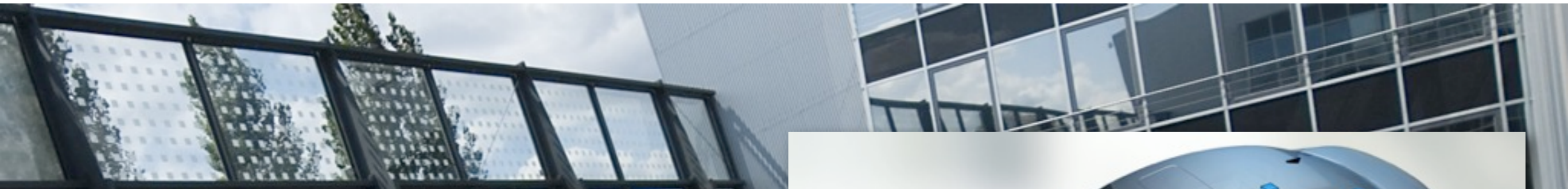


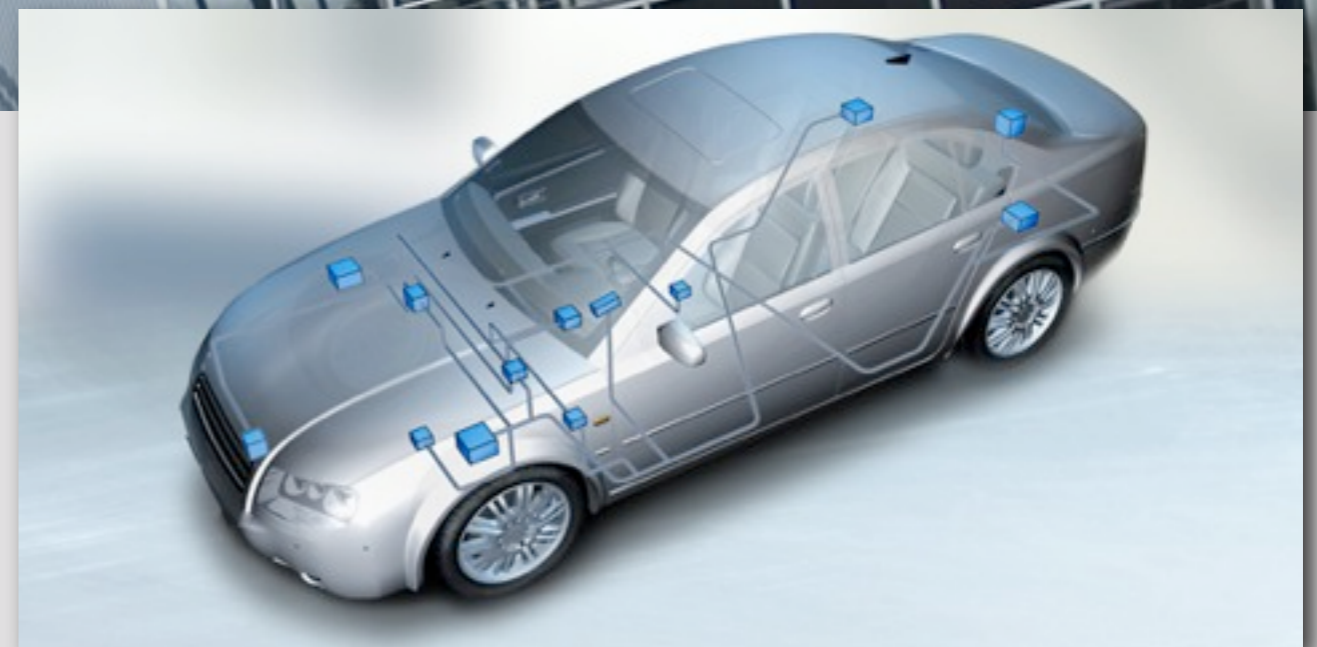
Model-Based Development of Automotive Software

