Modellbasierter Entwurf SPS-basierter elektronischer Stellwerke

Prof. Dr. Wilhelm (Willi) Hasselbring

Lehrstuhl Software Engineering
https://se.informatik.uni-kiel.de/
&
Kompetenzverbund
Software Systems Engineering (KoSSE)
https://kosse-sh.de/

Tagung Softwarearchitektur 2011, Hamburg 20.07.2011

Der Kontext: KoSSE-Projekt MENGES

M	odellbasierte
E	ntwurfsmethoden für eine
N	eue
G	eneration
E	lektronischer
S	tellwerke

Das Team, welches die Inhalte dieses Vortrages erarbeitet hat:

AG Software Engineering, Universität Kiel
Wilhelm Hasselbring, Reiner Jung
AG Echtzeitsysteme und Eingebettete Systeme, Universität Kiel
Reinhard von Hanxleden, Christian Schneider

Funkwerk Information Technologies GmbH, Kiel
Holger Neustock, Heiko Schaefer, Elferik Schultz,
Hauke Fuhrmann, Uwe Klose

b+m Informatik AG, Melsdorf
Wolfgang Goerigk, Thomas Stahl,
Gregor Hennings, Steffen Weik, Stefan Zeug

MENGES ist ein Projekt im
Kompetenzverbund Software Systems Engineering KoSSE

Verbundprojekte im Themenbereich des Software System Engineering der Universitäten Kiel und Lübeck mit Unternehmen aus Schleswig-Holstein

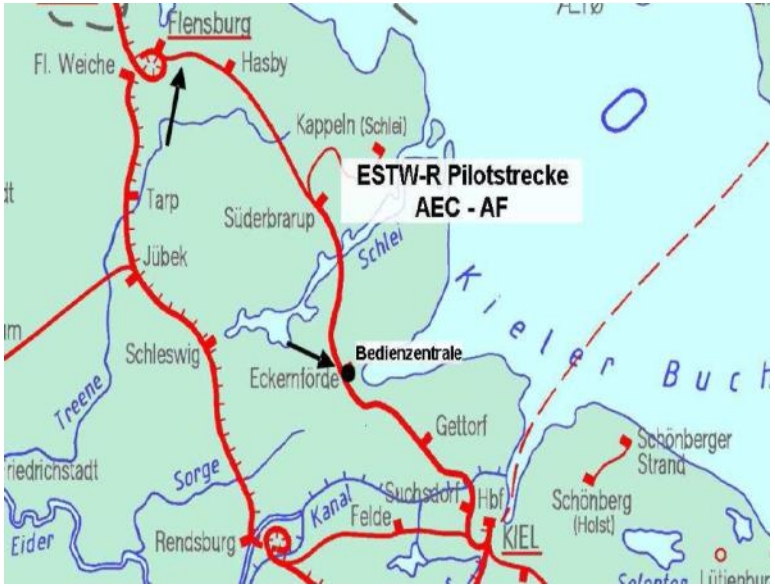
http://www.kosse-sh.de/

ZUKUNFTSprogramm
Schleswig-Holstein
Investition in Ihre Zukunft

MENGES: Motivation und fachliche Domäne

Sicherer Bahnbetrieb

- Elektronische Stellwerke, Gleise, Weichen, Signale, Fahrwege, …
- Hohe Sicherheitsanforderungen, technische Sicherheit, Verfahrenssicherheit, Betriebssicherheit, technisch verfahrensgesicherter Bedienplatz
- Zertifizierung nach CENELEC, SIL-4, sehr hohe Zertifizierungsaufwände

