

Entwurf einer domänenspezifischen Sprache für elektronische Stellwerke

Wolfgang Goerigk¹, Reinhard von Hanxleden⁴, **Wilhelm Hasselbring**³, Gregor Hennings¹,
Reiner Jung³, Holger Neustock², Heiko Schaefer², Christian Schneider⁴,
Elferik Schultz², Thomas Stahl¹, Steffen Weik¹ und Stefan Zeug¹

¹ b+m Informatik AG, 24109 Melsdorf

² Funkwerk Information Technologies GmbH, 24145 Kiel

³ Lehrstuhl Software Engineering, Universität Kiel

⁴ Lehrstuhl Echtzeitsysteme und Eingebettete Systeme, Universität Kiel

Kompetenzverbund Software Systems Engineering (KoSSE)

<http://kosse-sh.de/>



Der Kontext:

KoSSE-Projekt MENGES

M	odellbasierte	Das Team, welches die Inhalte dieses Vortrages erarbeitet hat:
E	ntwurfsmethoden für eine	AG Software Engineering, Universität Kiel Wilhelm Hasselbring, Reiner Jung
N	eue	AG Echtzeitsysteme und Eingebettete Systeme, Universität Kiel Reinhard von Hanxleden, Christian Schneider
G	eneration	Funkwerk Information Technologies GmbH, Kiel Holger Neustock, Heiko Schaefer, Elferik Schultz,
E	lektronischer	Hauke Fuhrmann, Uwe Klose b+m Informatik AG, Melsdorf
S	tellwerke	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
2/34

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

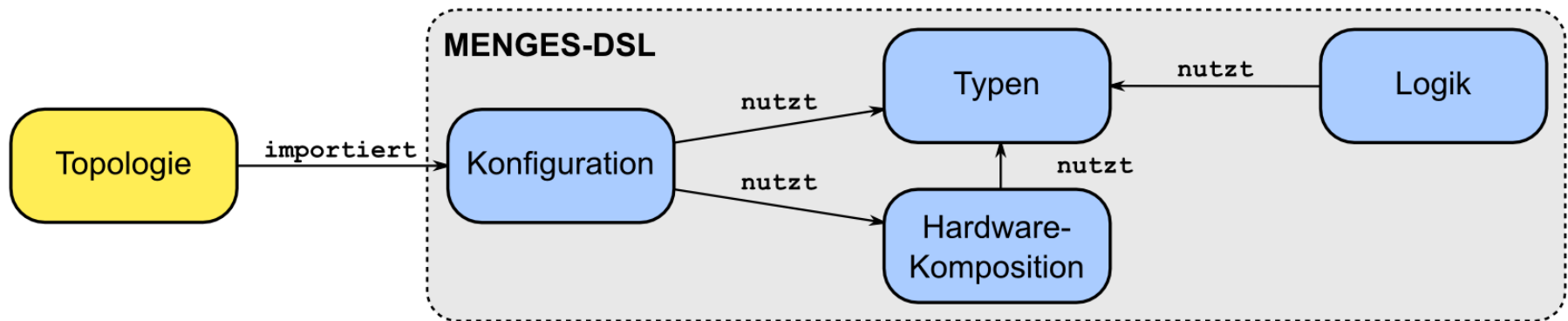


Bildquelle: Funkwerk IT GmbH

Agenda

1. Die Domäne
2. Die domänenspezifische Sprache (DSL)
3. Simulation und Profiling
4. Zusammenfassung und Ausblick

Teilsprachen der DSL



Typ	Komponenten-Typen, Konnektor-Typen, ...
Logik	Entscheidungsbäume, Prozesse, Automaten, Regeln
Konfiguration	Komponenten , Konnektoren, Anschlussbelegung
Hardware/Geräte	Komposition der SPS und DI/Os, Deployment und Instanzen
Topologie	Weichen, Signale, Schienen, Fahrwege etc. und deren Beziehungen

