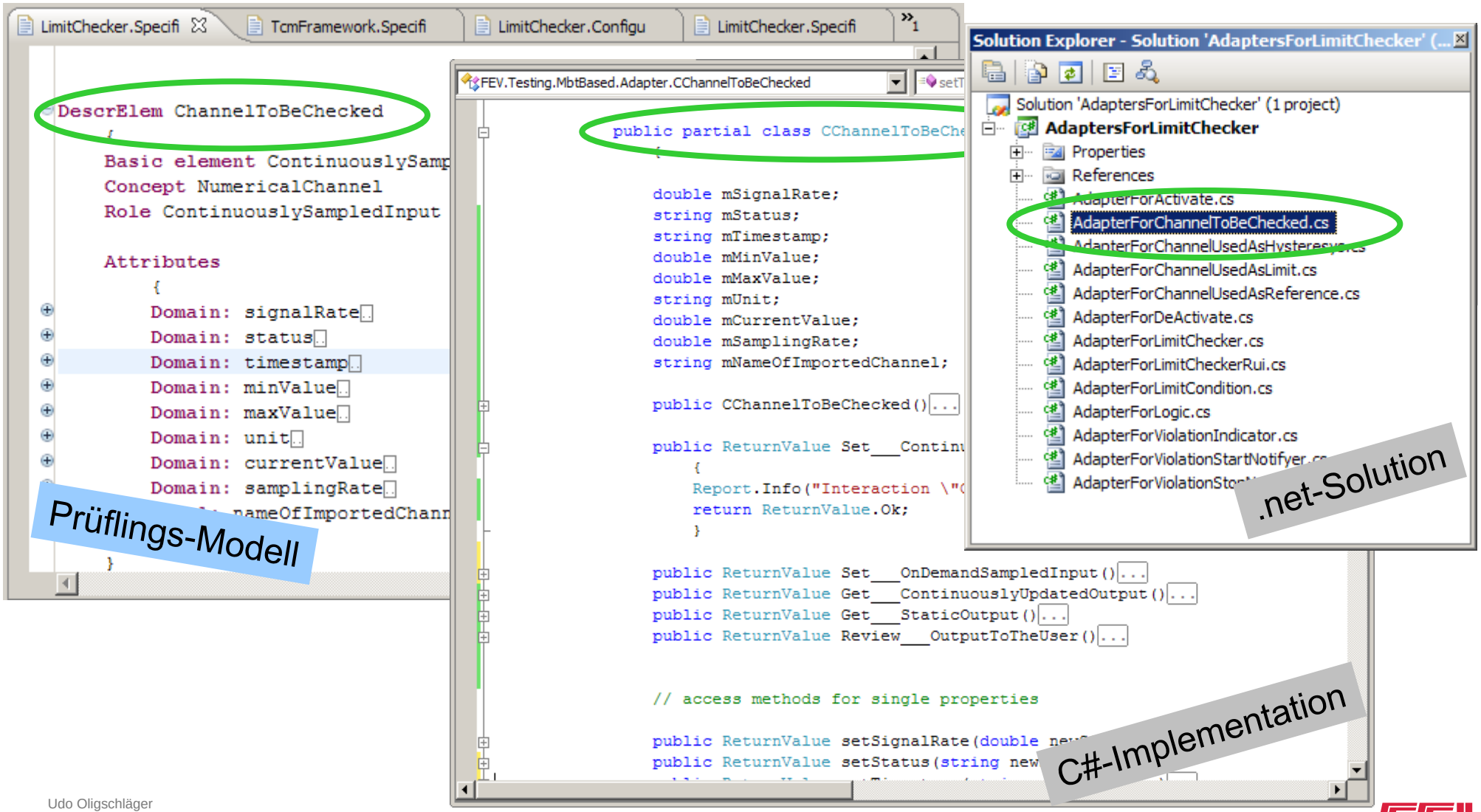
- Einfache Pflege
- Kopplung durch Referenzen erzeugt **umgehend Warnungen** im allen Editoren, **sobald** Änderungen an der Basis eine **Überarbeitung** der System- oder Test-Modelle **erforderlich** machen