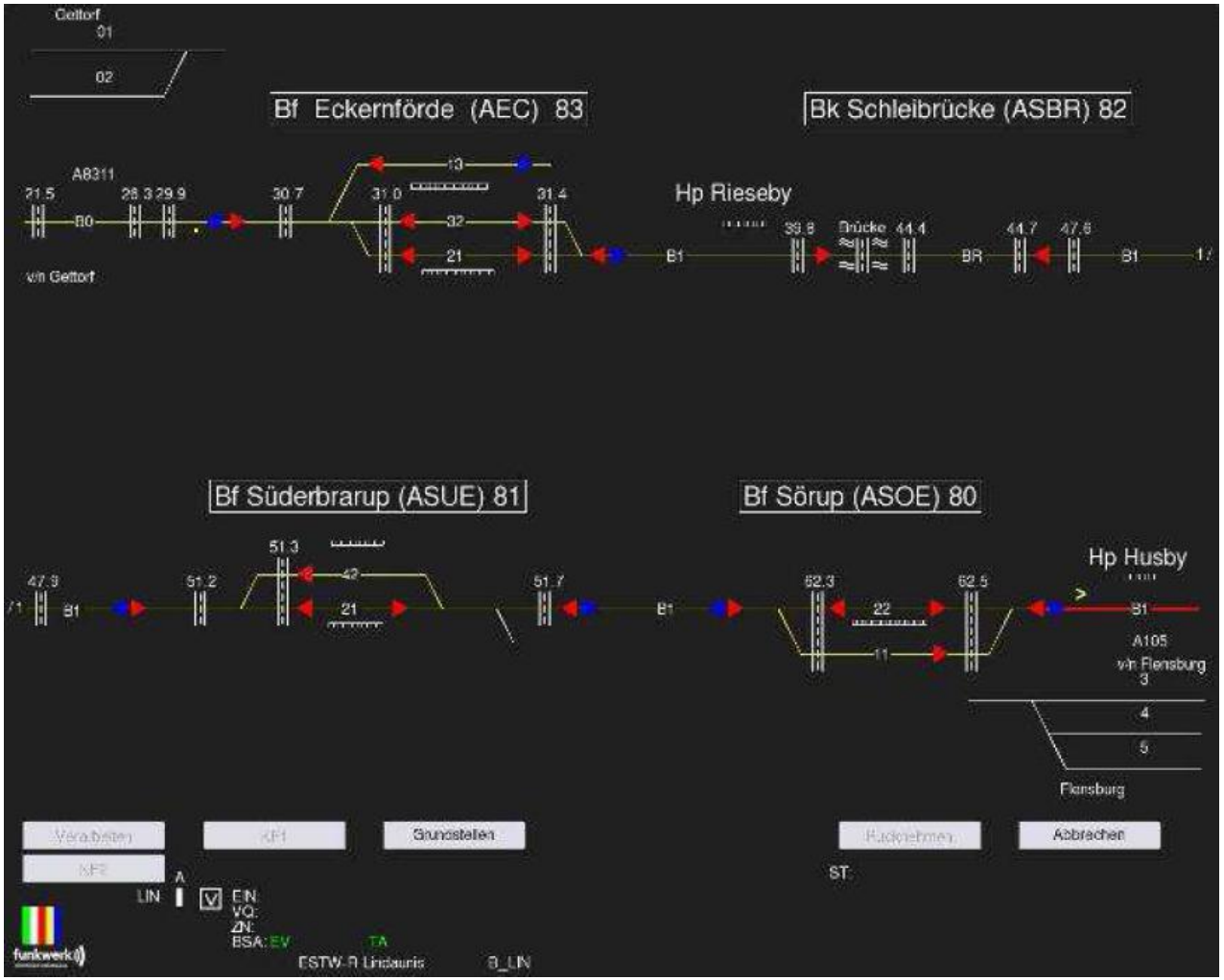


Bildquelle: Funkwerk IT GmbH

Wirtschaftlicher Bahnbetrieb

- Verlagerung von Personen- und Güterverkehr auf die Schiene, Zukunftssicherung bei gleichzeitiger Aufwands- und Kostenreduktion.
- Hoher Modernisierungsbedarf bei der Bahn, Pilotprojekt ESTW-R Lindaunis durch Funkwerk IT GmbH, Kompetenzbündelung im Kieler Raum.
- Modellgetriebener, generativer Ansatz als Chance für **Kostensenkungen** und **Effizienzsteigerungen** in der Signaltechnik.

Regionalstellwerk ESTW-R Lindaunis



Bildquelle: Funkwerk IT GmbH

MENGES: Zielplattform

Speicherprogrammierbare Steuerungen (SPS)

- SPS sind Standardkomponenten in der Industrieautomatisierung.
- SPS sind verschleißfrei, flexibel einsetzbar, relativ klein und kostengünstig, echtzeitfähig, zuverlässig, erlauben schnelle Fehleranalyse, geringerer Stromverbrauch.
- Hohe Sicherheit und Kostenreduktion in der Entwicklung und im Betrieb durch Einsatz von kompakten Standardlösungen mit langen Lebenszyklen.

Softwareentwicklung für SPS

- SPS bieten Funktionsblöcke einfacher Art, zyklische Verarbeitung.
- Als eingebettete Systeme mit begrenztem Speicher.
- Spezifische Beschränkungen in der Programmierung für einen zertifizierten Entwicklungsprozess.