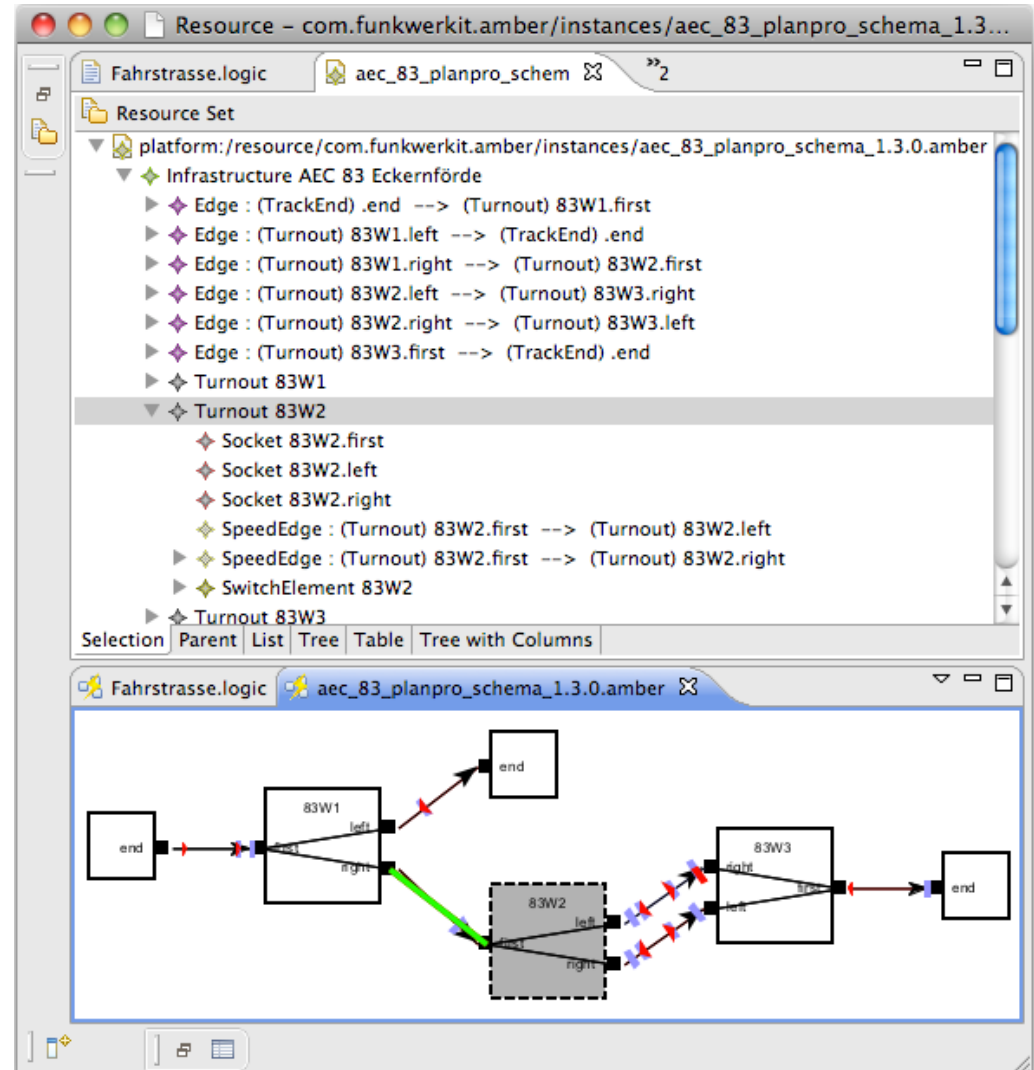
Topologie

Visualisierung für AMBER

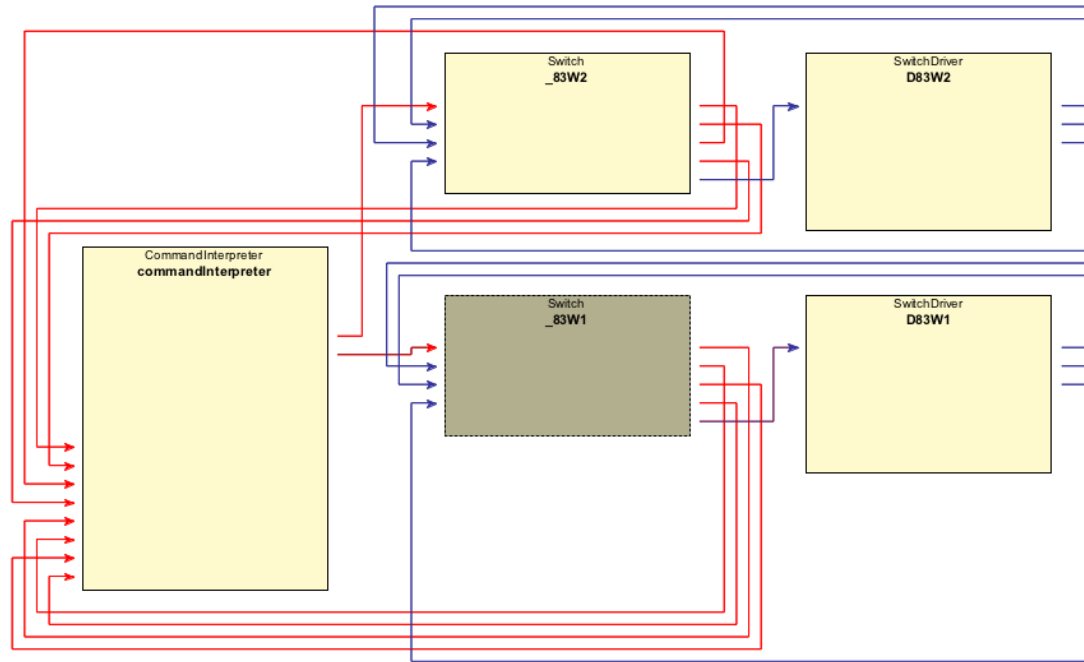
- EMF Meta-Modell
- XMI Serialisierung

Editor

- basiert auf EMF-Tree
- Interaktive Visualisierung



Konfigurationen: Ein Lokalstellwerk



Komponenten

- Spezifikation durch Prototypen
- Kapazitäten in der Hardwarebeschreibung

Konnektoren

- Spezifikation durch Komponenten-Tupel
- Keine explizite Angabe von Kapazitäten

Komponenten-Typen

```
type Switch {  
  
    properties  
        boolean turning;  
  
    communication roles  
        SwitchCommands.Switch command;  
