

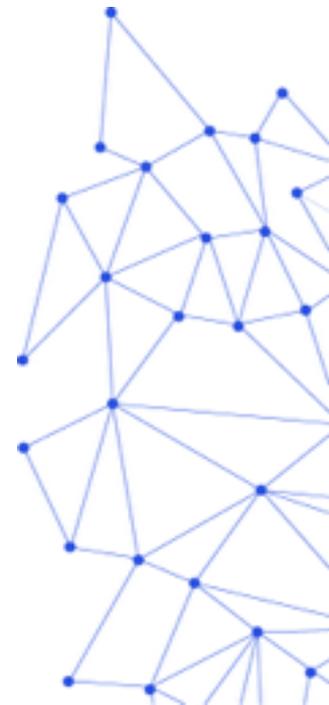
March 2015

View Integration in Model-Based Development

McGill NECSIS Workshop 2015

Bernhard Schätz, Sebastian Voss and Vincent Aravantinos

fortiss GmbH
An-Institut Technische Universität München



Model-Based Engineering of Embedded Systems

What is the benefit of a model-based design of embedded software systems in the car industry?

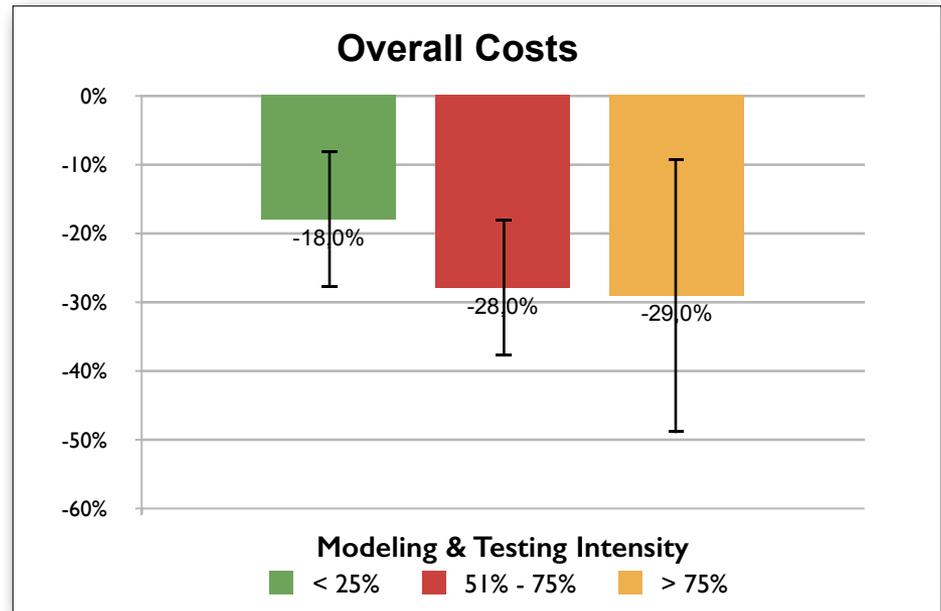
Manfred Broy
Technical University Munich, Germany
Sascha Kirstan
Altran Technologies, Germany
Helmut Krcmar
Technical University Munich, Germany
Bernhard Schätz
fortiss, Germany
Jens Zimmermann
Altran Technologies, Germany

ABSTRACT

Model-based development becomes more and more popular in the development of embedded software systems in the car industry. On the websites of tool vendors many success stories can be found, which report of efficiency gains from up to 50% in the development, high error reductions and a more rapid increase of the maturity level of developed functions (The Mathworks, 2010) (dSPACE 2010) just because of model-based development. Reliable and broadly spread research that analyze the status quo of model-based development and its effects on the economics are still missing. This article describes the results of a global study by Altran Technologies, the chair of software and systems engineering and the chair of Information Management of the University of Technology in Munich which examines the costs and benefits of model-based development of embedded systems in the car industry.

1. INTRODUCTION

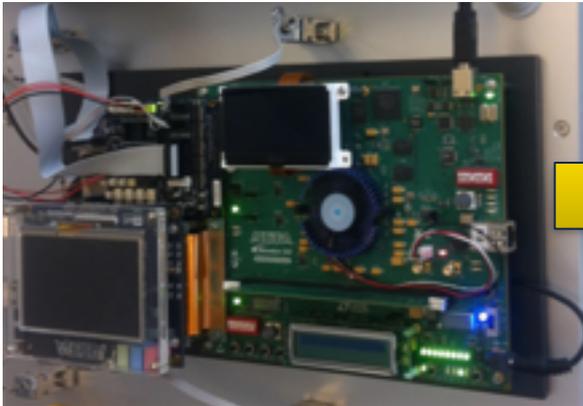
In the last 20 years the value chain in the car industry has changed drastically. All car producers and suppliers worldwide have worked on improvements in the area of mechanics, the improvement of quality requirements, and improvements in the logistic area. A lot of the potential in these areas is already exploited. A main differentiation factor turns out to be the electronics area, where a change from hardware to software development is carried out. The meaning electronics will have in the next years has been analyzed by a study of Mercer Management Consulting (Mercer, 2004). The study focuses mainly on the question how the cost factors in the development of a car will change until the year 2015 in comparison to the year 2002. In 2015 the costs for the development of electronics will have a value of 35% of the total car production costs. Whereas areas as power train and body have small increases, the costs for the development of electronic systems will be almost tripled. The predicted increases result from a variety of innovations which are being expected in this area. The majority of innovations are realized with embedded systems and especially with software. "90 percent of the future innovations in the car will be based on electronics and from that 80 percent will be realized by software" (Lederer, 2002). However, today's software development has big challenges to master like shortened development times for the cars in total versus longer development times for the software, high safety requirements and especially the growing complexity because of the rising number of functions and the increasing interaction between the functions. To master these challenges car producers and suppliers conduct a paradigm change in the software development from hand-coded to model-based development.



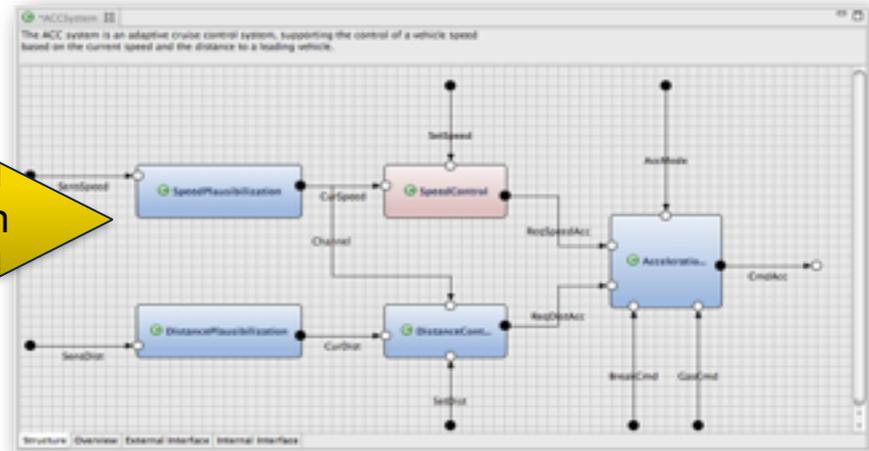
Increase efficiency of development by division of complexity

Basic Principles of Model-Based Engineering

1. Abstraction



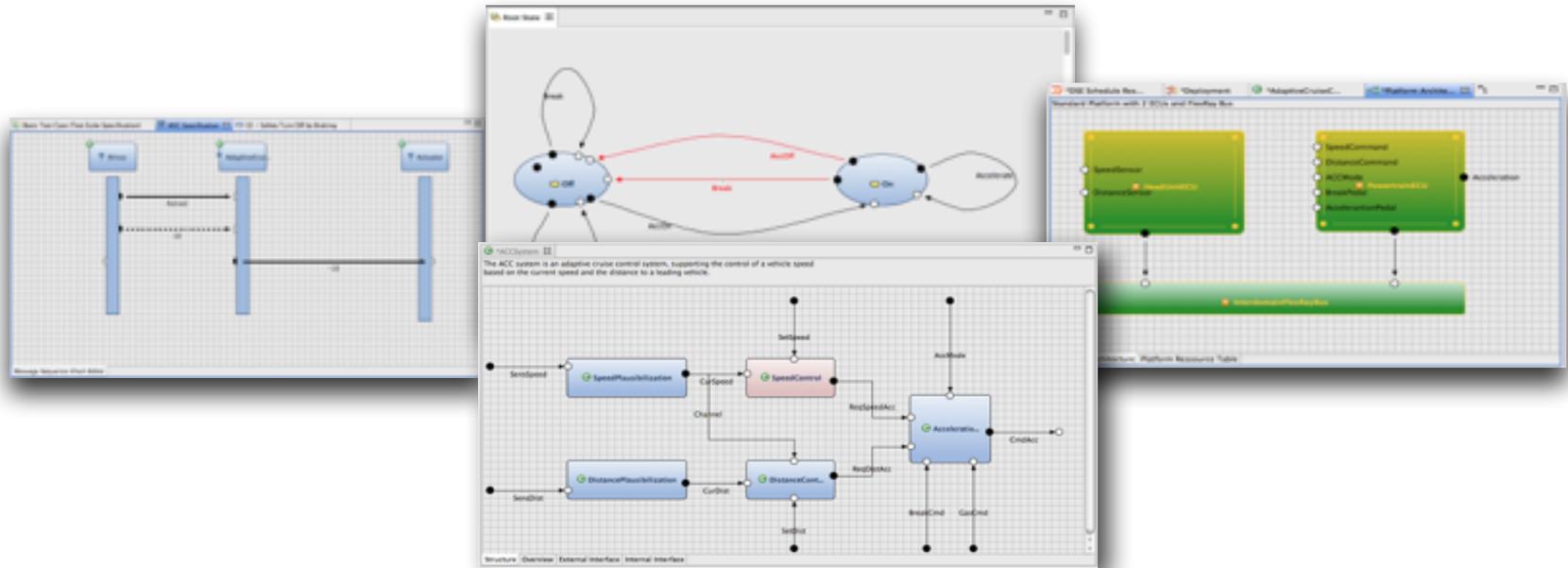
Abstraction



Idealized modeling assumptions eliminate accidental complexity of implementation level