From the State of the Art to Future Possibilities



SimPar 2010
Darmstadt
15. November 2010

Bernhard Schätz
fortiss gGmbH, München



fortiss - Who we are

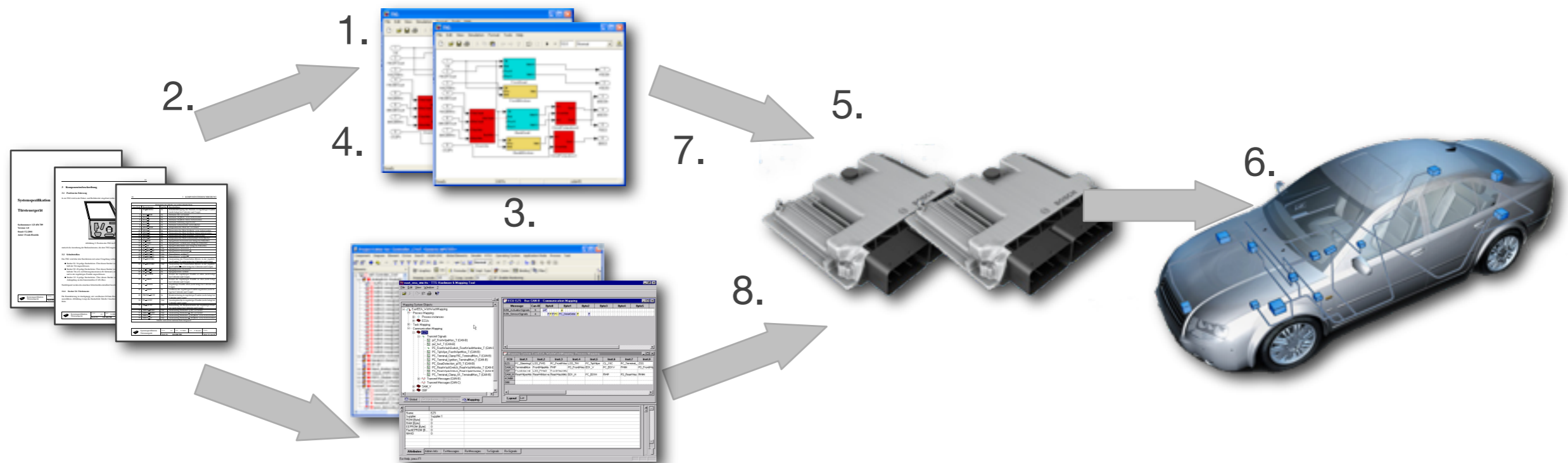


- Private non-profit transfer and research institute of Technische Universität München
- Participators
 - Technische Universität München
 - LfA, Förderbank Bayern
 - Fraunhofer Gesellschaft
- Sponsored by Bavarian Ministry for Economics, Transportation, and Technology since January 1st, 2009
- Focus: Innovations in research and development of software intensive systems for technical and business applications

Development of embedded software: Challenges

- Complexity of domains
 - Application domains, e.g., power-train, body, infotainment
 - Engineering domains, e.g., mechanics, electric/electronics, informatics
 - Complexity of technology
 - Platform, e.g., memory limitations, bandwidth limitations
 - Distribution, e.g., sensors, control units, busses
 - Complexity of functionality
 - Increase in size, e.g., comfort electronics, infotainment
 - Increase in interconnection, e.g., electronic stability, adaptive cruise control
 - Complexity of development
 - Supply chain, e.g., middleware, applications
 - Product line, e.g., equipment, country specifics
- ➔ **Complexity reduction through use of models and tool support**