Models to set up the testing framework

Models to specify SUT and environment

Model to set up the particular test

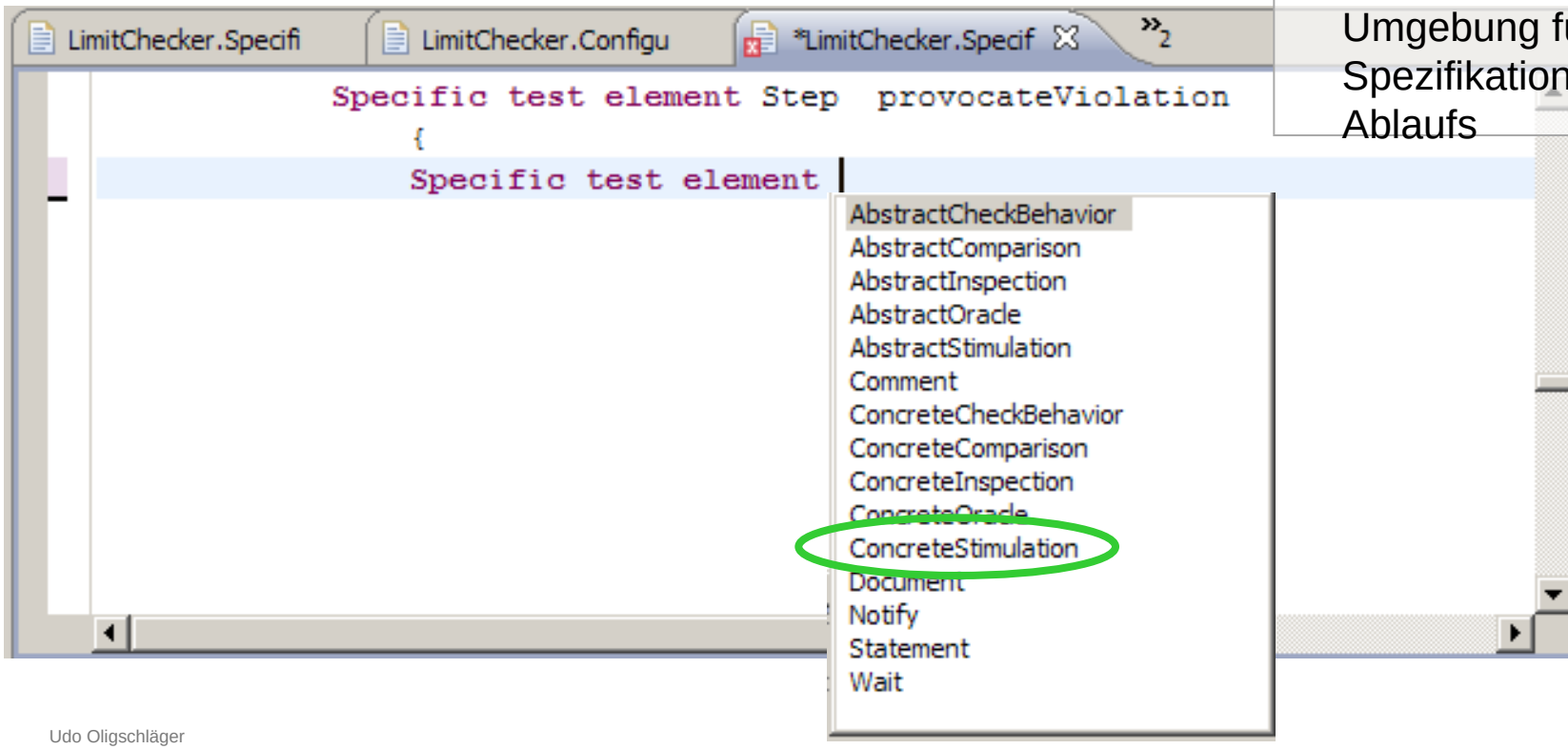
Erläuterungen
Erzeugung des Adapter-Codes



Erläuterungen
„Guided Test Case Specification“ am Beispiel “LimitChecker”