        DeviceControl.Logic driver;  
  
    conditionals  
        controlCommands();  
  
    state machines  
        ProtocolStates protocolSwitch();  
  
}
```

Konnektoren-Typen

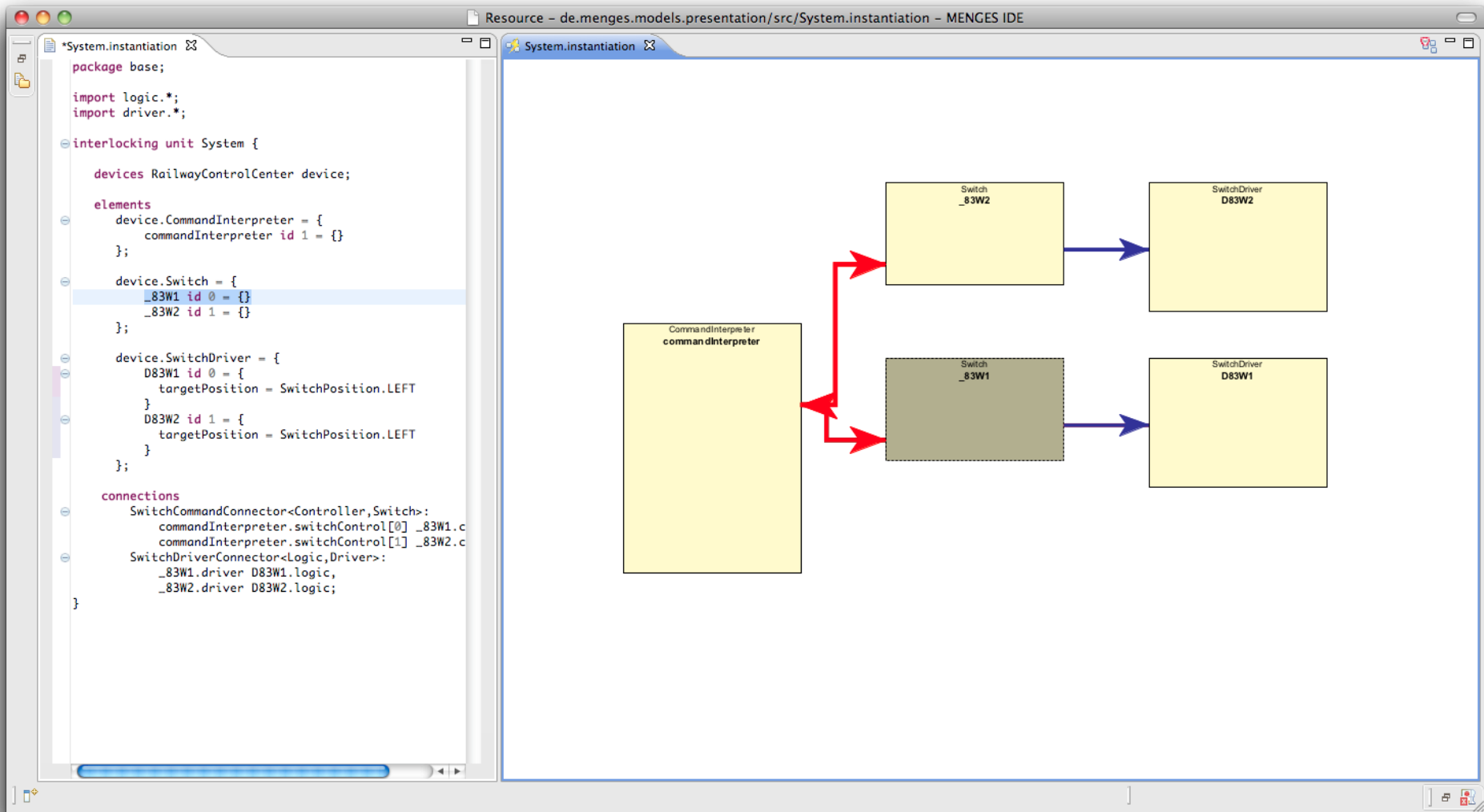
Kommunikation mit Nachrichten

Asynchrone, zustandsbehaftete Kommunikation zwischen zwei Partnern

```
connector SwitchCommands: Controller, Switch {  
  messages  
    turn();  
    accept();  
    success();  
    error();  
    denied();  
  
  rules  
    Controller sends turn ->  
      Switch sends ((accept, (success|error)) | denied) ->  
        Controller;  
}
```