Steuerungen im Feld

Bildquelle: Funkwerk IT GmbH

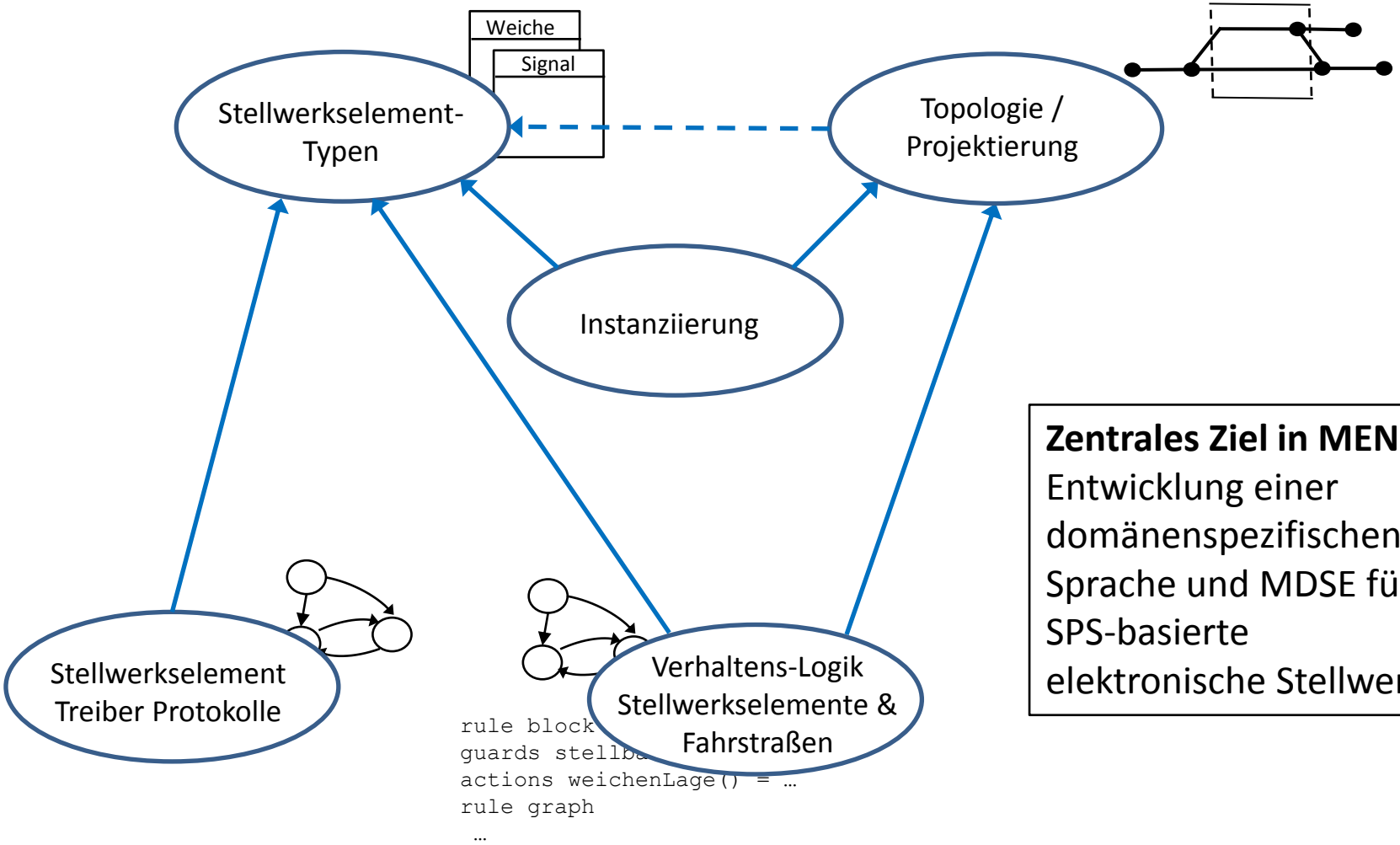
Steuerungen im Labor-Testfeld

Bildquelle: Funkwerk IT GmbH

Agenda

1. Die Domäne
2. Die domänenspezifische Sprache (DSL)
3. Modellgetriebene Instrumentierung und Analyse
4. Werkzeuginfrastruktur
5. Zusammenfassung und Ausblick