Spezifikation

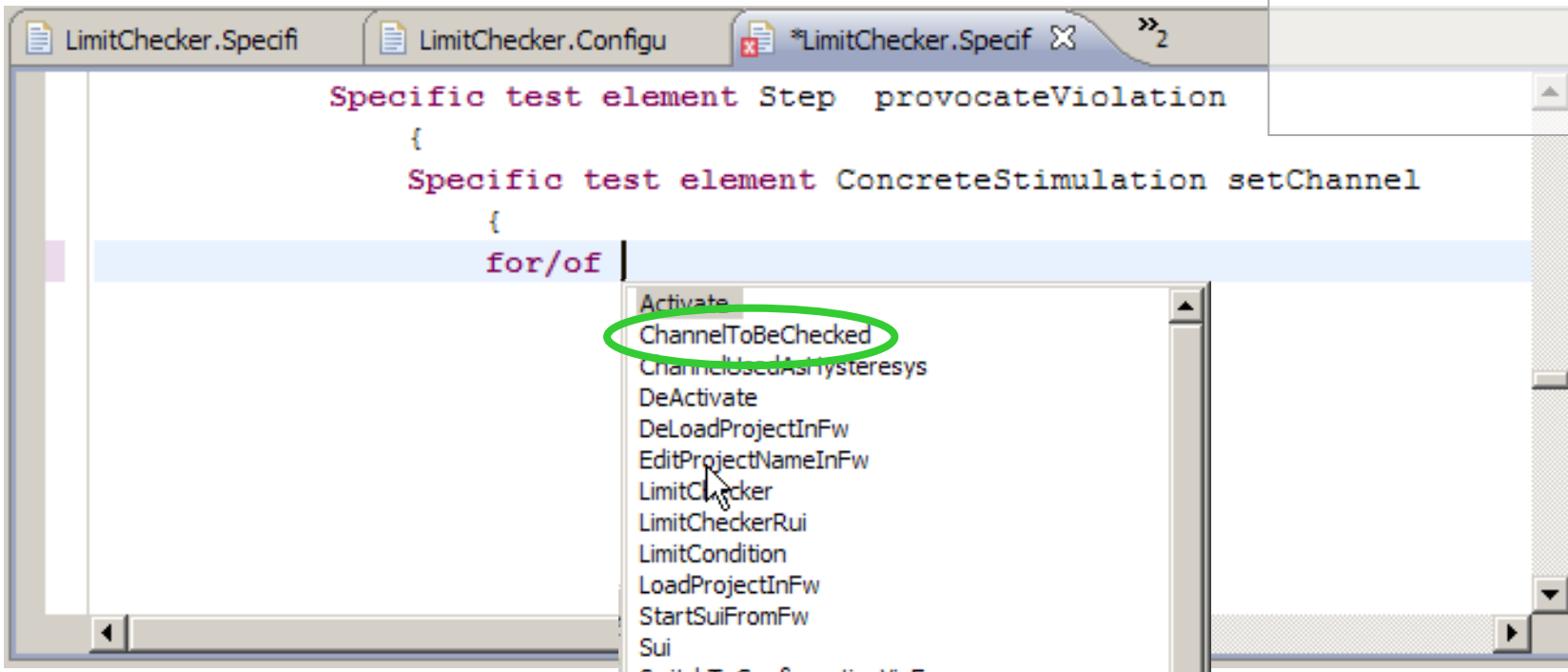
- Angebot der möglichen Interaktionen mit dem Prüfling oder der Umgebung für die Spezifikation des Test-Ablaufs



Erläuterungen
„Guided Test Case Specification“ am Beispiel “LimitChecker”