Basic Principles of Model-Based Engineering

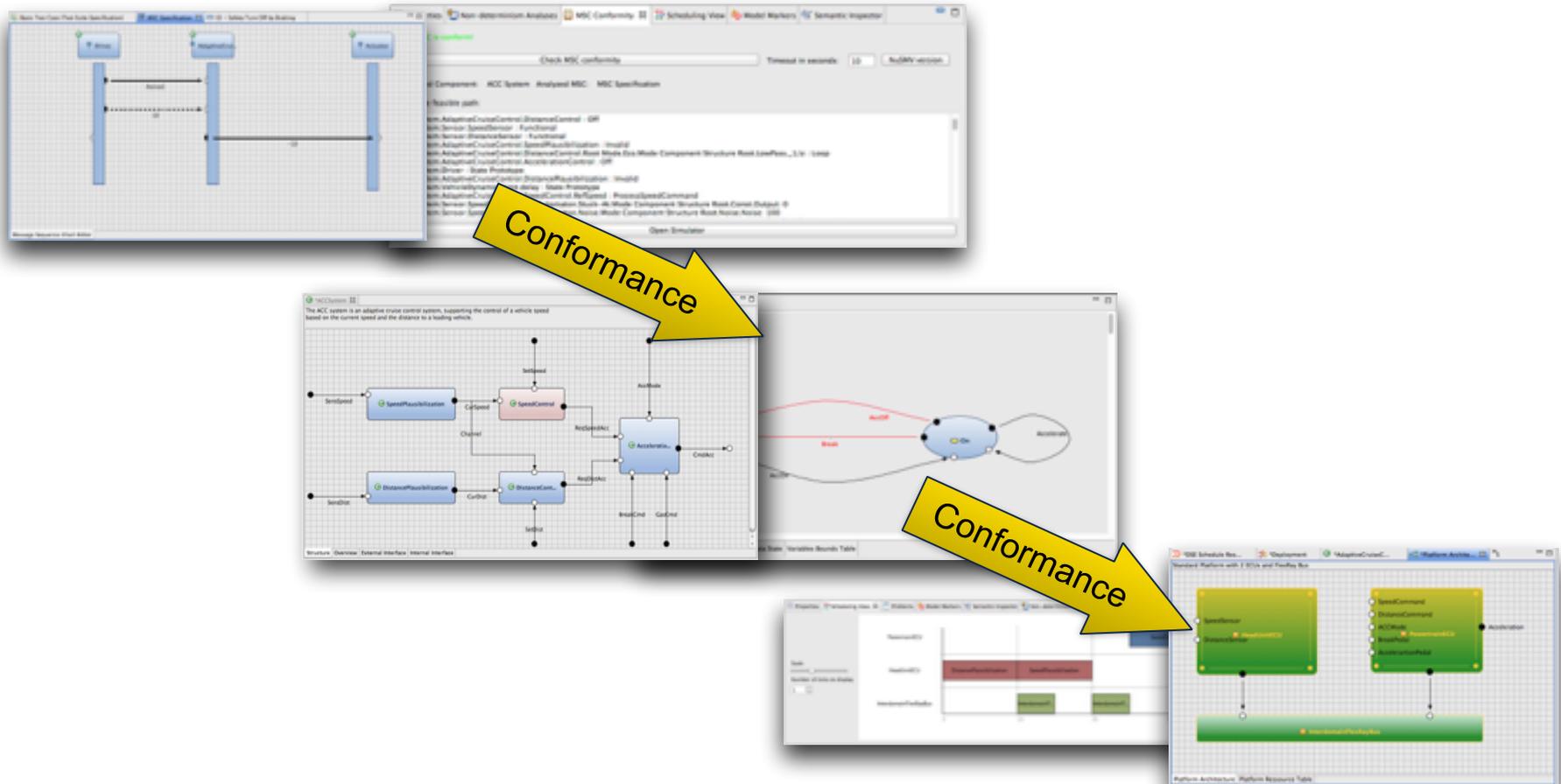
2. Views



Modular projections of system reduce essential complexity of problem

Basic Principles of Model-Based Engineering

3. Analysis/Synthesis



Automated steps ensure conformance across abstractions and views

Example: MBSE with AF3

AF3-Demo — see af3.fortiss.org

The screenshot shows the AF3 website interface. At the top, there is a navigation menu with links for HOME, MAIN FEATURES, DOWNLOAD, DOCS, USE CASES, RESEARCH, and ABOUT US. The AF3 logo is in the top right corner. The main heading is "AF3" followed by the tagline "Seamless Model-based Development.. From requirements analysis to platform". Below this is a flowchart illustrating the development process: it starts with documents representing requirements, moves through modeling and code generation, then to hardware architecture (represented by a car chassis), and finally to a complete car model. Below the flowchart, a paragraph describes AF3 as a model-based development tool for distributed, reactive, embedded software systems. A news section highlights two events: "20 FEB AF3 VERSION 2.7 HAS BEEN RELEASED!" and "20 NOV AF3 NEW WEBSITE!". The footer contains a detailed navigation menu, contact information for InPressum Datenschutz AG, and copyright information for Fortiss GmbH.

af3.fortiss.org

HOME MAIN FEATURES DOWNLOAD DOCS USE CASES RESEARCH ABOUT US

AF3

Seamless Model-based Development..
From requirements analysis to platform

AF3 IS A MODEL BASED DEVELOPMENT TOOL FOR DISTRIBUTED, REACTIVE, EMBEDDED SOFTWARE SYSTEMS

AF3 is a powerful tool to develop embedded systems using models from the requirements to the hardware architecture, passing by the design of the logical architecture, the deployment and the scheduling. AF3 provides advanced features to support the user ensuring the quality of his/her system: format analyses, synthesis methods, space exploration visualization...

20 FEB AF3 VERSION 2.7 HAS BEEN RELEASED!
AF3 version 2.7 has been released! We are proud to announce the release of AF3 version 2.7! This...

20 NOV AF3 NEW WEBSITE!
We are proud to launch the new website for AutoFocus! Contact us if you have some feedback.

HOME MAIN FEATURES DOWNLOAD DOCS USE CASES RESEARCH ABOUT US

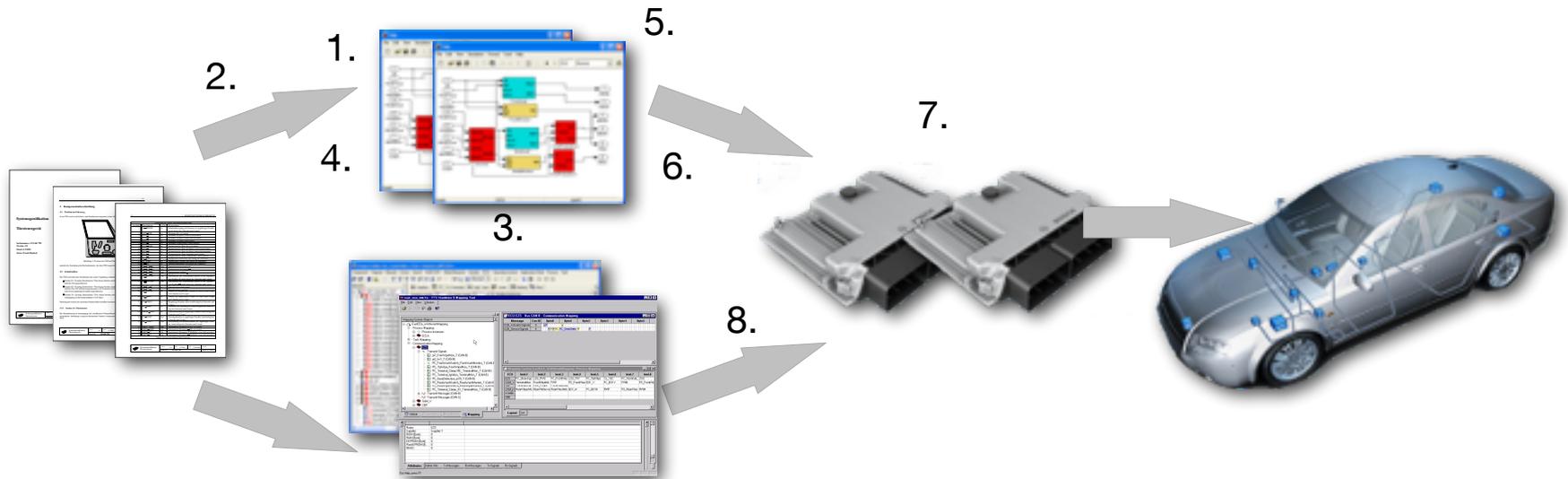
REQUIREMENTS MODELING CODE GENERATION FORMAL VERIFICATION DESIGN SPACE EXPLORATION TESTING

SCREENCASTS USE CASES SELECTED PROJECTS DEMONSTRATORS RESEARCH ABOUT US TEAM NEWS

IMPRESSUM DATENSCHUTZ AG

© 2011 FORTISS GMBH - AN INSTITUT TECHNISCHE UNIVERSITÄT MÜNCHEN

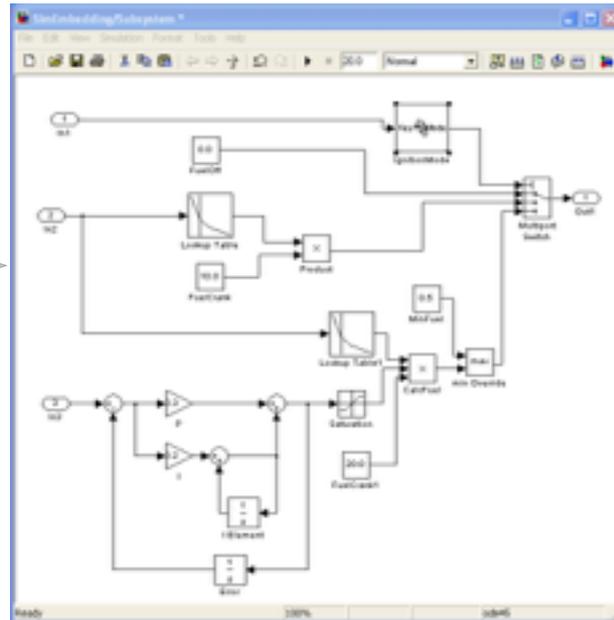
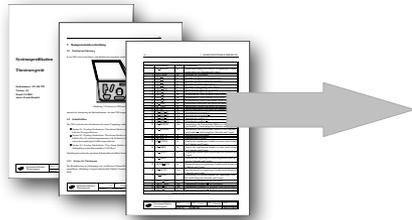
Automotive Software Development: Process



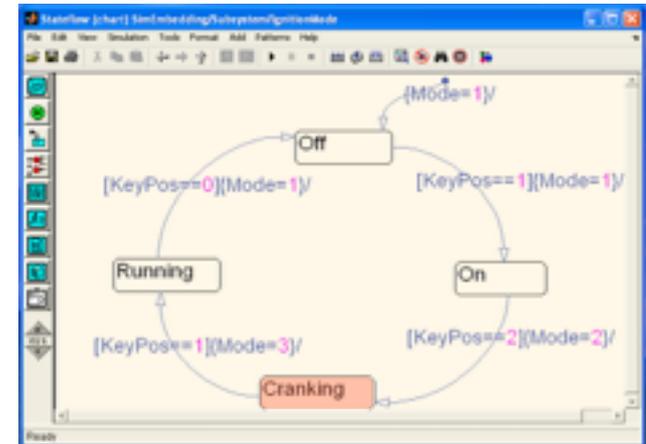
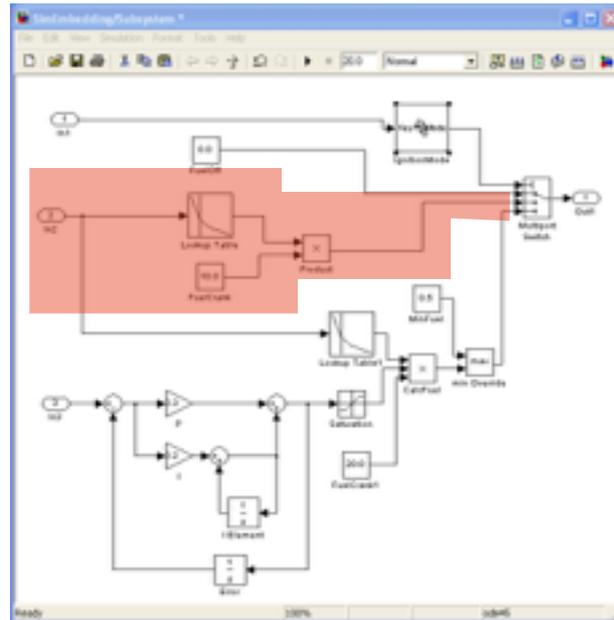
State-of-Art Development: Deficit by lacking models or inadequate use

- 1.Example Component Specification: Lacking domain concepts
- 2.Example Requirements Specification: Lacking descriptive techniques
- 3.Example Implementation: Lacking view integration
- 4.Example Component Design: Lacking consistency analysis
- 5.Example Component Evolution: Lacking maintenance support
- 6.Example Component Verification: Lacking property verification
- 7.Example Verification Specification: Lacking test case generation
- 8.Example system integration: Lacking design space exploration

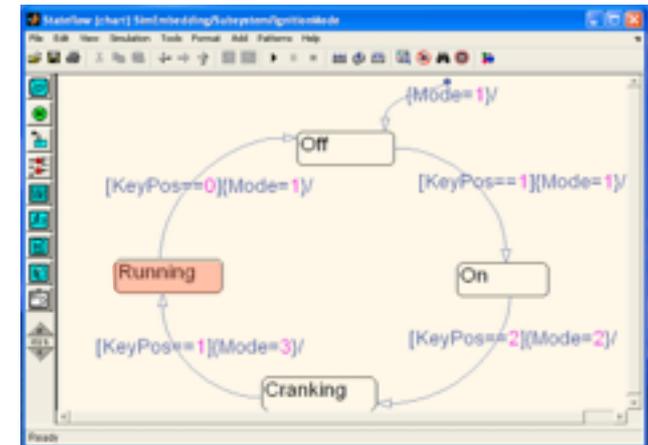
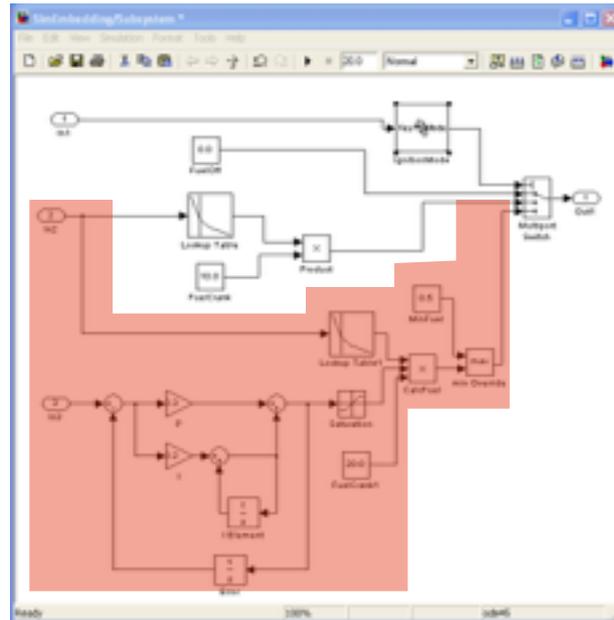
Example 1: Component specification