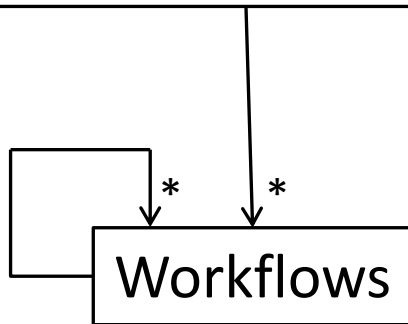
Weiterhin wird die zustandslose Kommunikation von Werten unterstützt.

Pragmatik



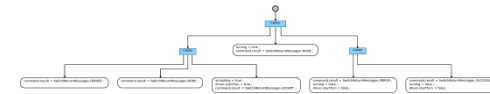
Verhaltensbeschreibung

Entscheidungsbäume



Protokollautomaten

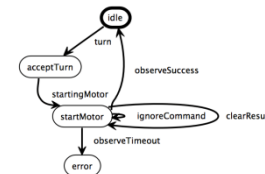
Beispiel: Eine Weiche soll umgelegt werden



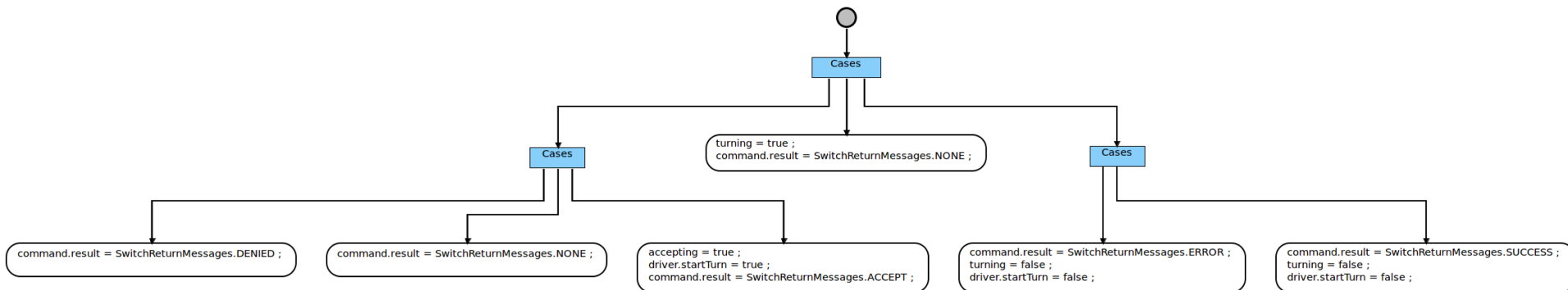
Beispiel: Weiche umlegen



Beispiel: Protokoll einer Weiche



Entscheidungsbaum: Eine Weiche soll umgelegt werden



```
,
case accepting && !command.turn : {
    turning = true;
    command.result = SwitchReturnMessages.NONE;
}
case turning: {
    case !driver.motorRunning && driver.timeout: {
        command.result = SwitchReturnMessages.ERROR;
        turning = false;
        driver.startTurn = false;
    }
}
```

Workflow: Weiche umlegen

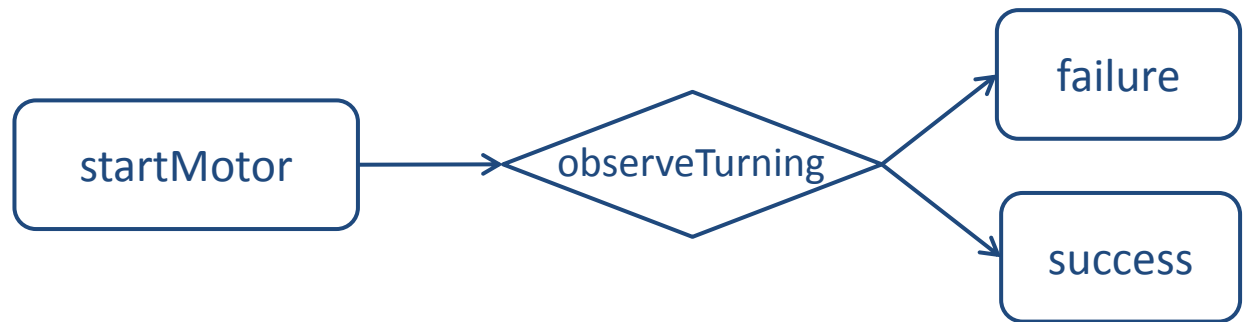
```
workflow turnSwitch  
  start startMotor;
```

```
  action startMotor  
    driver.start  
    send command  
  }
```

```
  action failure {
```

```
  action success {
```

```
  decision observeTurning {  
    case !driver.motorRunning && driver.timeout : failure();  
    case !driver.motorRunning && [...]: success();  
  }  
}
```