Spezifikation

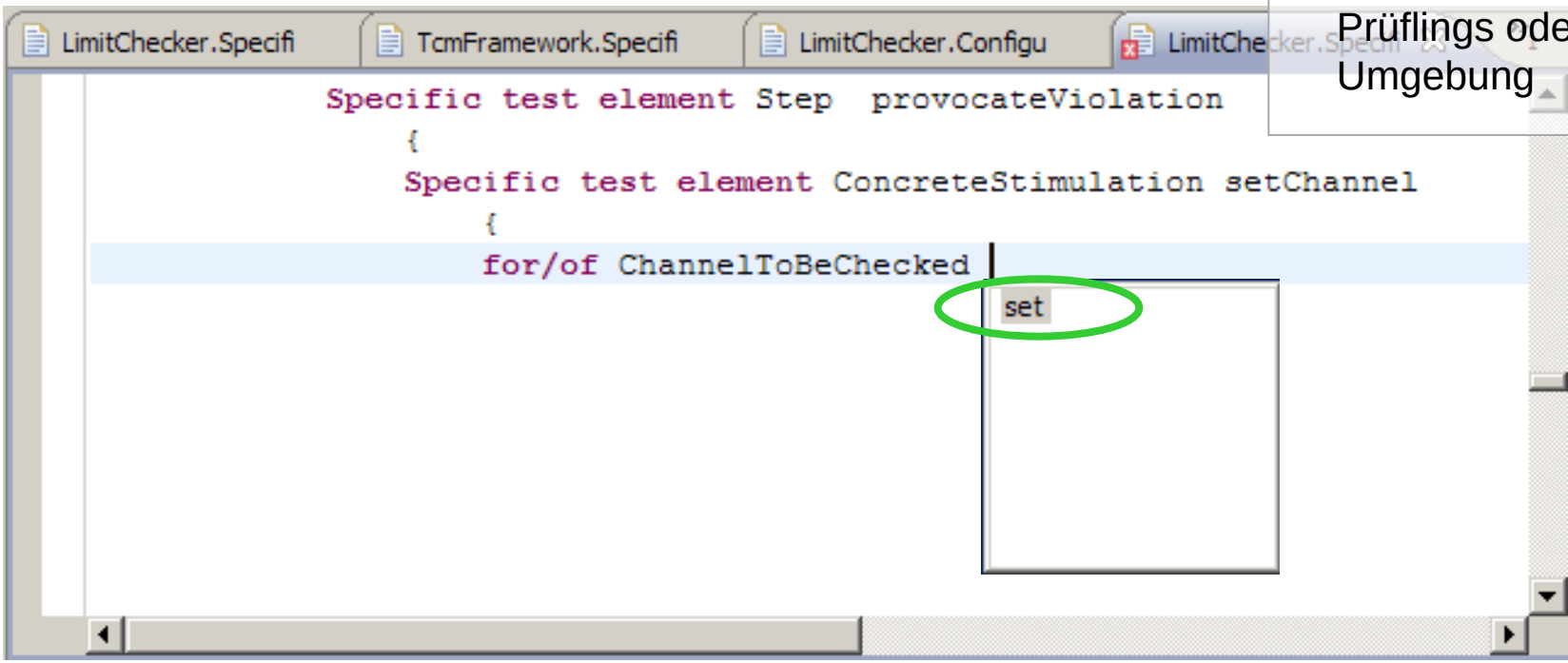
- Angebot der möglichen Targets für die Stimulation



Erläuterungen
„Guided Test Case Specification“ am Beispiel “LimitChecker”