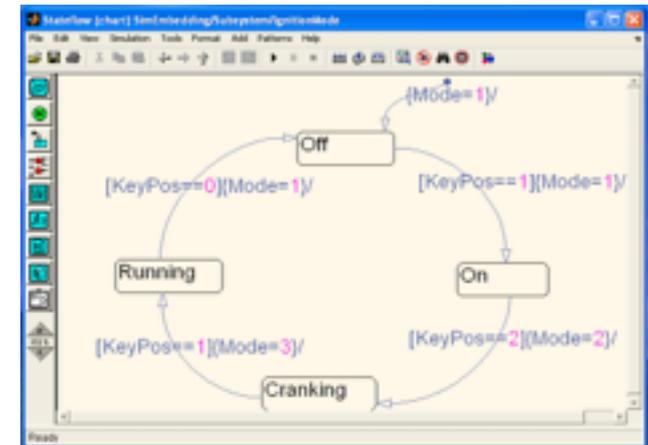
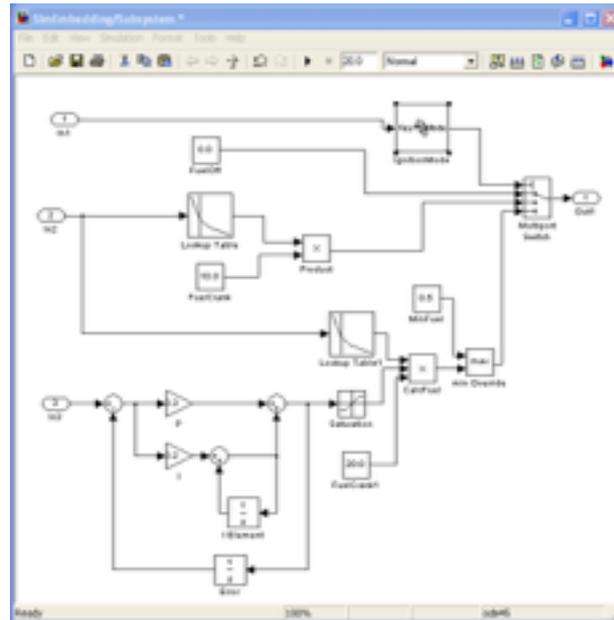
Example 1: Component specification



Example 1: Component specification



Example 1: Component specification

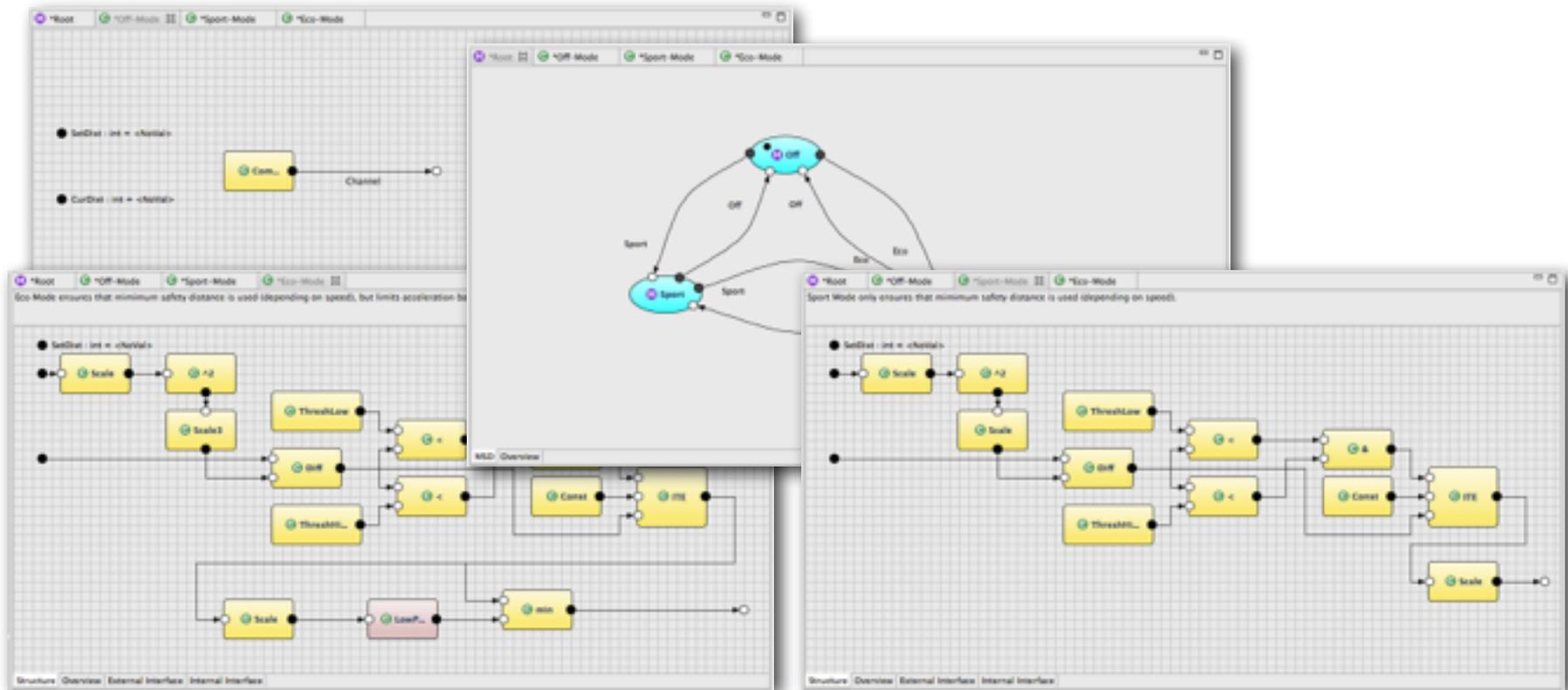


State-of-art: Inadequate models - Models capture technical solution

Problem: Lacking concepts of problem domain

Consequence: Inadequate models of functionality of application

Domain-specific models: Modes of operation



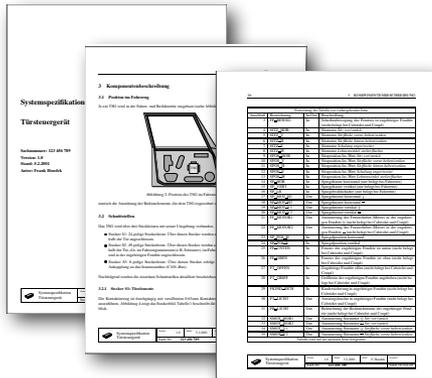
Improvement: Adequate models - Application domain vs technology domain

Method: Extension of modeling techniques by domain concepts

Further examples: Signal classes (state/event signals), timing constraints

Improvement: More effective modeling, more targeted use

Example 2: Requirement Specification



40 6 FUNKTIONEN

Signal	Fahrzeugtyp	TSU	Reaktion
	Limousine Rechtslenker	Links	— ignorieren —
		Rechts	Bewegen der Scheibe hinten rechts mittels S2_FF_MOTOR1 und S2_FF_MOTOR2
	Coopé, Cabriolet Rechtslenker	Beide	— ignorieren —
S2_FFHB	Limousine Linkslenker	Links	Bewegen der Scheibe links hinten mittels S2_FF_MOTOR1 und S2_FF_MOTOR2, wenn S2_FKIND_SICH nicht gesetzt
		Rechts	Bewegen der Scheibe rechts hinten, mittels S2_FF_MOTOR1 und S2_FF_MOTOR2, wenn S2_FKIND_SICH nicht gesetzt
	Coopé, Cabriolet Linkslenker, Rechtslenker	Beide	— ignorieren —

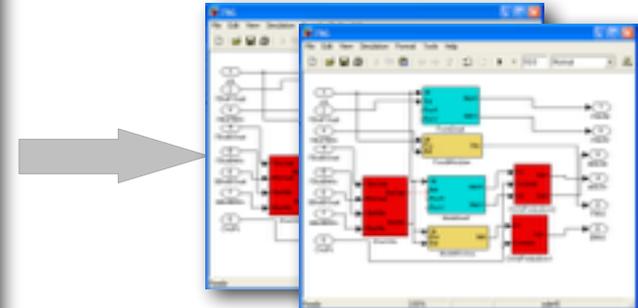
Tabelle 8: Reaktion auf Fenstersteuerungsbefehle.

Öffnen einer dem TSG zugeordneten Scheibe Falls ein Signal anliegt, das das Öffnen einer Scheibe zur Folge hat (siehe Tabelle 8), wird auf dem entsprechenden Motor die Spannung $-12V$ angelegt. Liegt die Batteriespannung zum Beginn der Scheibenbewegung unterhalb von 10V, so wird die Scheibenbewegung nicht durchgeführt. Statt dessen wird die CAN-Botschaft B_JOW_WIN = 1 gesendet. Für die Bewegung sind folgende Fälle zu beachten:

- Ist die relevante Schalterstellung gleich *Fenster runter man.* oder ist die relevante CAN-Botschaft WIN_OP = 01, so wird die Scheibe nach unten bewegt. Die Bewegung endet, wenn
 - das entsprechende Signal nicht mehr anliegt (bzw. nicht mehr gesendet wird),
 - oder sich die Scheibe in der unteren Position befindet (d.h. F_UNTEN bzw. FE_UNTEN),
 - oder ein anderer Befehl zum Bewegen dieser Scheibe zu einem späteren Zeitpunkt eingeht; in diesem Fall wird der neue Bewegungsbefehl bearbeitet,
 - oder der Scheibenbewegungssensor (F_BEWEG bzw. FE_BEWEG) keine Signale sendet, obwohl der Scheibenmotor angesteuert wird und sich die Scheibe noch nicht in der unteren Position befindet; in diesem Fall wird die Botschaft ERROR_WIN = 1 gesendet und der Fehlercode 0x35 in den Fehlerspeicher eingetragen (siehe Abschnitt 6.10)
 - oder die Ansteuerung länger als 3 sec. dauert, ohne daß erkannt wird, daß sich die Scheibe in der unteren Position befindet; in diesem Fall wird die Botschaft ERROR_WIN = 1 gesendet und der Fehlercode 0x35 in den Fehlerspeicher eingetragen (siehe Abschnitt 6.10).
- Wurde die Schalterstellung *Fenster runter auto.* oder die CAN-Botschaft WIN_OP = 10 erkannt, so wird die Scheibe solange nach unten bewegt, bis
 - sich die Scheibe in der unteren Position befindet (d.h. F_UNTEN bzw. FE_UNTEN),
 - oder ein anderer Befehl zum Bewegen dieser Scheibe zu einem späteren Zeitpunkt eingeht; in diesem Fall wird der neue Bewegungsbefehl bearbeitet,
 - oder der Scheibenbewegungssensor (F_BEWEG bzw. FE_BEWEG) keine Signale sendet, obwohl der Scheibenmotor angesteuert wird und sich die Scheibe noch nicht in der unteren Position befindet; in diesem Fall wird die Botschaft ERROR_WIN = 1 gesendet und der Fehlercode 0x35 in den Fehlerspeicher eingetragen (siehe Abschnitt 6.10)

²Z.B. Gestängebruch.

Systemspezifikation Türsteuergert	Version	1.0	Datum	5.2.2001	Autor	F. Houdek	Freigegeben
	Sach-Nr.	133 456 789			Seite 40 von 49		



State-of-art: Lacking models - No modeling of required (forbidden) behavior

Problem: Lacking models for non-constructive specifications

Consequence: No explicit reference specifications for (early) verification

Descriptive Techniques: Interaction Sequences

The screenshot displays a software development tool interface with three overlapping windows:

- Use Case Window:** Shows a use case titled "Safety Turn Off by Braking" with ID "S2". The description states: "If the Driver issues a break command via the Brake pedal then the Adaptive Cruise Control is turned off".
- Message Sequence Chart Editor:** Shows an interaction between "Driver" and "AdaptiveCruiseControl". The Driver sends an "Active()" message to AdaptiveCruiseControl, followed by a return message. A dashed line indicates a time constraint of "10" units.
- Advanced Contracts Window:** Displays a table of verification results for properties related to the use case.