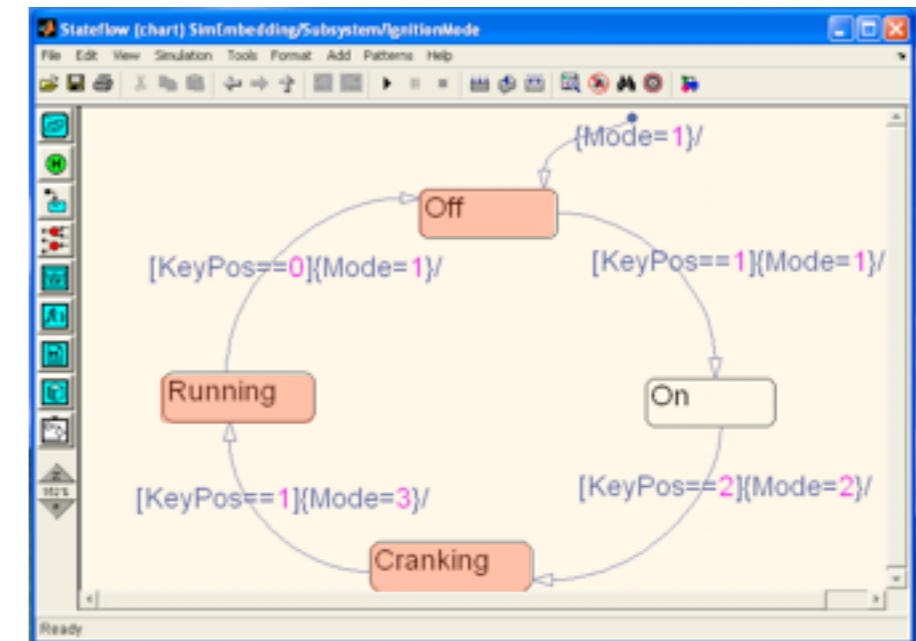
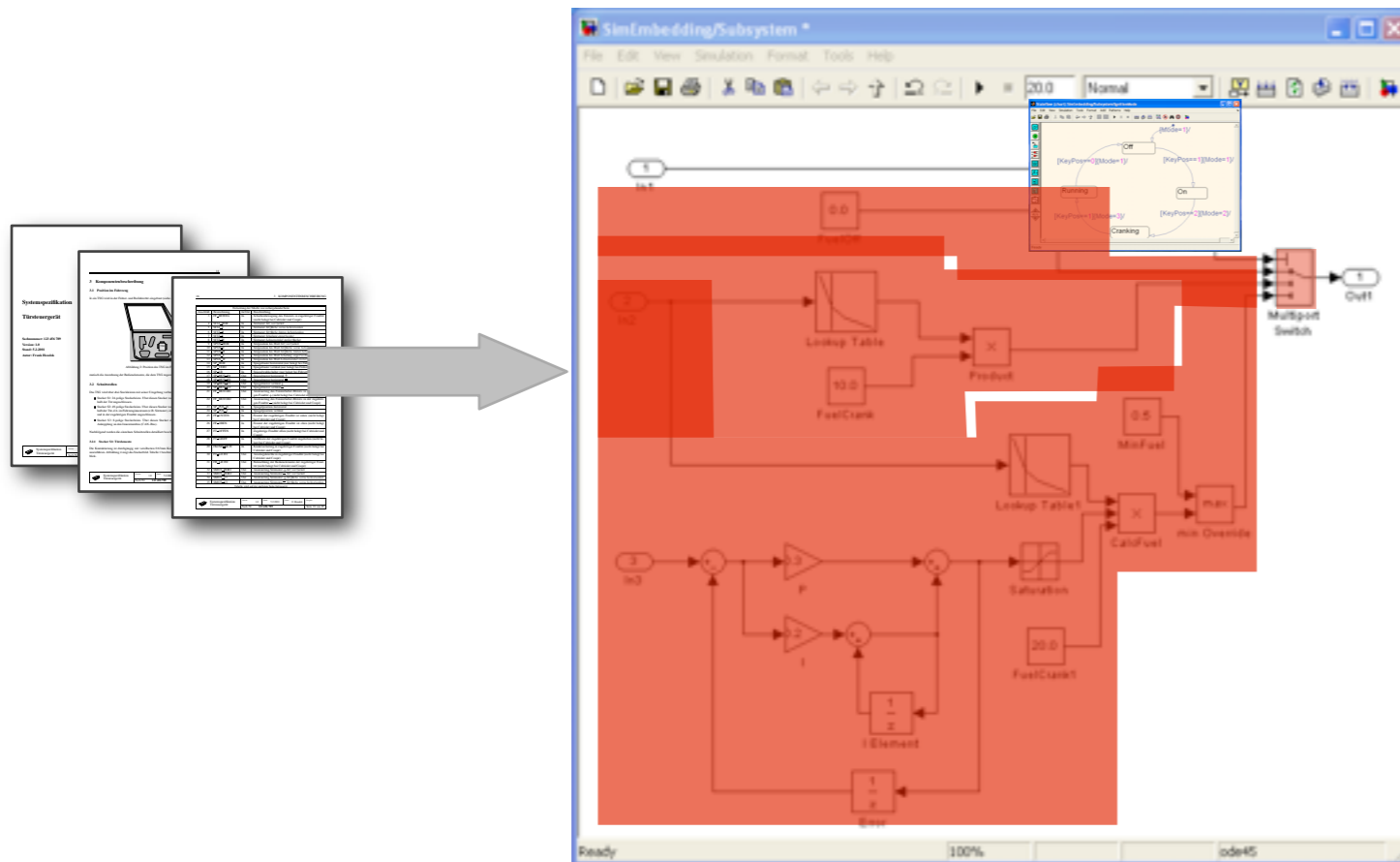
Automotive Software Development: Process