DSL-Partitionen

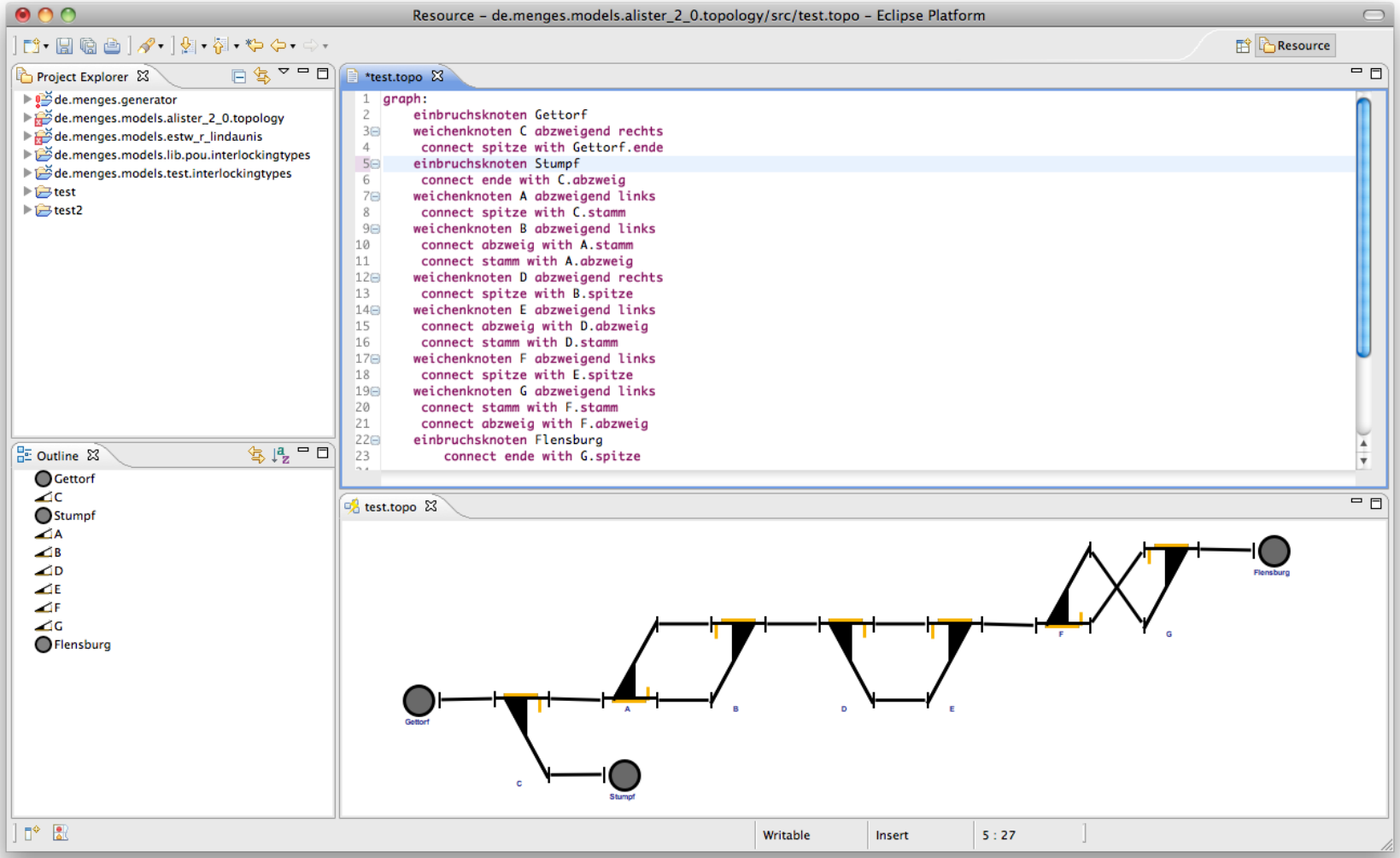


Zentrales Ziel in MENGES:
Entwicklung einer domänenspezifischen Sprache und MDSE für SPS-basierte elektronische Stellwerke