Property	Verification con	Last check date	Last check res.
Property 'CmdAcc == 0' is false After 'breakCmd > 0' until 'breakCmd == 0'.	Acceleration...	Fri Sep 06 0...	SUCCESS
Property 'CmdAcc <= 0' is false After 'breakCmd == 0' until 'breakCmd == 0'.	Acceleration...	Fri Sep 06 0...	FAIL

Below the table, there are controls for verification: Timeout in seconds (50), Prover to be used (NuSMV), and Number of analyzed steps (10). A section for defining properties is also visible:

Property 'P' is false After 'Q' until 'R':
Q: breakCmd >= 0
P: CmdAcc <= 0
R: breakCmd == 0
Verification context: AccelerationControl

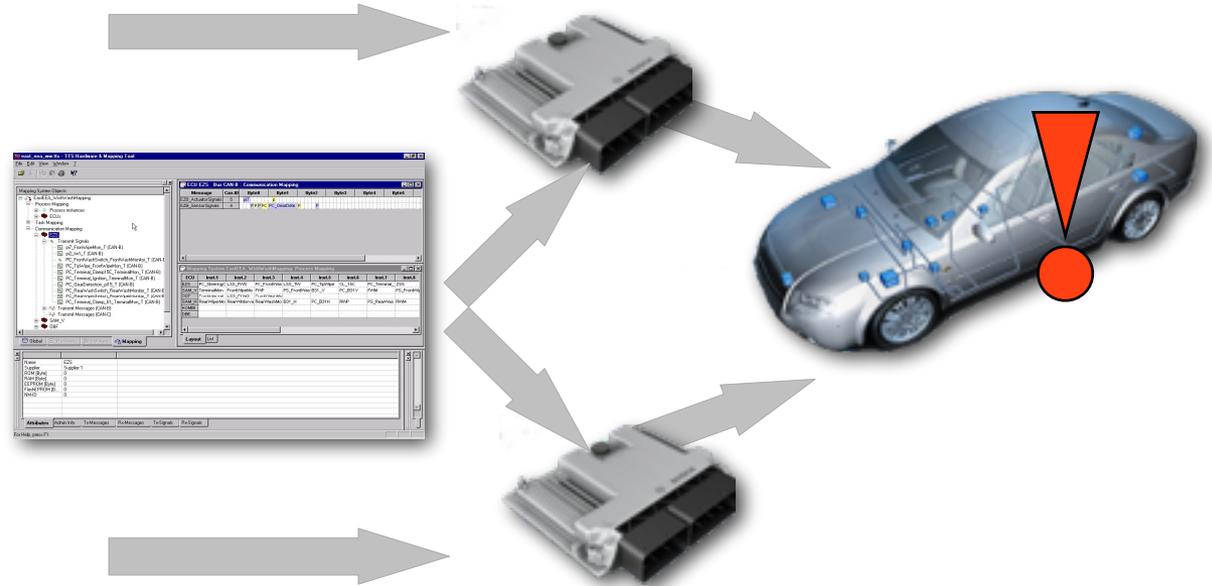
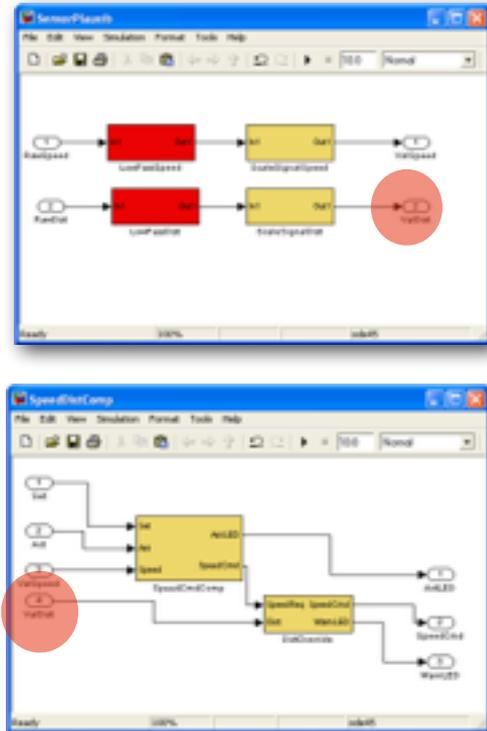
Improvement: Early phases - Specifications of requirements

Methods: Description of intended blackbox behavior

Further examples: Property languages

Improvement: Specification without details on construction

Example 3: Implementation

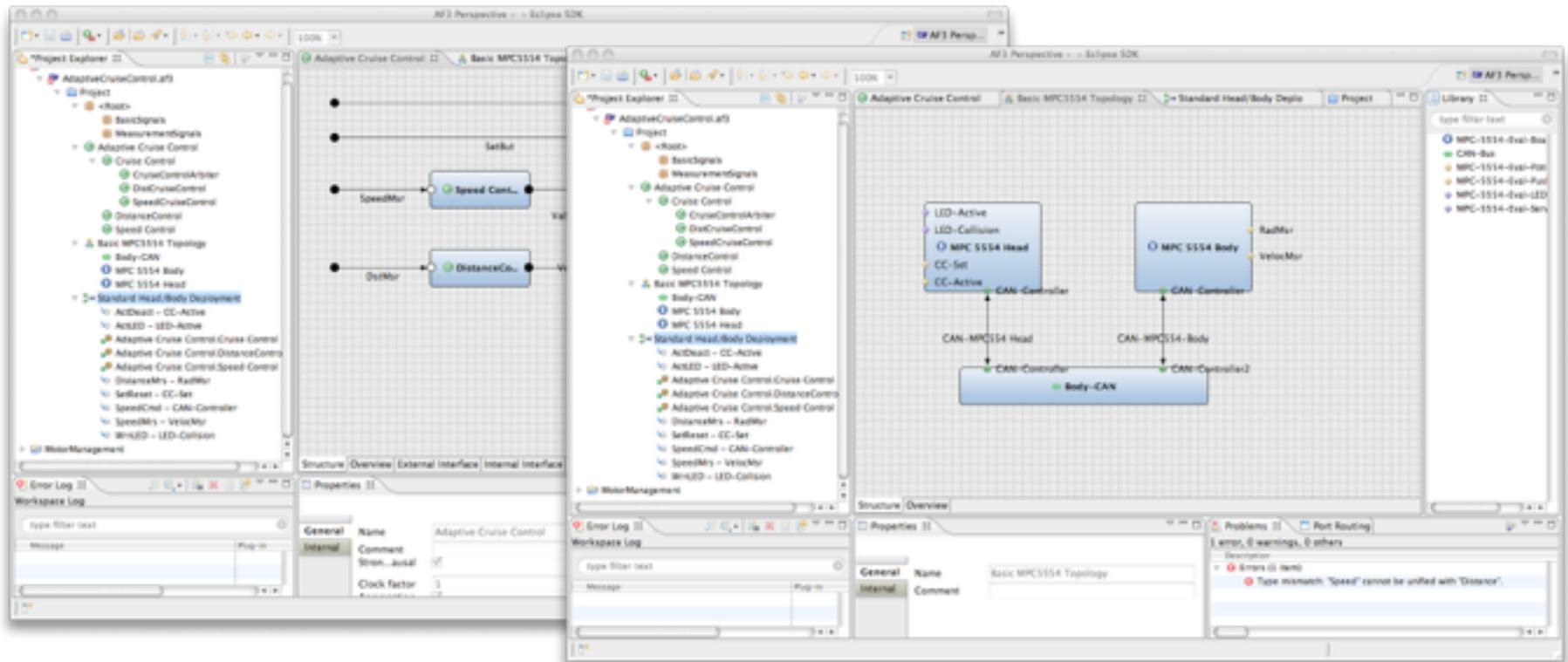


State-of-art: Isolated models - Models for individual steps of development

Problem: Gaps in development process

Consequence: Defects by contradicting decisions in later phases

Integrated models: Component-topology-mapping



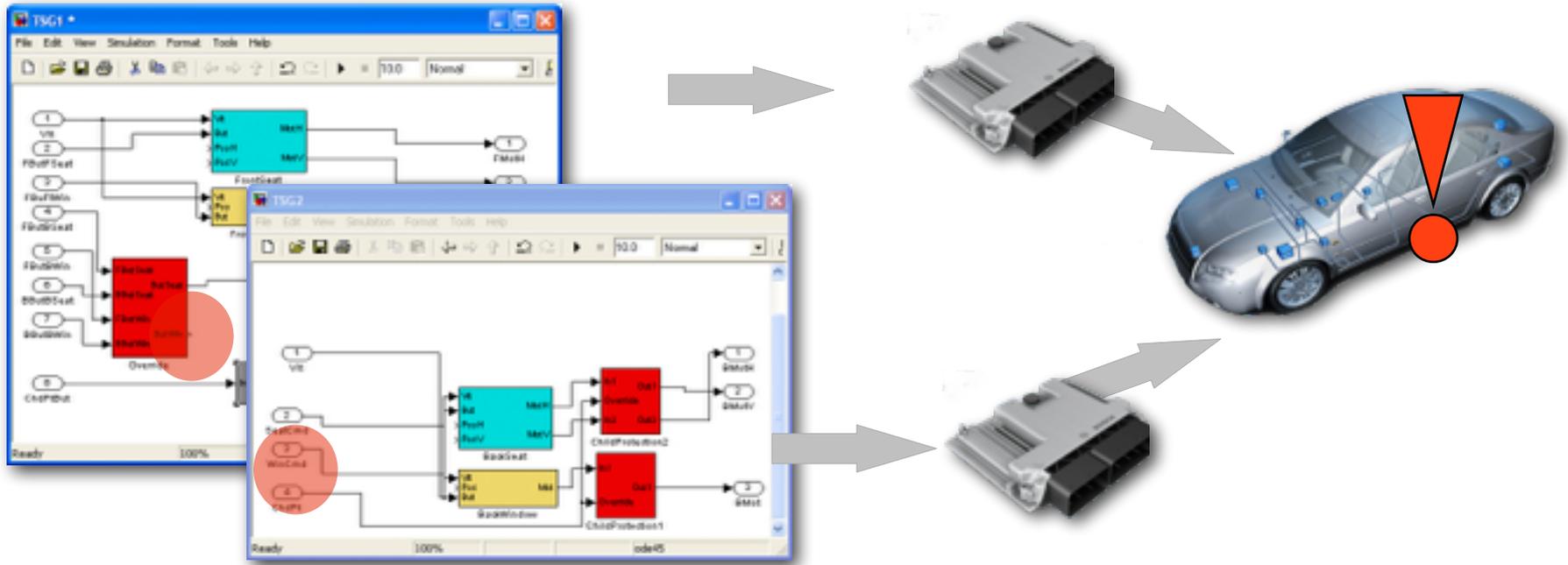
Improvement: Integrated models - Coherent description of system

Methods: Integrated formalization of all system views

Further Examples: Test case refinement (design-level/implementation level)

Improvement: Avoidance of redundancy/inconsistency

Example 4: Component Design



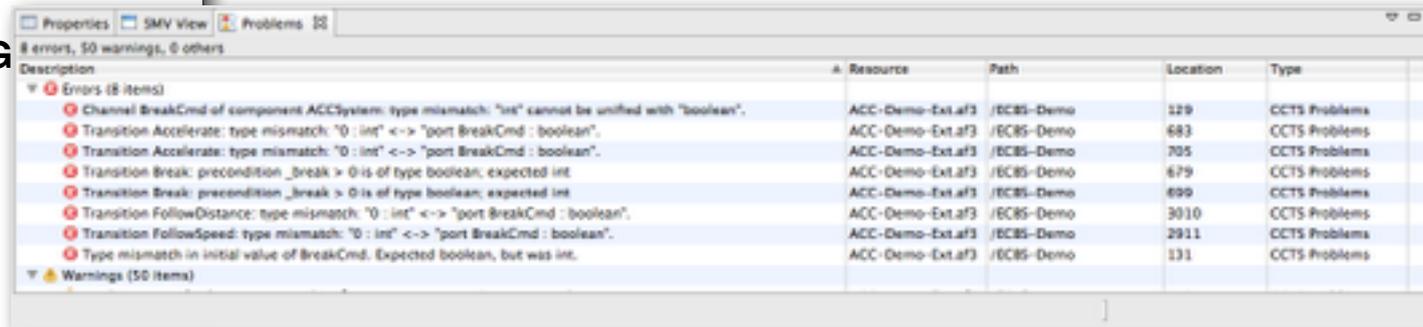
State-of-art: Late quality assurance - Late detection of defects

Problem: Lacking analysis support for early development steps

Consequence: Conservation of early defects throughout the development process

Analysis Method: Conformance Checking of Models

Modeling G



Description	Resource	Path	Location	Type
Channel BreakCmd of component ACCSystem: type mismatch: 'int' cannot be unified with 'boolean'.	ACC-Demo-Ext.af3	/ECBS-Demo	129	CCTS Problems
Transition Accelerate: type mismatch: '0 : int' <-> 'port BreakCmd : boolean'.	ACC-Demo-Ext.af3	/ECBS-Demo	683	CCTS Problems
Transition Accelerate: type mismatch: '0 : int' <-> 'port BreakCmd : boolean'.	ACC-Demo-Ext.af3	/ECBS-Demo	705	CCTS Problems
Transition Break: precondition_break > 0 is of type boolean; expected int	ACC-Demo-Ext.af3	/ECBS-Demo	679	CCTS Problems
Transition Break: precondition_break > 0 is of type boolean; expected int	ACC-Demo-Ext.af3	/ECBS-Demo	699	CCTS Problems
Transition FollowDistance: type mismatch: '0 : int' <-> 'port BreakCmd : boolean'.	ACC-Demo-Ext.af3	/ECBS-Demo	3010	CCTS Problems
Transition FollowSpeed: type mismatch: '0 : int' <-> 'port BreakCmd : boolean'.	ACC-Demo-Ext.af3	/ECBS-Demo	2911	CCTS Problems
Type mismatch in initial value of BreakCmd. Expected boolean, but was int.	ACC-Demo-Ext.af3	/ECBS-Demo	131	CCTS Problems