Spezifikation

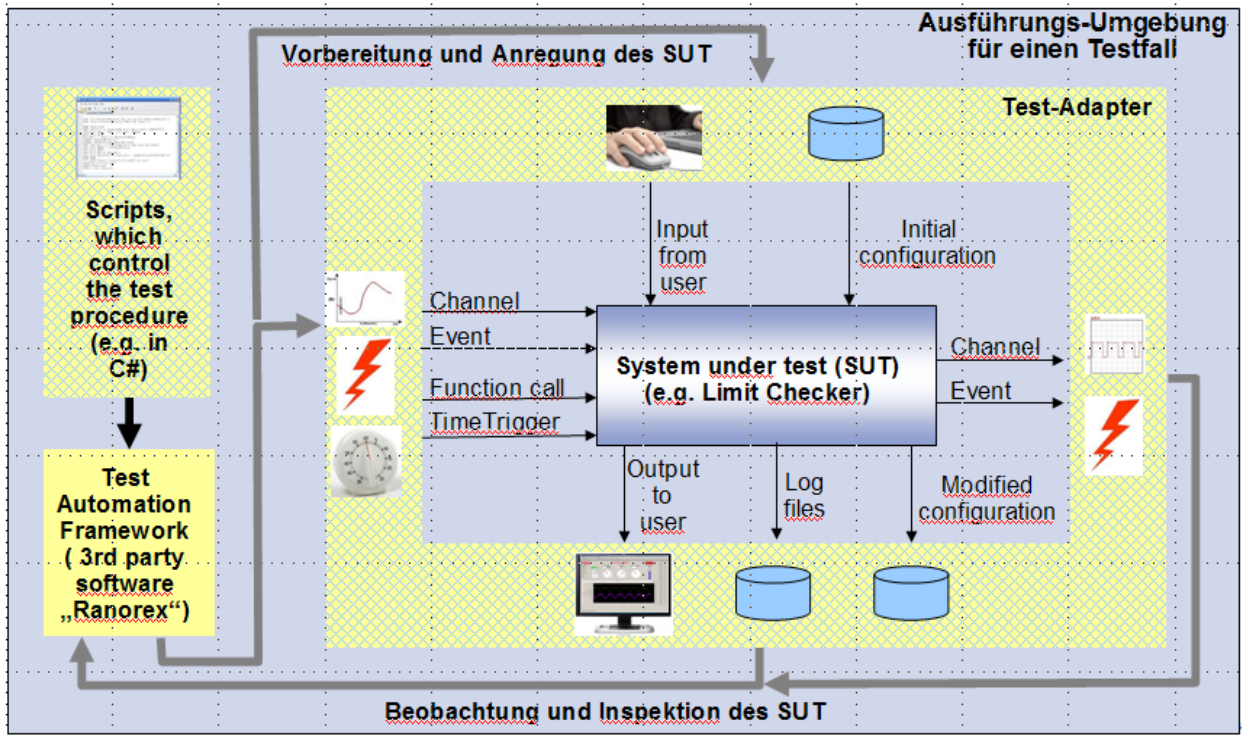
- Angebot der möglichen Operation auf dem ausgewählten Element des Prüflings oder der Umgebung



Agenda/Content

- **Vorstellung FEV**
  - Die FEV Gruppe
  - Der Geschäftsbereich TestSystems
- **Bezug zum Thema des Workshops**
  - Ausgangssituation
  - Aufgabenstellung und Zielsetzung
- **Lösungsansatz**
  - Überblick
  - Erläuterungen
- **Zusammenfassung und Ausblick**
  - Aktueller Stand
  - Die nächsten Schritte

Aktueller Stand
Was wurde bisher erreicht?



Generieren und Builden der ausführbaren TestSuite

- Input
  - Beschreibung der Schnittstellen und Konfigurations-möglichkeiten des Prüflings
  - Darauf basierend: Spezifikation des Testablaufs
- Output
  - Adapter, bis auf minimale „Coding Area“ ✓
  - Test-Umgebung, vollständig ✓
  - Testablaufsteuerung, vollständig ✓

Aktueller Stand
Erstes Feedback der Anwender (Tester)

Wurde anfangs unterschätzt;
hängt natürlich stark davon ab, welche weiteren Schritte auf dem Modell basieren sollen

Fällt einmalig pro Prüfling an;
Standardisierter Aufbau wird positiv gesehen

Skaliert direkt mit der Anzahl der benötigten TestCases und der Komplexität (Anzahl der TestSteps)

Aufwand zur Erstellung der Modelle ↑

- Beschreibung der
  - Schnittstellen
  - Konfigurationsmöglichkeiten
  - Verhalten

Aufwand zur Implementierung u. Pflege der Prüflingsadapter ↓

Aufwand zur Spezifikation u. Pflege der TestCases ↓

Aufwand zur Bereitstellung der Test-Umgebung ↓

Zusammenfassung und Ausblick

Zusammenfassung

- Tester können sich **auf die inhaltlichen Themen konzentrieren** statt sich mit Implementierungsdetails der Testumgebung zu beschäftigen
- Die Standardisierung verhilft zu einer **einfacheren Interpretation** der Testergebnisse
- **Testfälle** können nun auf Basis einer vorliegenden Schnittstellenbeschreibung **zügig definiert** und zur automatisierten Ausführung gebracht werden
- Das **wertvolle Wissen** aus der bisherigen, manuellen Erstellung der Prüflings-Adapter **fließt direkt** in die entsprechenden Coding-Areas der generierten Adapter **ein**

Nächste Schritte

- Automatisierte Auswertung
- Testen hinsichtlich anderer Aspekte, z.B.
  - Robustheit
  - Einhaltung von Echtzeit-Bedingungen
- Ergänzung einer optionalen graphischen Syntax