Stellwerkselement-Typen

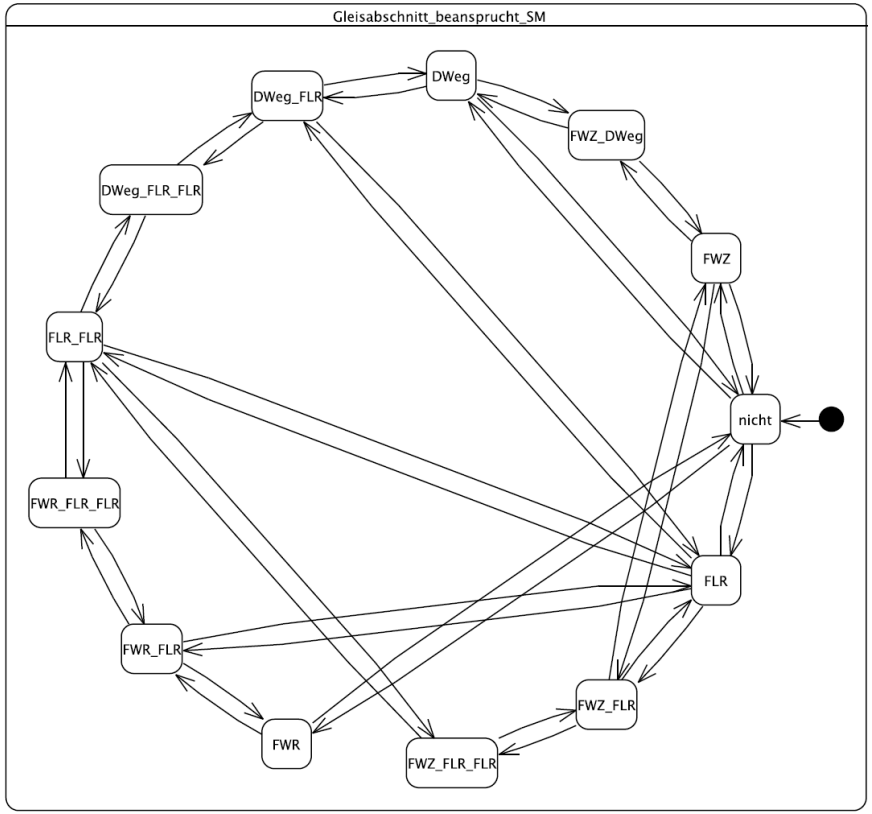
```
abstract interlocking element Beanspruchbar {

        statevars
                beansprucht: (
                   nicht,   // nicht beansprucht
                   DWeg,    // beansprucht als Durchrutschweg-Element
                   FLR,     // beansprucht im Flankenschutzraumn
                   FWR,     // beansprucht im Fahrweg einer Rangierstrasse
                   FWZ,     // beansprucht im Fahrweg einer Zugstrasse
                   DWeg_FLR, FLR_FLR, FWR_FLR, FWZ_FLR,
                   DWeg_FLR_FLR, FWR_FLR_FLR, FWZ_FLR_FLR,
                   FWZ_DWeg
                );

        attributes
                  befahrbarkeitsSperre          : boolean;
                  reserviert                    : boolean;
                  freiMeldung                   : boolean;
                  verschluss                    : boolean;
                  zustandsDatenVorhanden        : boolean;

        predicates
                  istBeanspruchtInFahrstrasse() =
                            ((beansprucht != nicht && beansprucht != FLR)
                             && beansprucht != FLR_FLR);
}
```