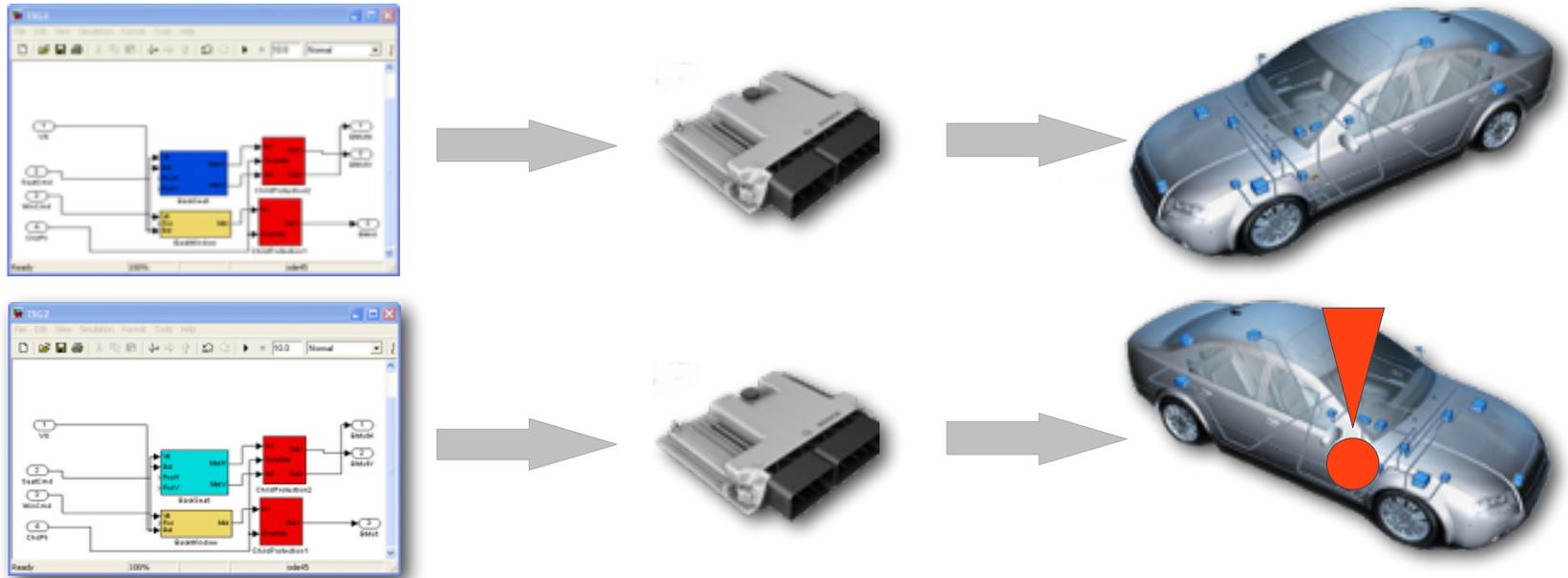
Improvement: Early Quality assurance - Early sound models via front-loading QS

Method: Automatic checks for model structure/model content

Further examples: Type conformance, definedness of signals

Improvement: Early defect elimination

Example 5: Model Maintenance



State-of-art: Unsystematic reuse - Introduction of change anomalies

Problem: Lacking Analysis methods supporting maintenance

Consequence: Erschwerung und Verteuerung von Wartungsaufgaben

Analysis Method: Clone Detection

The screenshot displays the results of a clone detection analysis. A central window titled 'Clone table (Main)' shows a table of clones. To the left, a circuit diagram shows an input signal branching into two parallel paths, each containing a gain block and a summing junction. To the right, another circuit diagram shows a similar structure but with a different component configuration. Below the table, two more circuit diagrams are shown, one with red highlights indicating detected clones. On the far right, two system model diagrams are shown, each with a table of parameters and a block diagram below it.

Clone ID	Start	End	Count	Similarity	Component
Clone 1	100	105	2	100%	addition
Clone 2	110	115	2	100%	addition
Clone 3	120	125	2	100%	addition
Clone 4	130	135	2	100%	addition
Clone 5	140	145	2	100%	addition
Clone 6	150	155	2	100%	addition
Clone 7	160	165	2	100%	addition
Clone 8	170	175	2	100%	addition
Clone 9	180	185	2	100%	addition
Clone 10	190	195	2	100%	addition
Clone 11	200	205	2	100%	addition
Clone 12	210	215	2	100%	addition
Clone 13	220	225	2	100%	addition
Clone 14	230	235	2	100%	addition
Clone 15	240	245	2	100%	addition
Clone 16	250	255	2	100%	addition
Clone 17	260	265	2	100%	addition
Clone 18	270	275	2	100%	addition
Clone 19	280	285	2	100%	addition
Clone 20	290	295	2	100%	addition
Clone 21	300	305	2	100%	addition
Clone 22	310	315	2	100%	addition
Clone 23	320	325	2	100%	addition
Clone 24	330	335	2	100%	addition
Clone 25	340	345	2	100%	addition
Clone 26	350	355	2	100%	addition
Clone 27	360	365	2	100%	addition
Clone 28	370	375	2	100%	addition
Clone 29	380	385	2	100%	addition
Clone 30	390	395	2	100%	addition
Clone 31	400	405	2	100%	addition
Clone 32	410	415	2	100%	addition
Clone 33	420	425	2	100%	addition
Clone 34	430	435	2	100%	addition
Clone 35	440	445	2	100%	addition
Clone 36	450	455	2	100%	addition
Clone 37	460	465	2	100%	addition
Clone 38	470	475	2	100%	addition
Clone 39	480	485	2	100%	addition
Clone 40	490	495	2	100%	addition
Clone 41	500	505	2	100%	addition
Clone 42	510	515	2	100%	addition
Clone 43	520	525	2	100%	addition
Clone 44	530	535	2	100%	addition
Clone 45	540	545	2	100%	addition
Clone 46	550	555	2	100%	addition
Clone 47	560	565	2	100%	addition
Clone 48	570	575	2	100%	addition
Clone 49	580	585	2	100%	addition
Clone 50	590	595	2	100%	addition
Clone 51	600	605	2	100%	addition
Clone 52	610	615	2	100%	addition
Clone 53	620	625	2	100%	addition
Clone 54	630	635	2	100%	addition
Clone 55	640	645	2	100%	addition
Clone 56	650	655	2	100%	addition
Clone 57	660	665	2	100%	addition
Clone 58	670	675	2	100%	addition
Clone 59	680	685	2	100%	addition
Clone 60	690	695	2	100%	addition
Clone 61	700	705	2	100%	addition
Clone 62	710	715	2	100%	addition
Clone 63	720	725	2	100%	addition
Clone 64	730	735	2	100%	addition
Clone 65	740	745	2	100%	addition
Clone 66	750	755	2	100%	addition
Clone 67	760	765	2	100%	addition
Clone 68	770	775	2	100%	addition
Clone 69	780	785	2	100%	addition
Clone 70	790	795	2	100%	addition
Clone 71	800	805	2	100%	addition
Clone 72	810	815	2	100%	addition
Clone 73	820	825	2	100%	addition
Clone 74	830	835	2	100%	addition
Clone 75	840	845	2	100%	addition
Clone 76	850	855	2	100%	addition
Clone 77	860	865	2	100%	addition
Clone 78	870	875	2	100%	addition
Clone 79	880	885	2	100%	addition
Clone 80	890	895	2	100%	addition
Clone 81	900	905	2	100%	addition
Clone 82	910	915	2	100%	addition
Clone 83	920	925	2	100%	addition
Clone 84	930	935	2	100%	addition
Clone 85	940	945	2	100%	addition
Clone 86	950	955	2	100%	addition
Clone 87	960	965	2	100%	addition
Clone 88	970	975	2	100%	addition
Clone 89	980	985	2	100%	addition
Clone 90	990	995	2	100%	addition

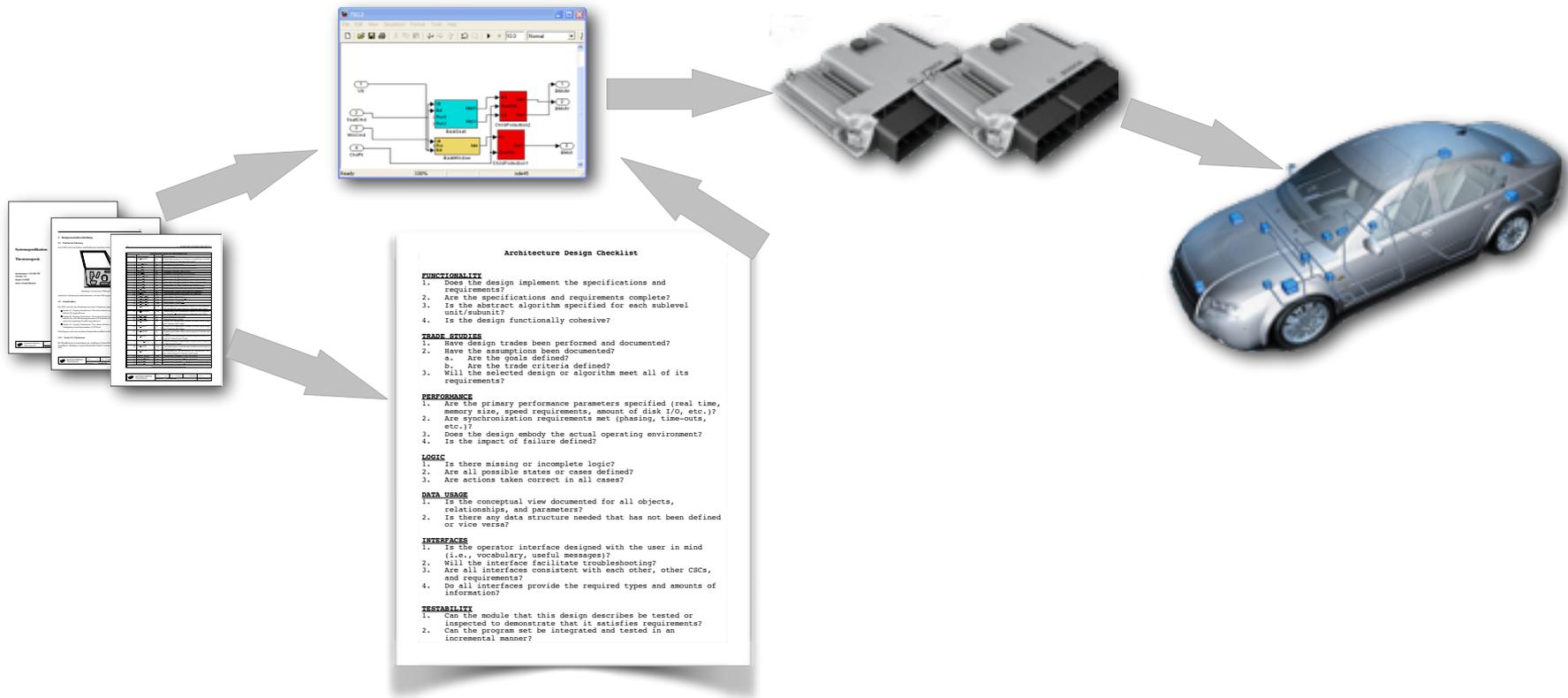
Improvement: Identification of Identical Parts - Avoidance of change anomalies