Protokollautomat für eine Weiche

```
state machine protocolSwitch{  
    start state idle;
```

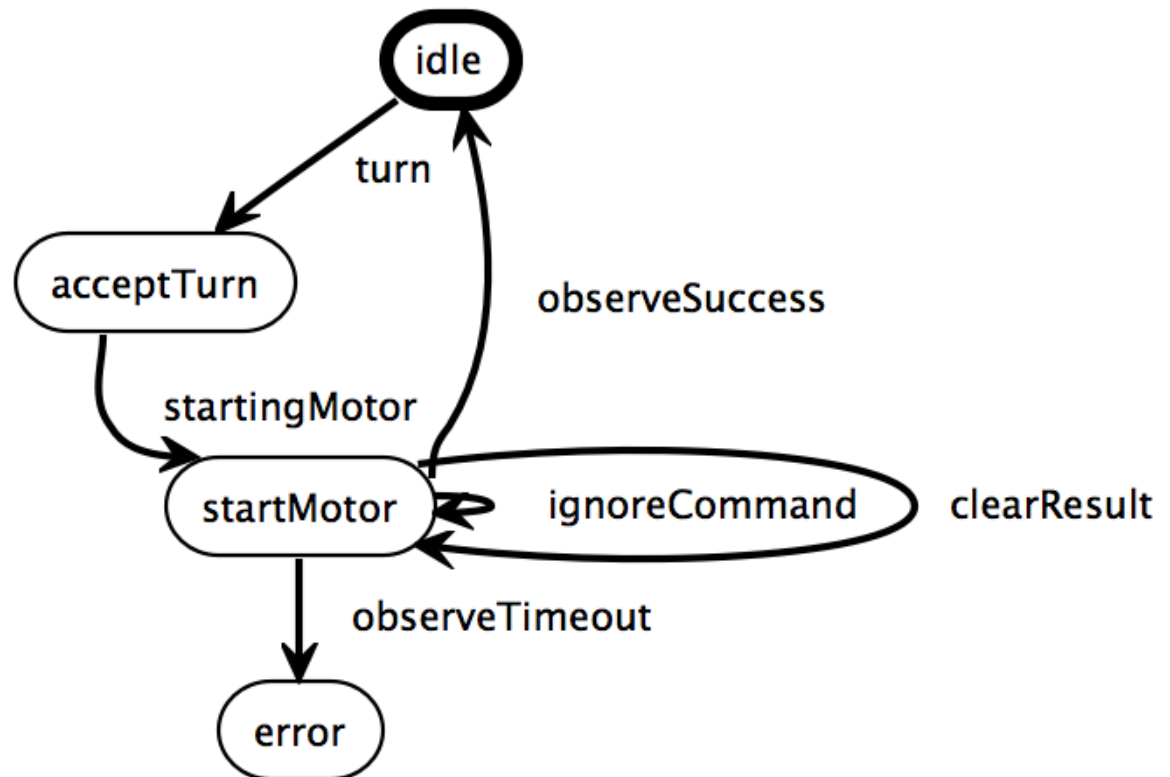
```
    from idle transition turn  
        if ( comm
```

```
    from acceptTurn transition star  
        if ( !comn  
    from startMotor transition obse  
        if (!driver
```

```
    from startMotor transition igno  
        if ( comm
```

```
    from startMotor transition clea  
        if ( !comn
```

```
    from startMotor transition obse  
        if (!driver
```



```
        driver.startTurn = false;  
        command.result = SwitchReturnMessages.SUCCESS;
```

```
    }
```