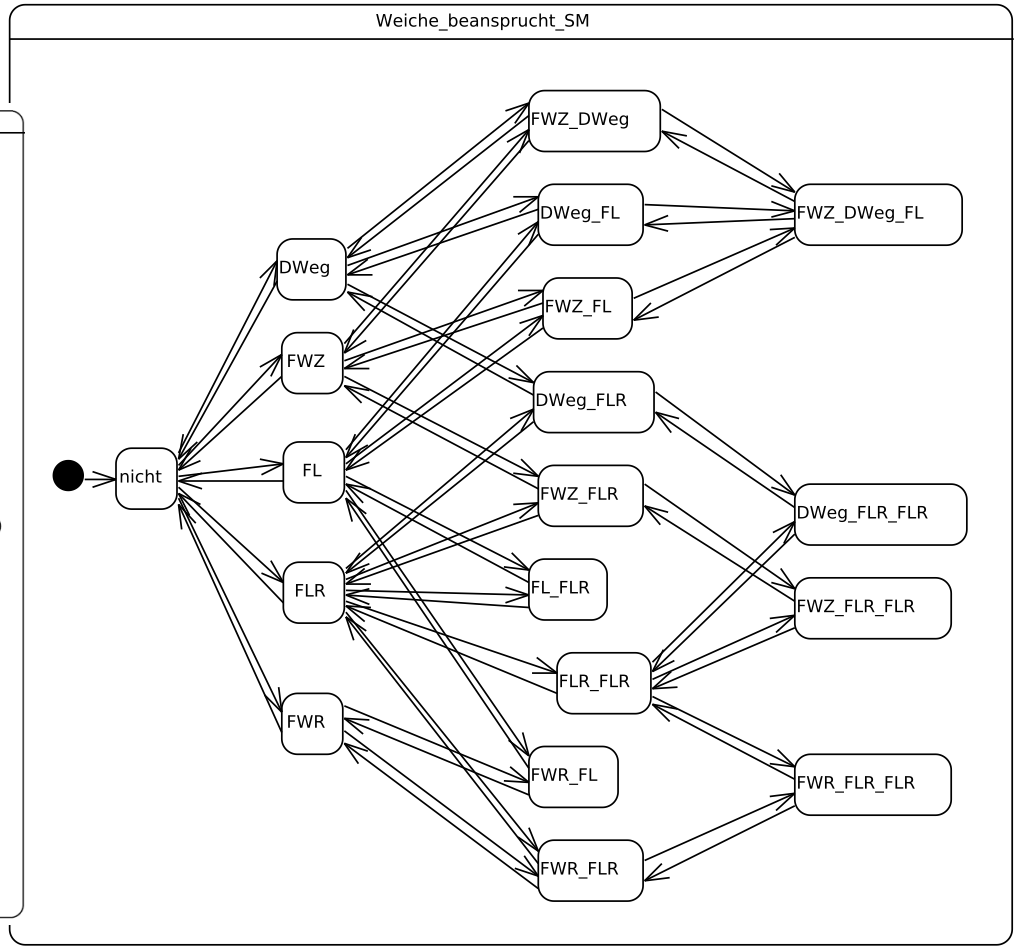
Topologie / Projektierung



Verhaltensspezifikation von Stellwerkselementen

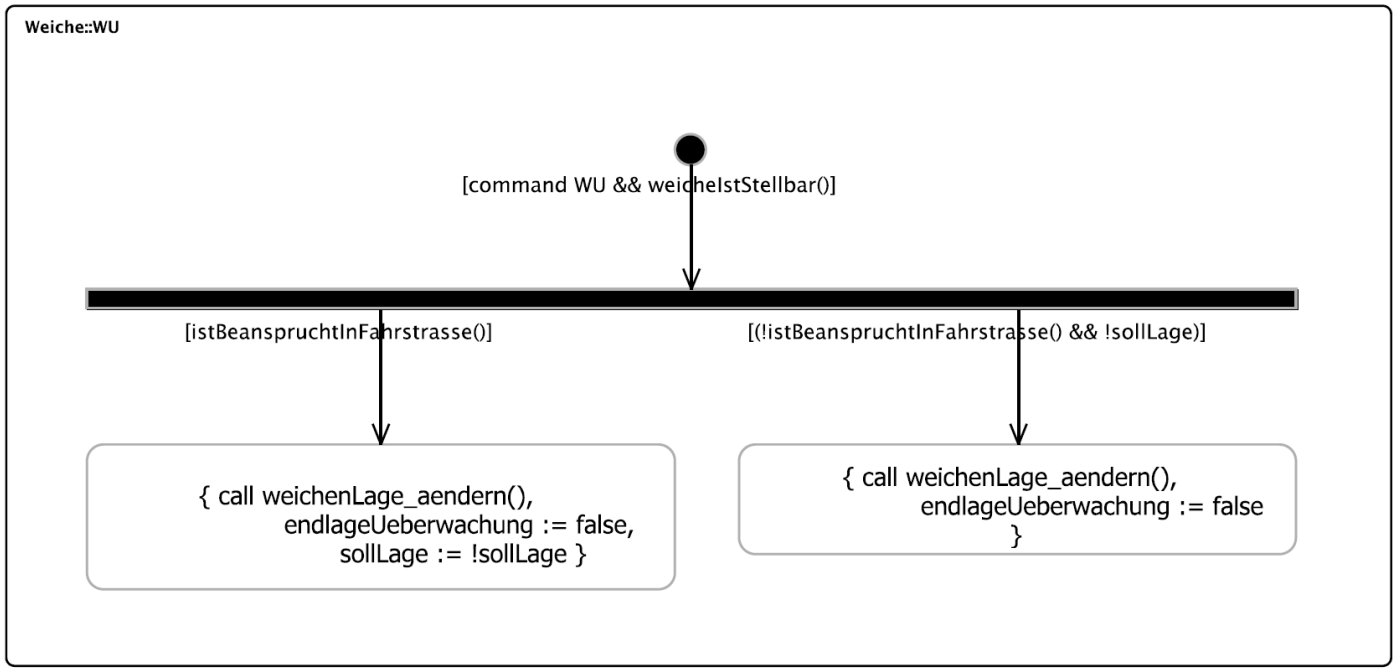
```
state machine Beansprucht_SM {
        references beansprucht in Beanspruchbar;
        transitions
              nicht            --> DWeg
              DWeg             --> nich
```