Methods: Automatic analysis of (structure of) component behavior

Further examples: Requirements clones, IP-Libraries, product lines

Improvement: Simplification of model maintenance

Example 6: Component Verification

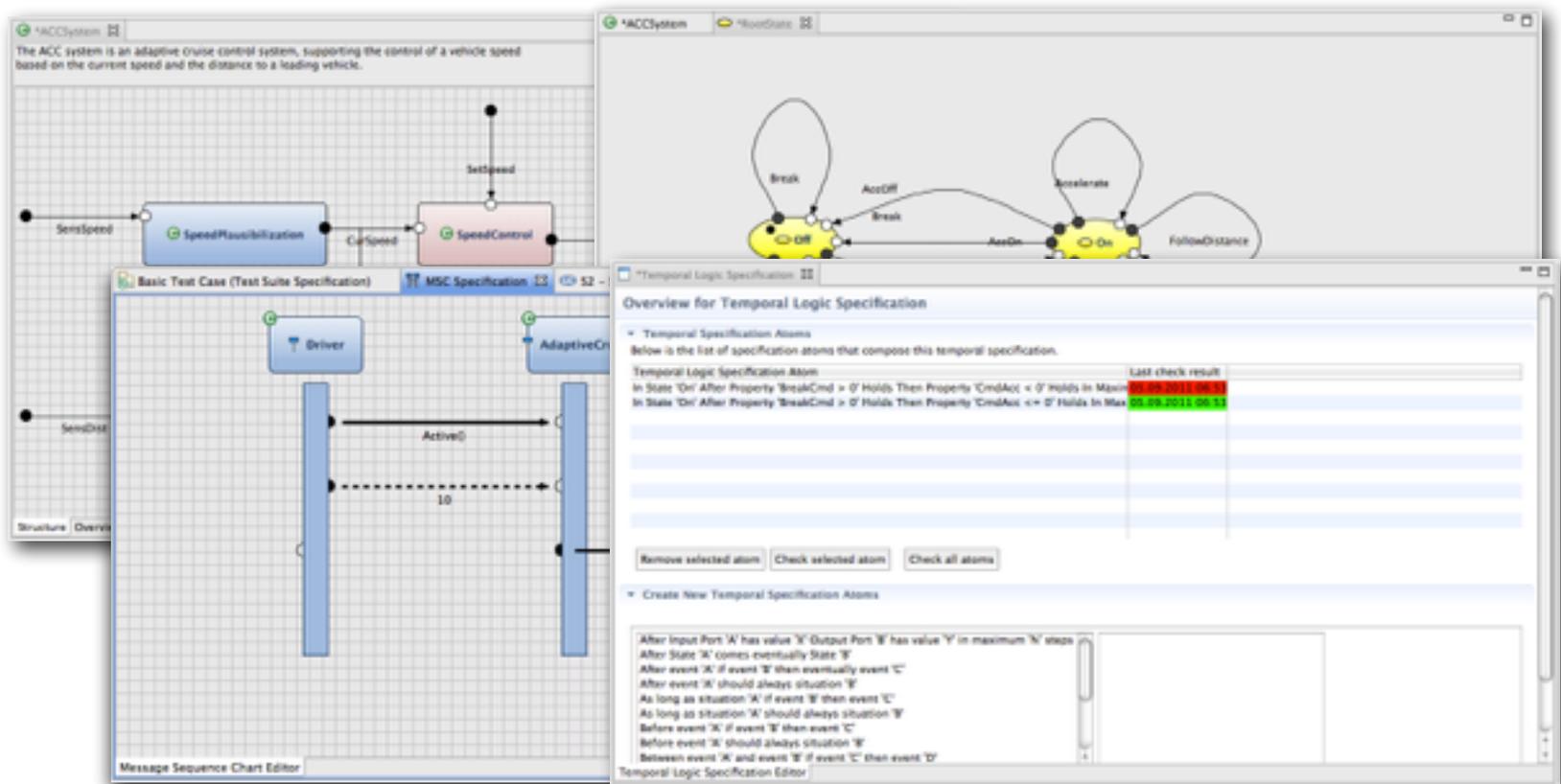


State-of-art: Manual Review Prozess - Expensive early verification

Problem: Lacking automatization of early validation

Consequence: Expensive and error-prone verification of core properties

Analysis method: Property Verification



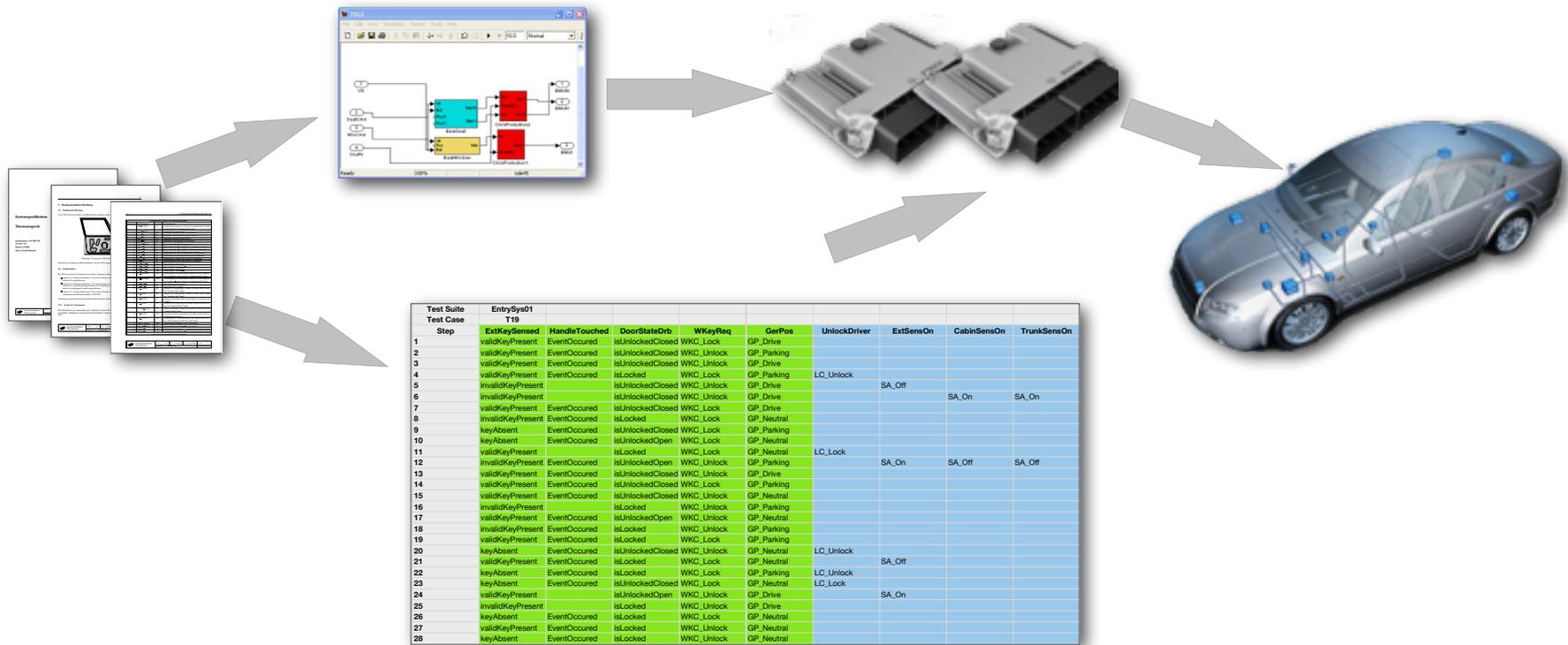
Improvement: Automated property verification - Systematic early verification

Method: Automated analysis by formal verification

Further examples: Checking of determinism

Improvement: Early defect identification

Example 7: Verification

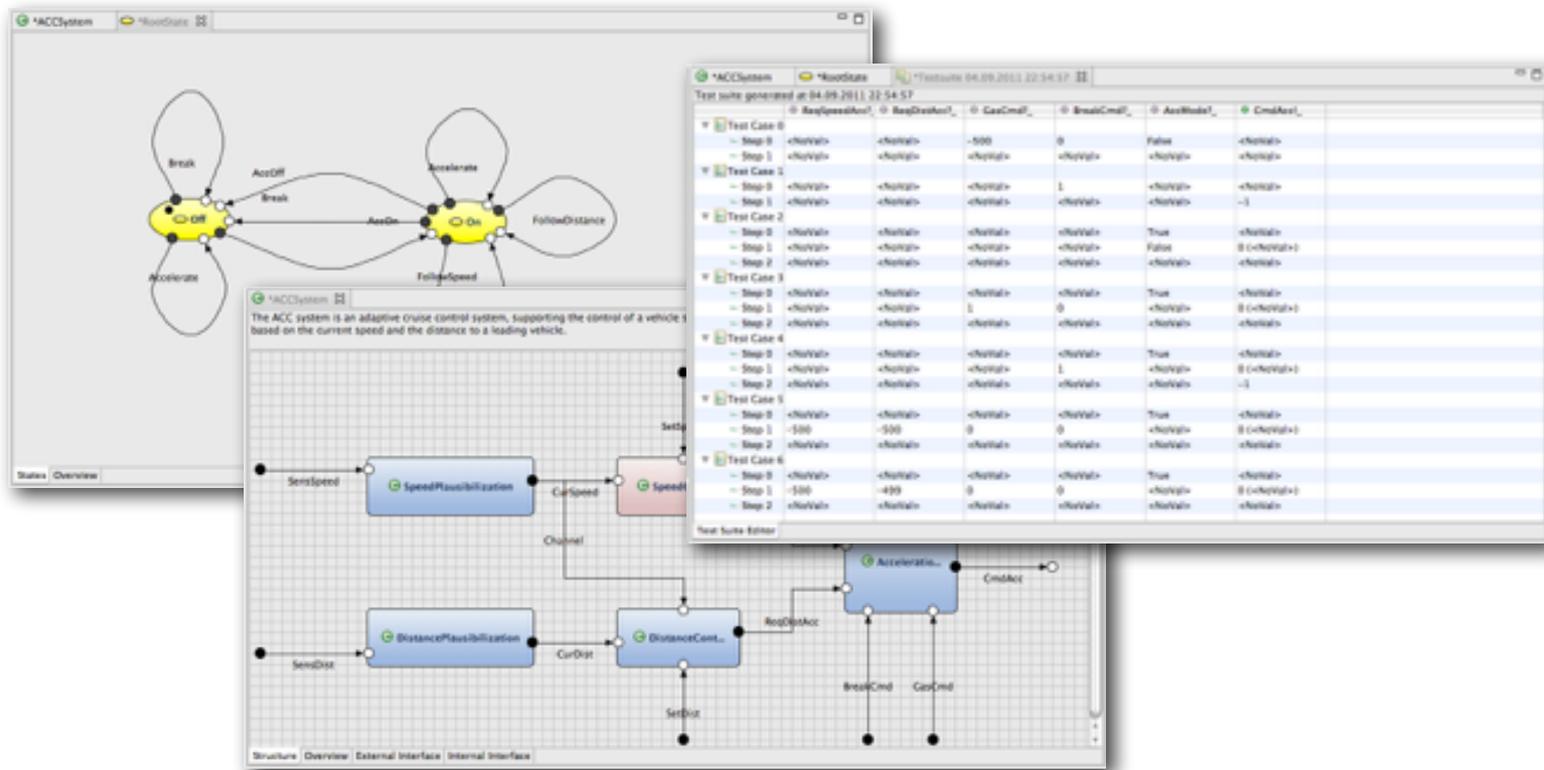


State-of-art: Manual test case definition - Expensive test suite construction

Problem: Lacking automatization of test case definition

Consequence: Expensive and un-systematic verification of functionality

Synthesis method: Test case generation



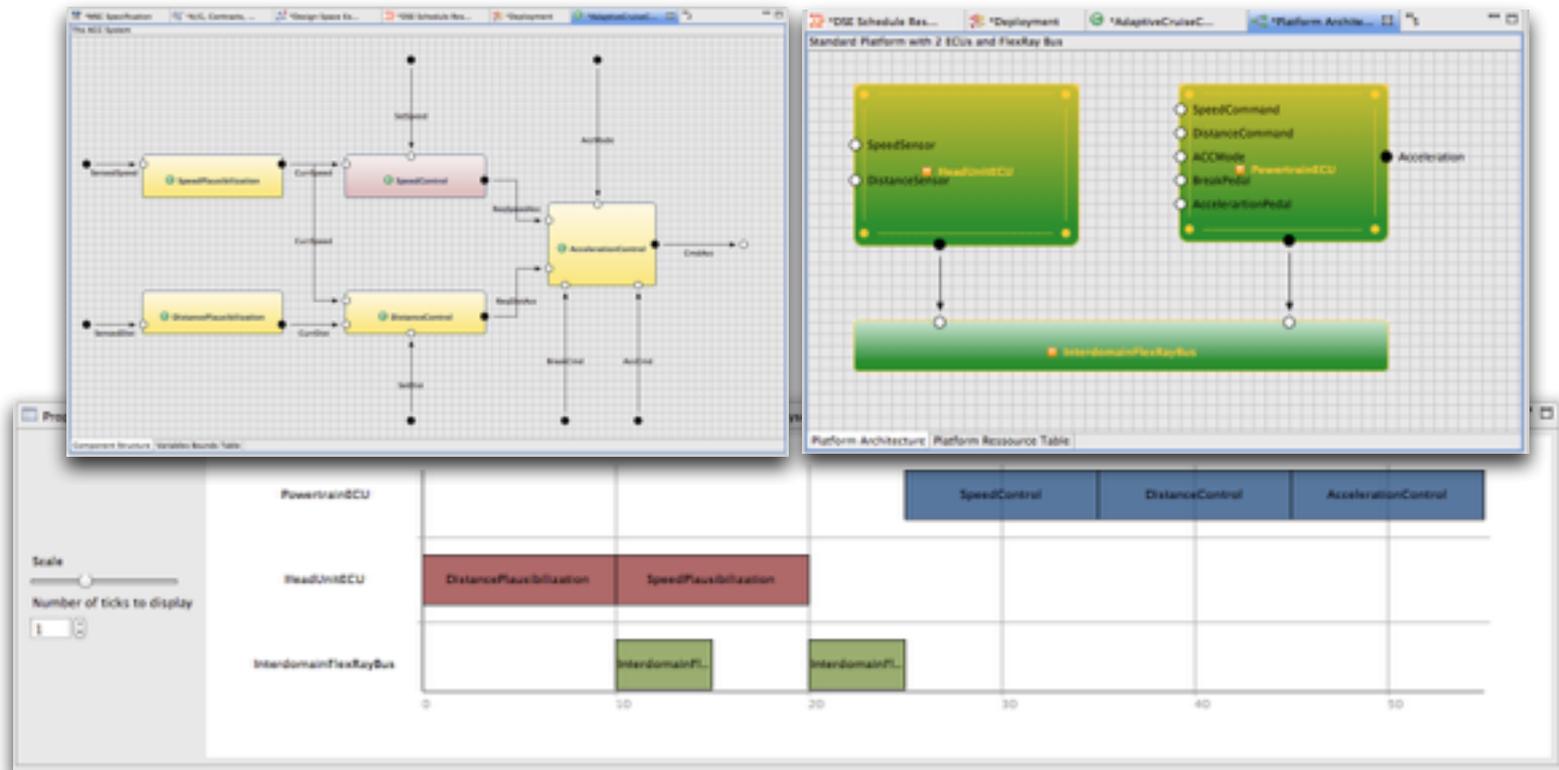
Improvement: Synthesis of models - Optimization of test suit construction