Hardwarekomposition

Spezifikation von Gerätetypen

- I/O-Beschreibung
- Anschlussschema

Geräte

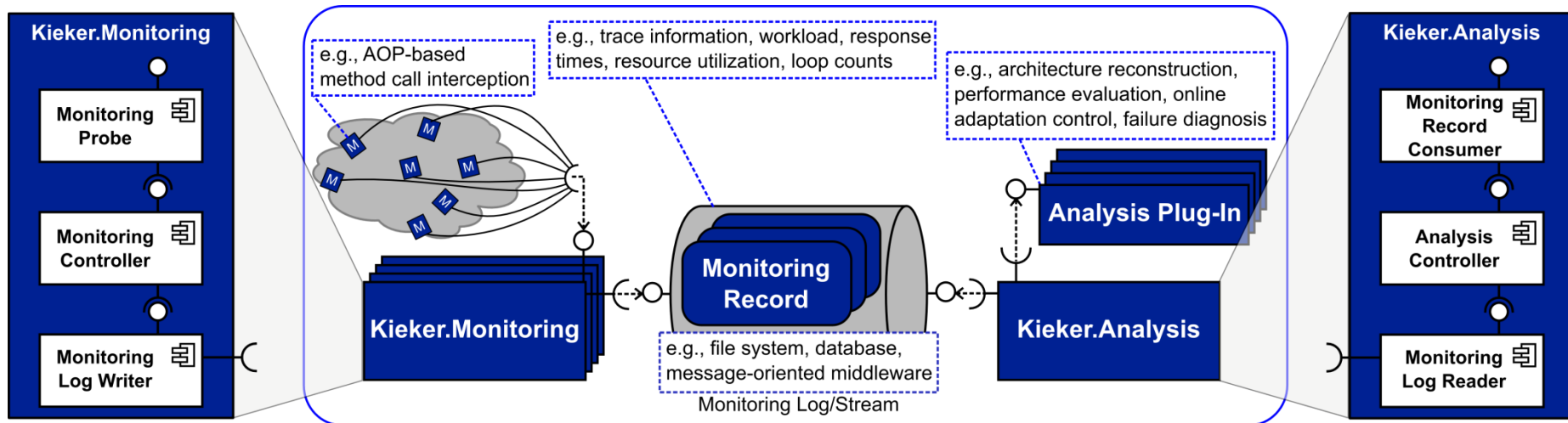
- Instanziierung der Softwaretypen
- Instanziierung der Gerätetypen
- Kopplung der Geräte-Instanzen

Agenda

1. Die Domäne
2. Die domänenspezifische Sprache (DSL)
3. Simulation und Profiling
4. Zusammenfassung und Ausblick

Monitoring und Profiling mit **Kieker**

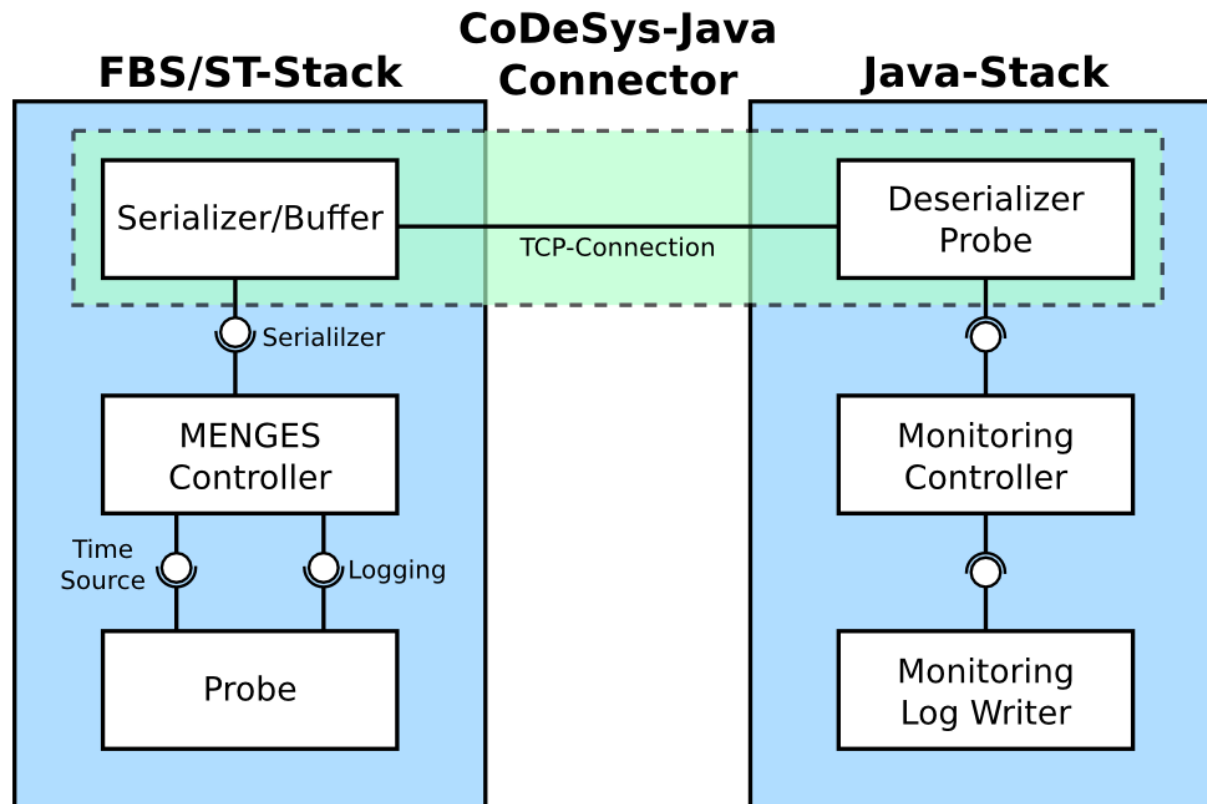
Das Kieker-Monitoring-Framework, <http://kieker-monitoring.net/>