Verhaltensspezifikation zum Umlegen einer Weiche

```
rule block Bedienkommando_WU references Weiche {

    local predicates
    local procedures

    rule graph

      --> [(command WU && weicheIstStellbar())] {
          -> [istBeanspruchtInFahrstrasse()] /


          -> [(!

              }
    }
```



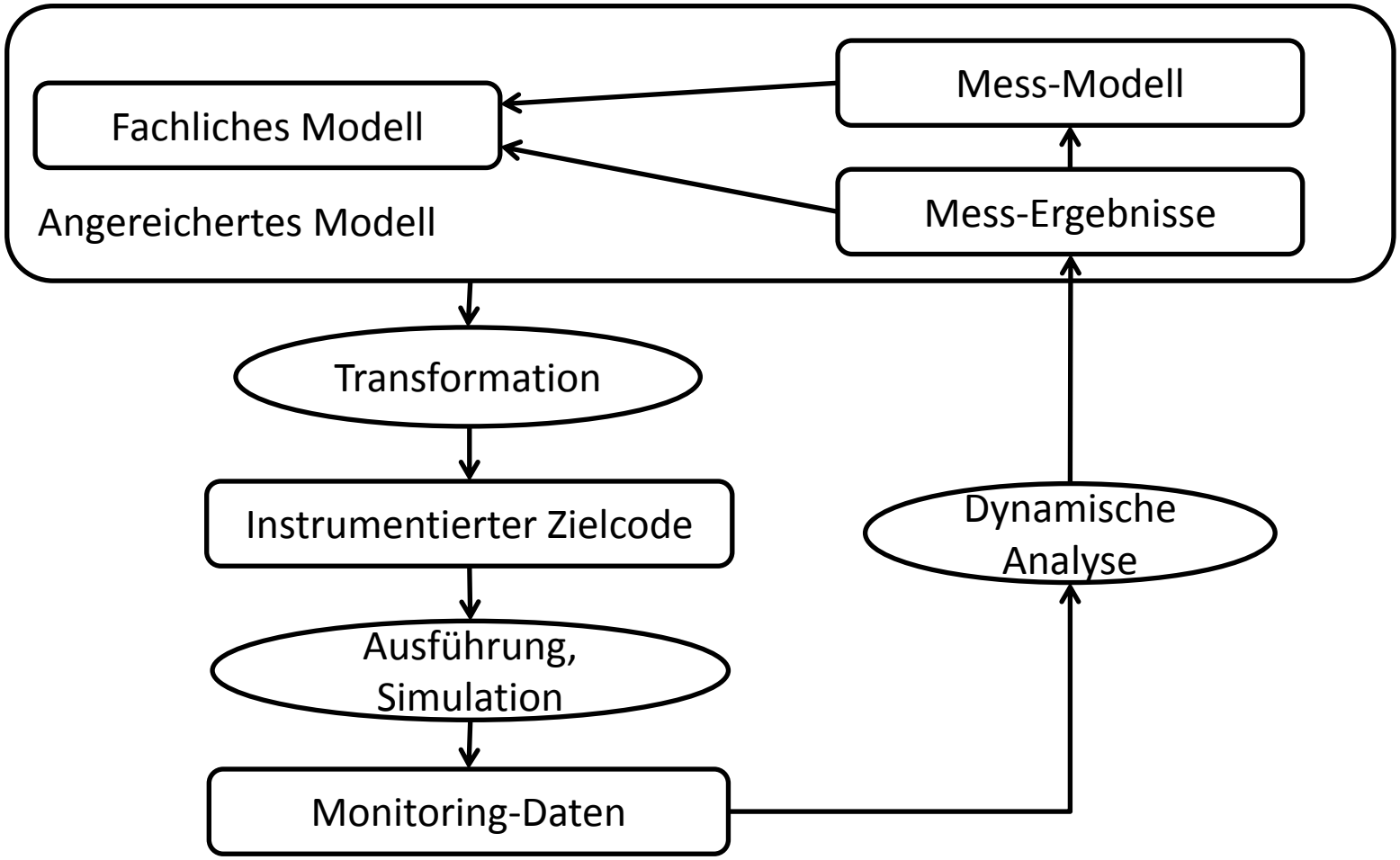
Verhaltensspezifikation zum Aufbau von Fahrstraßen

```
rule block bedKomZSE references Fahrstrasse () {
  immediate protocol (Status)
    zulassen: if [true] then {
      foreach (element in fahrweg) {
        call element.reserviere(this);
      };
      quittung -> abgelehnt;
    };
    reservieren: if [all (element in fahrweg) {
      element.istReserviertFuer(this)
    }] then {
      quittung -> ok;
    };
    abbruch: if [quittung == abgelehnt];
}
```