Method: Automatic model construction by generative search

Further examples: Load tests, robustness tests, model validation

Improvement: Improved efficiency of system verification

Synthesis method: Schedule generation



Improvement: Automated development steps - Generated modes

Method: Automated model construction by generative search

Further examples: (Mixed-criticality-)deployment generation, code generation

Improvement: Accelerated exploration of design alternatives

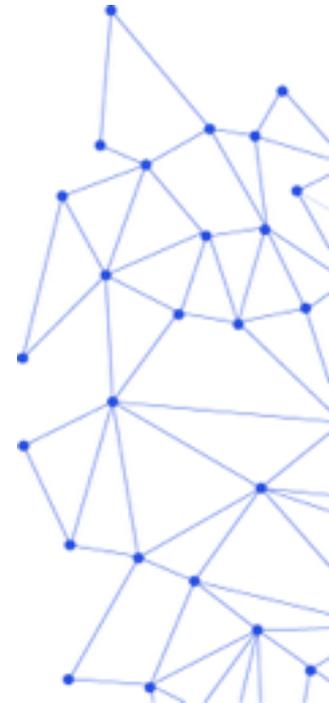
March 2015

Clone Analysis in Model-Based Development

McGill NECSIS Workshop 2015

Bernhard Schätz

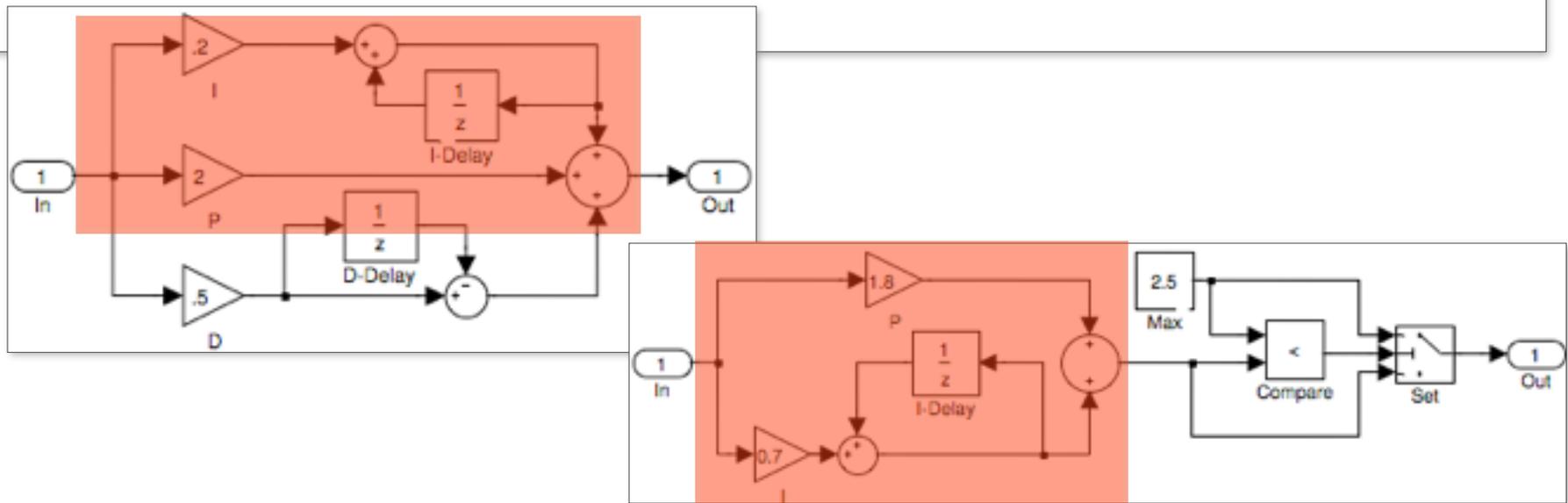
fortiss GmbH
An-Institut Technische Universität München



What's a Clone?

Software clones are segments of code that are similar according to some definition of similarity.

Ira Baxter, 2002

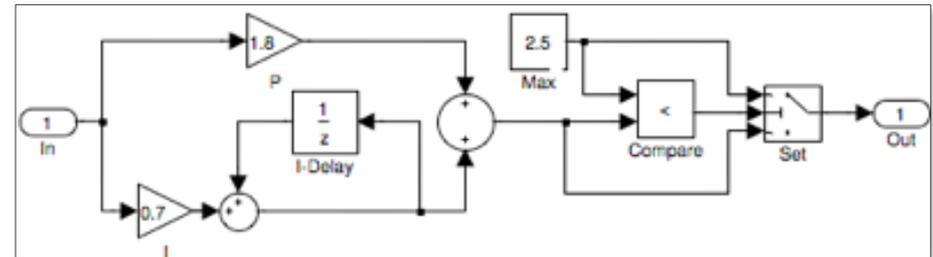
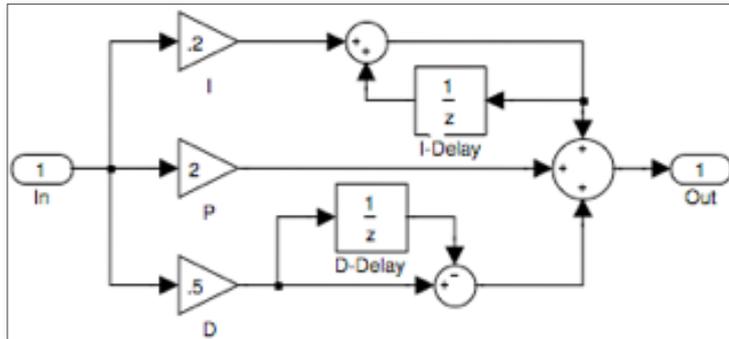


Reusing functionality is good engineering practice

Not being aware of reuse is a dangerous pitfall

Clone analysis detects unwanted forms of reuse

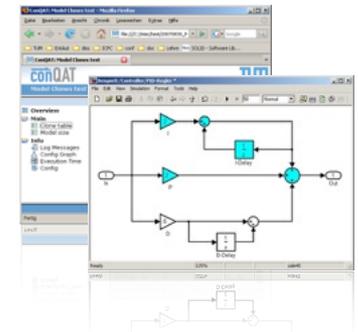
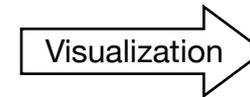
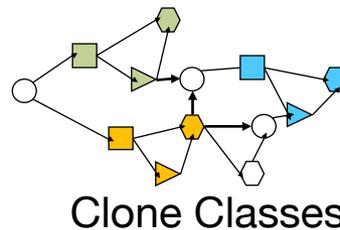
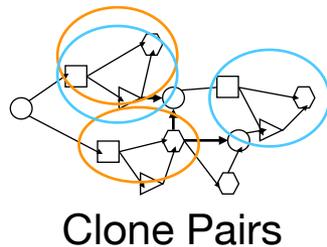
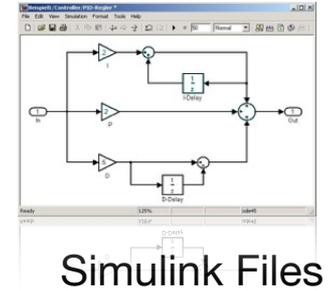
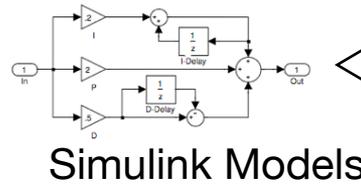
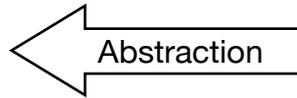
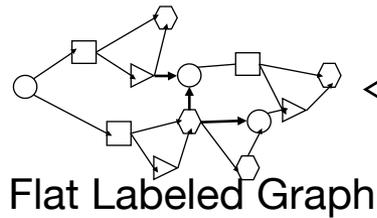
Dataflow Clones



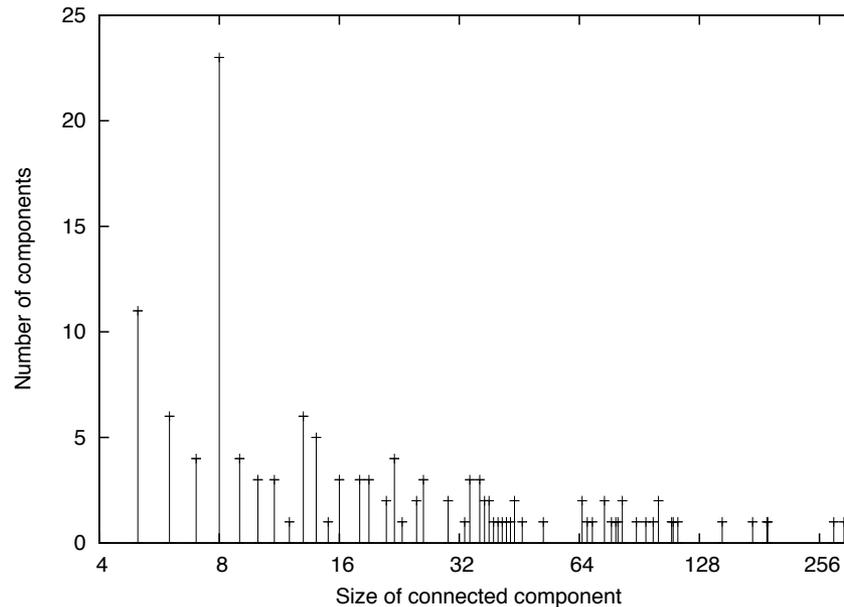
Basic Criteria:

- Functionally independent (Abstracted elements)
- Reusable (Connected)
- Functionally complex (Size of functional elements)
- General (Number of instances)

Clone Detection Pipeline



Practical Case Study Results



Simulink/TargetLink Model (ca. 20.000 blocks, 71 files):

- Identified: 139 clone classes after filtering
- Most clones are relatively small & singular/infrequent
- Most clones affect several files/transcend several hierarchies
- Includes clones of library blocks
- 37% of relevant blocks are part of at least one clone class
- Clone elimination substantially reduces models

Practical Case Study Results

Clone Size	Number of Clone Classes
5 - 10	76
11 - 15	35
15 -20	17
> 20	11

Simulink/TargetLink Model (ca. 20.000 blocks, 71 files):

- Identified: 139 clone classes after filtering
- Most clones are relatively small & singular/infrequent
- Most clones affect several files/transcend several hierarchies
- Includes clones of library blocks
- 37% of relevant blocks are part of at least one clone class
- Clone elimination substantially reduces models

Practical Case Study Results

Cardinality of Clone Class	Number of Clone Classes
2	108
3	20
4	10
5	1

Simulink/TargetLink Model (ca. 20.000 blocks, 71 files):

- Identified: 139 clone classes after filtering
- Most clones are relatively small & singular/infrequent
- Most clones affect several files/transcend several hierarchies
- Includes clones of library blocks
- 37% of relevant blocks are part of at least one clone class
- Clone elimination substantially reduces models

Practical Case Study Results

Number of Models	Number of Clone Classes
1	43
2	81
3	12
4	2

Simulink/TargetLink Model (ca. 20.000 blocks, 71 files):

- Identified: 139 clone classes after filtering
- Most clones are relatively small & singular/infrequent
- Most clones affect several files/transcend several hierarchies
- Includes clones of library blocks
- 37% of relevant blocks are part of at least one clone class
- Clone elimination substantially reduces models