- Java-basiert
- Anbindung an C#, Visual Basic, COBOL
- Web-basiertes Analyse-Framework mit Plugin-Architektur (ab Version 1.5)
- Peer-reviewed tool der SPEC Research Group:
<http://research.spec.org/>



Simulation und Profiling von SPS



Werkzeuginfrastruktur

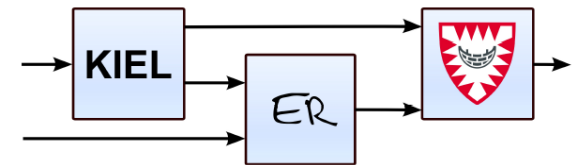
- Integrierte Entwicklungsumgebung
- Entwicklung textueller DSLs
- Code Generierung
- Pragmatik, grafische Sichten
- SPS-Simulation
- Profiling und Analyse
- Projektmanagement (Trac, SVN, Hudson...)



Xtext



Xtend



Kieker

<http://kieker-monitoring.net/>

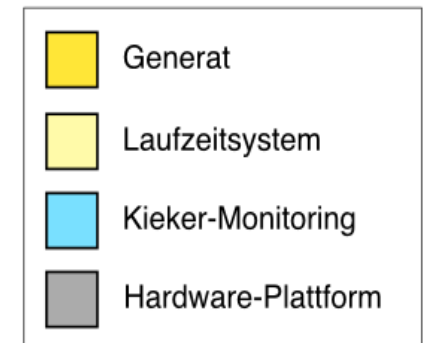
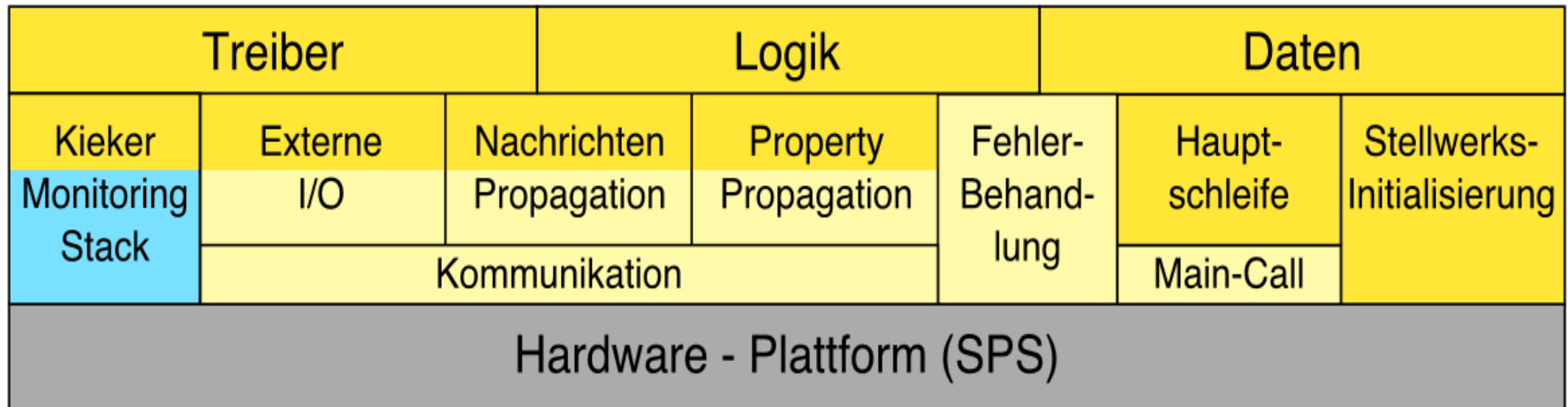
Zusammenfassung

- Domänenspezifische Sprache für eingebettete Systeme mit Steuerungsaufgaben
- Teilsprachen für bestimmte Sichten und Aufgaben
- Schnittstellen zur Projektierung
 - Integration von Planungssystemen via AMBER
- Textuelle DSL
 - Pragmatik: Automatisch generierte grafische Sichten
- Profiling-Ansatz mit Kieker