Agenda

1. Die Domäne
2. Die domänenspezifische Sprache (DSL)
3. <u>Modellgetriebene Instrumentierung und Analyse</u>
4. Werkzeuginfrastruktur
5. Zusammenfassung und Ausblick

Modellgetriebene Instrumentierung und Analyse



Ein ähnlicher Ansatz wird auch genutzt in https://kosse-sh.de/dynamod

(siehe Vortrag heute morgen beim AK MDA)

DynaMod
Dynamic Analysis for Model-Driven Modernization

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Modellgetriebene Instrumentierung und Analyse

```
…
Kieker_Log_Type(type:=EXECUTION_TIME_ENTER,
          value:=pThis)
Kieker_Log_Bool(value:=pThis.reserviert)
Kieker_Log_Beanspruchung(value:=pThis.beanspruchung)
Kieker_Log_Flush()

IF pThis.superElement.reserviert THEN
          IF GA_Weiche_HatSolllage
                    (pThis:=pThis) AND
             GA_Weiche_NichtBeansprucht
                    (pThis:=pThis) THEN
                    pThis.beanspruchung := WEICHE_FW_Z;
          ELSIF     NOT GA_Weiche_HatSolllage
                    (pThis:=pThis) AND
             GA_Weiche_IstStellbar
                    (pThis:=pThis) AND
             GA_Weiche_NichtBeansprucht
                    (pThis:=pThis) THEN
                    pThis.beanspruchung := WEICHE_FW_Z;
          END_IF
END_IF

Kieker_Log_Type(type:=EXECUTION_TIME_EXIT,
          Value:=pThis)
Kieker_Log_Bool(value:=pThis.reserviert)
Kieker_Log_Beanspruchung(value:=pThis.beanspruchung)
Kieker_Log_Flush()

…
```

```
; Weiche_ZPZ_Bedingungen_beanspruchen_FW-Z
anspruche references GaWeiche () {
  rule graph
  --> [reserviert] {
            --> [hatSolllage() && nichtBeansprucht()] {
                        action beanspruchung -> fw_z;
            }
            --> [(!hatSolllage() && (istStellbar() &&
                        nichtBeansprucht() ))] {
                        action beanspruchung -> fw_z;
            }
  }
```

```
abstract interlocking element Besetztelement {
          properties
                    boolean reserviert;
}
```

```
interlocking element GaWeiche
extends Besetztelement
implements Bedienbar, Beanspruchbar
{
          statevars
                    beanspruchung: (
                              nicht,
                              fw_z
                    );
}
```

ne = 'reserviert'

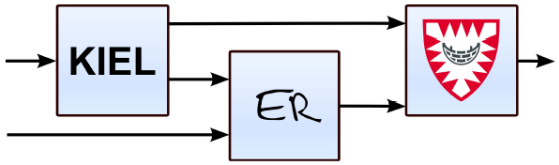
e = 'beanspruchung'

pruche'

Beispielfrage: Wie verhält sich die Ausführungszeit zu der Eigenschaft `reserviert` und dem Zustand von `beanspruchung`?

Werkzeuginfrastruktur

- Integrierte Entwicklungsumgebung
- Entwicklung textueller DSLs
- Code Generierung
- Pragmatik, grafische Sichten
- SPS-Simulation
- Profiling und Analyse
- Projektmanagement (TinyPM, Trac, SVN, …)



http://kieker.sourceforge.net/

Zusammenfassung

- Verbesserung der Kommunikation mit den Domänenexperten
  - Durch Verwendung domänenspezifischer Sprachelemente
- Erkennung und Abstraktion von Entwicklungs-mustern mittels neuer Darstellungs- und Eingabeformen
  - Paralleles Arbeiten in Text und Grafik
- Modelltransformation für durchgängige und effiziente Entwicklung
  - Codegenerierung mit hohem Automatisierungsgrad

Ausblick

- Projektlaufzeit von Januar 2010 bis Dezember 2012
  - Halbzeit
- Zurzeit intensive Evaluation der DSLs auf Basis einer ersten Durchstich-Implementierung.
  - U.A. mittels GQM
- Weiterhin:
  - Schaffung von Schnittstellen zur Projektierung (automatische Integration von Beständen / Planungen aus Planungssystemen)
  - Automatische Generierung von Testfällen auf Basis der MENGES-Spezifikation (inkl. Fault Injection)
  - Instrumentierung / Profiling und Rückfluss der Ergebnisse in die Spezifikation und Modellierung