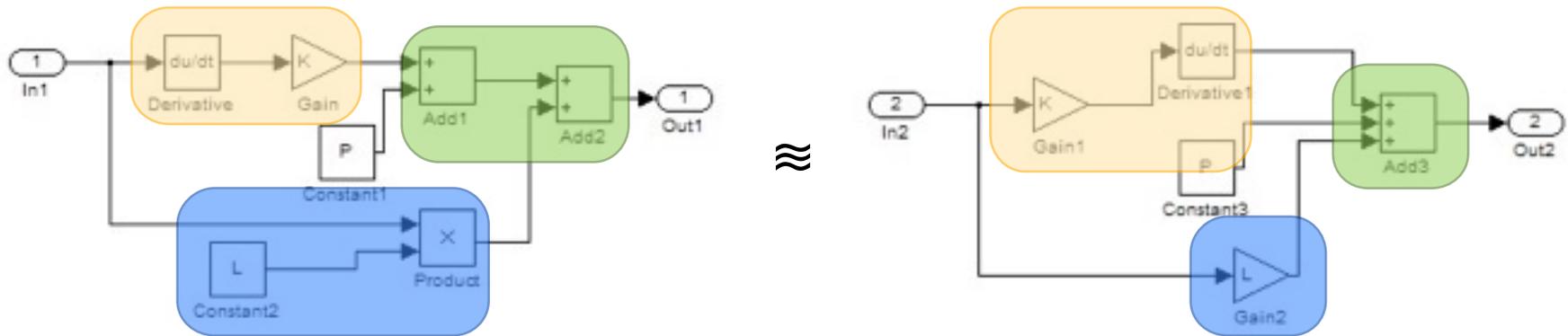
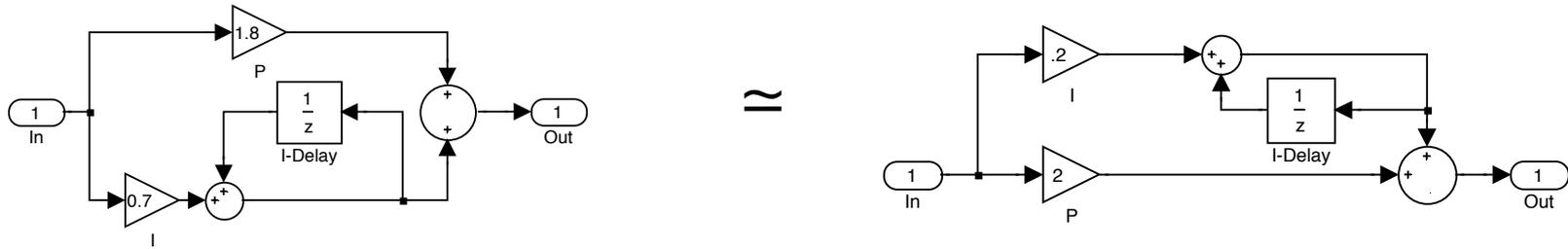
Practical Case Study Results

	Before Elimination		After Elimination	
	# Nodes	#Edges	# Nodes	# Edges
SIM	428	415	387	379
SEM	1741	2029	470	704
ECW	2312	2274	1315	984
AUT	98251	90056	66944	66026

Simulink/TargetLink Model (ca. 20.000 blocks, 71 files):

- Identified: 139 clone classes after filtering
- Most clones are relatively small & singular/infrequent
- Most clones affect several files/transcend several hierarchies
- Includes clones of library blocks
- 37% of relevant blocks are part of at least one clone class
- Clone elimination substantially reduces models

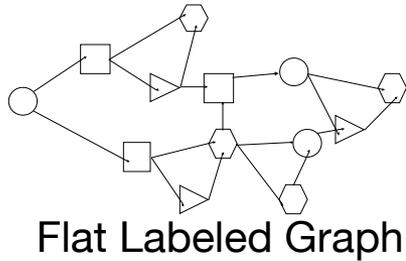
Syntactic vs Semantic Data Flow Clones



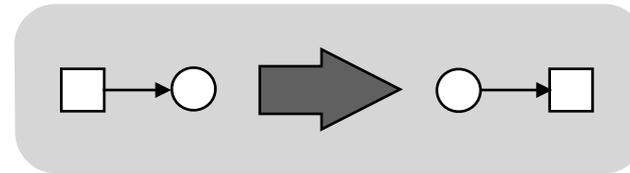
Notion of Similarity

- Syntactic clones (Type 3): Topologically equivalent dataflow (\cong)
- Semantic clones (Type 4): Computationally equivalent dataflow (\approx)

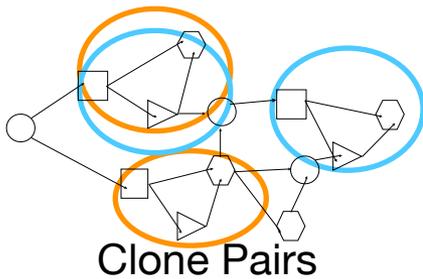
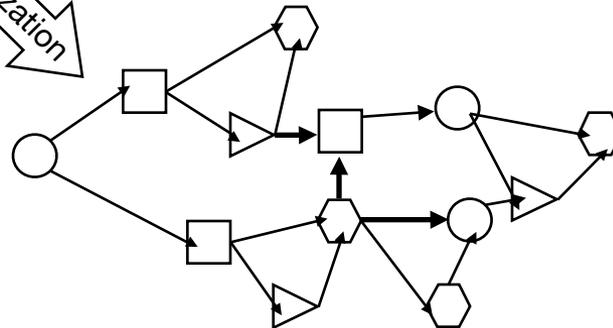
Semantic Clone Detection Pipeline



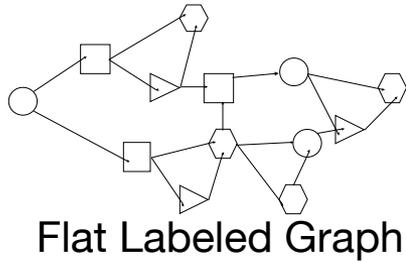
Transformation Rule



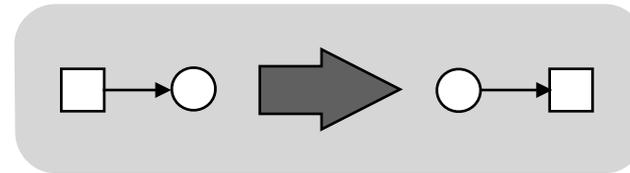
Normalization



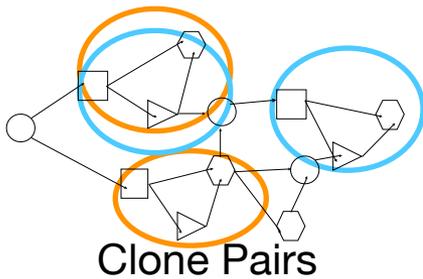
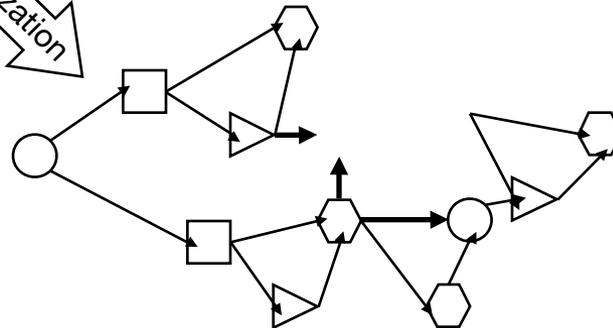
Semantic Clone Detection Pipeline



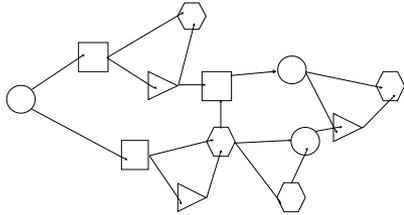
Transformation Rule



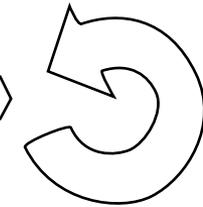
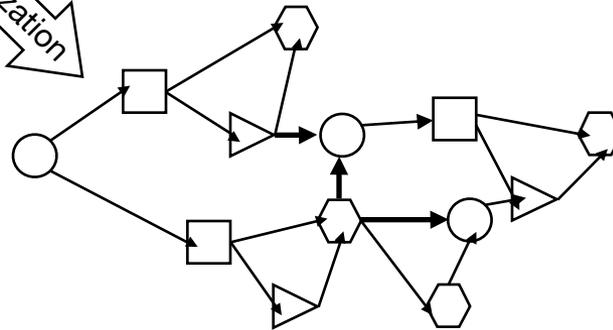
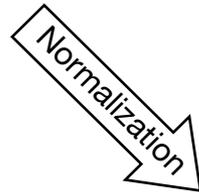
Normalization



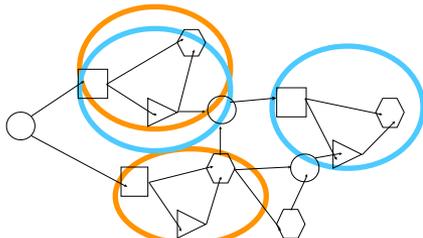
Semantic Clone Detection Pipeline



Flat Labeled Graph

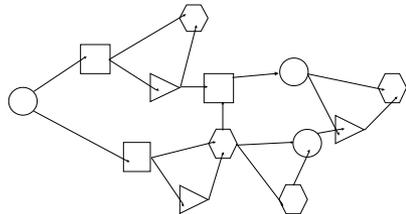


Transformation

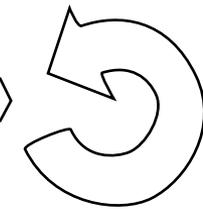
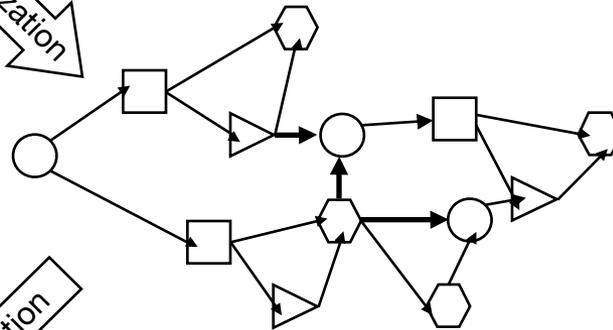
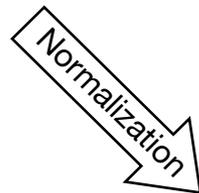


Clone Pairs

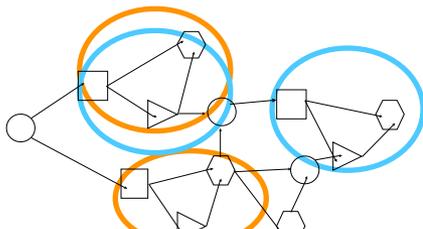
Semantic Clone Detection Pipeline



Flat Labeled Graph



Transformation



Clone Pairs

Automotive Case Study Results

Rule	# Executions
Gain for Multiplying by Constant	104
Bias for Adding a Constant	58
Joining Consecutive Product Blocks	42
Joining Consecutive Sum Blocks	40
Placing Gain Block before Integrator Block	28
Joining Consecutive Gain Blocks	16
Sum Rule in Integration	6
Power Rule	5
Distribution of Multiplication over Addition	4
Replacing Comp. to Const. by Comp. to Zero	4
Placing Gain Block before Derivative Block	2
Joining Consecutive Mux Blocks	2
Joining Consecutive Bias Blocks	2
Trigonometric functions	2
Elimination of Rounding Blocks	2
Math functions	2
Replacing Unary Minus Block by Gain Block	2

Normalization of Simulink model (ca. 1400 blocks):
- Substantial transformation: 321 applications of 16 rules

Automotive Case Study Results

Clone Size	Number of Clones	Number of Clones
4-6	46	43
7-10	11	26
11-15	5	7
16-20	0	6
21-30	1	0
> 30	5	5
Total	68	87

Normalization of Simulink model (ca. 1400 blocks):

- Substantial transformation: 321 applications of 16 rules
- More clones: 87 clones vs. 68 clones

Automotive Case Study Results

Clone Size	Number of Clone Classes	Number of Clone Classes
4-6	31	17
7-10	8	13
11-15	4	4
16-20	0	4
21-30	1	0
> 30	4	4
Average clone	12,7	14,5

Normalization of Simulink model (ca. 1400 blocks):

- Substantial transformation: 321 applications of 16 rules
- More clones: 87 clones vs. 68 clones
- Larger clones: 14.5 blocks vs. 12.7 blocks

Automotive Case Study Results

Clone Class Cardinality	Number of Clone Classes (without)	Number of Clone Classes (with normalization)
2	32	26
3	5	5
4	6	3
5	4	3
6	1	3
7	1	0
8	0	2
9	0	1
Total number of	49	42

Normalization of Simulink model (ca. 1400 blocks):

- Substantial transformation: 321 applications of 16 rules
- More clones: 87 clones vs. 68 clones
- Larger clones: 14.5 blocks vs. 12.7 blocks
- More frequent clones: 42 classes vs. 49 classes

Automotive Case Study Results

Clone Class Cardinality	Number of Clone Classes (without)	Number of Clone Classes (with normalization)
2	32	26
3	5	5
4	6	3
5	4	3
6	1	3
7	1	0
8	0	2
9	0	1
Total number of	49	42

Normalization of Simulink model (ca. 1400 blocks):

- Substantial transformation: 321 applications of 16 rules
- More clones: 87 clones vs. 68 clones
- Larger clones: 14.5 blocks vs. 12.7 blocks
- More frequent clones: 42 classes vs. 49 classes
- New clones: 2 additional classes