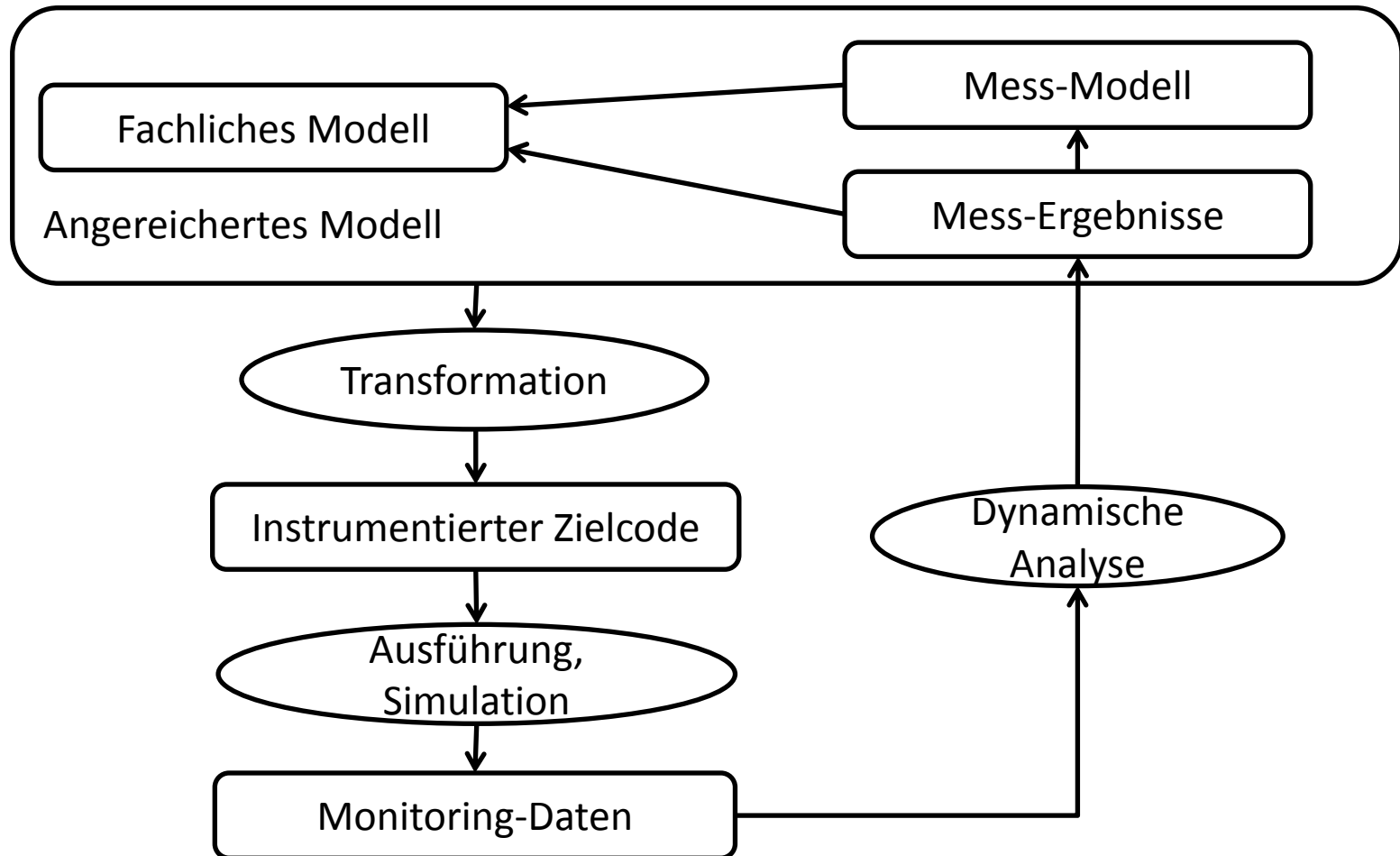
Ausblick

- Projektlaufzeit von Januar 2010 bis Dezember 2012
- Evaluation der DSL und der Codegenerierung auf Basis von Fallstudien bei Funkwerk IT.
 - Umsetzung der Evaluationsergebnisse:
Die DSL wird noch intensiv optimiert und weiterentwickelt.
- Automatische Generierung von Testfällen auf Basis der MENGES-Spezifikation und daraus abgeleiteten Funktionsbäumen
 - Siehe: Wasilewski & Hasselbring: A formal and pragmatic approach to engineering safety-critical rail vehicle control software. SE 2011.
- Instrumentierung / Profiling und Rückfluss der Ergebnisse in die Spezifikation und Modellierung
 - Modell-getriebene Instrumentierung und Analyse
- Anbindung von Model Checkern
 - Nicht in der jetzigen Projektlaufzeit vorgesehen, jedoch von vornherein avisiert

Schichten-Architektur eines Stellwerkskerns



Modellgetriebene Instrumentierung und Analyse



Ein ähnlicher Ansatz wird auch genutzt in
<http://kosse-sh.de/dynamod>