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

1. Example component specification: Lacking domain concepts
2. Example system specification: Lacking functional models
3. Example implementation: Lacking view integration
4. Example component design: Lacking consistency analysis
5. Example release planning: Lacking plan generation
6. Example verification: Lacking test generation
7. Example system maintenance: Lacking quality assurance
8. Example system integration: Lacking design space exploration

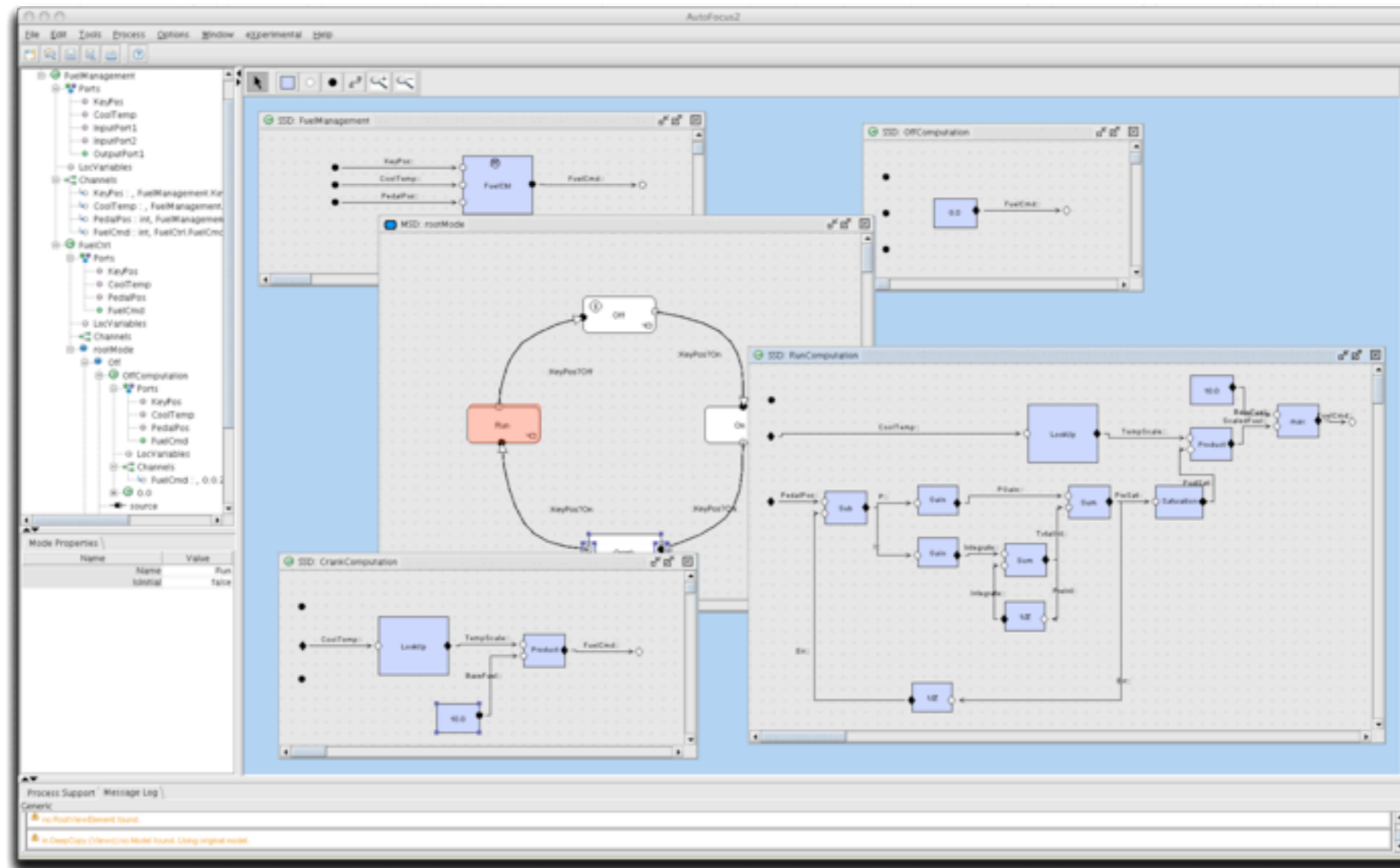
Example 1: Component specification



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

- Problem: Lacking concepts of problem domain
- Consequence: Inadequate models of functionality of application

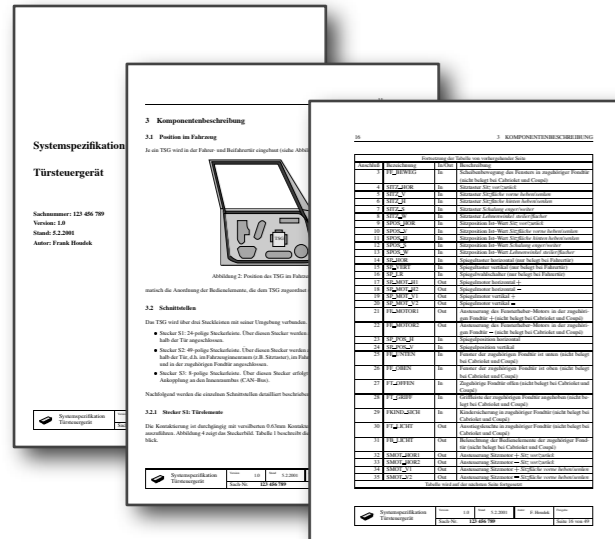
Domain-specific models: Modes of operation



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: Functional specification



40 6 FUNKTIONEN

Signal	Fahrzeugtyp	TSG	Reaktion
	Limousine Rechtslenker	Links	— ignorieren —
		Rechts	Bewegen der Scheibe hinten rechts mittels S2.FF_MOTOR1 und S2.FF_MOTOR2
S2.FFHB	Coupe, Cabriolet Rechtslenker	Beide	— ignorieren —
	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
S2.FFHB	Coupe, Cabriolet Linkslenker, Rechtslenker	Beide	— ignorieren —

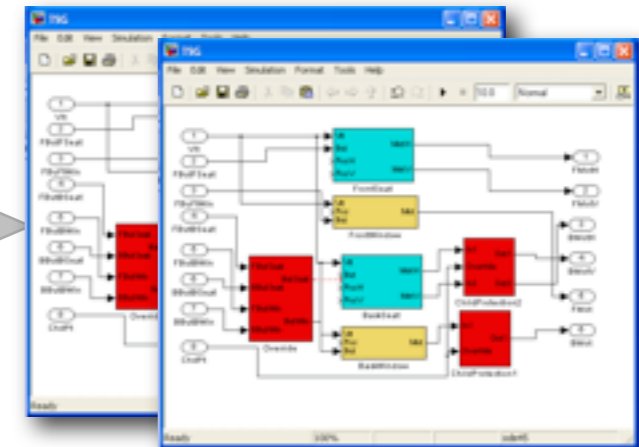
Tabelle 8: Reaktion auf Fenstersteuerbefehle.

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_LOW_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_x_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. FF_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. FF_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 befindet²; 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_x_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. FF_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. FF_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 Türsteuergerät	Version	1.0	Datum	5.2.2001	Autor	F. Houdek	Freigabe
	Sach-Nr.	123 456 789					Seite 40 von 49



State-of-art: Lacking models - No models of functional architecture

- Problem: Lacking models for usage view
- Consequence: No precise description of overall functionality

Models of early phases: Functions

The screenshot displays a software development environment with three overlapping windows. The top window shows a 'Resource Set' for 'BodyEletronics.systemmodel'. The middle window shows a tree view of the project structure, with 'Check Low Battery' and 'Stop Movement' functions highlighted in red. The bottom window shows the properties of the selected function, with a table of properties and values.

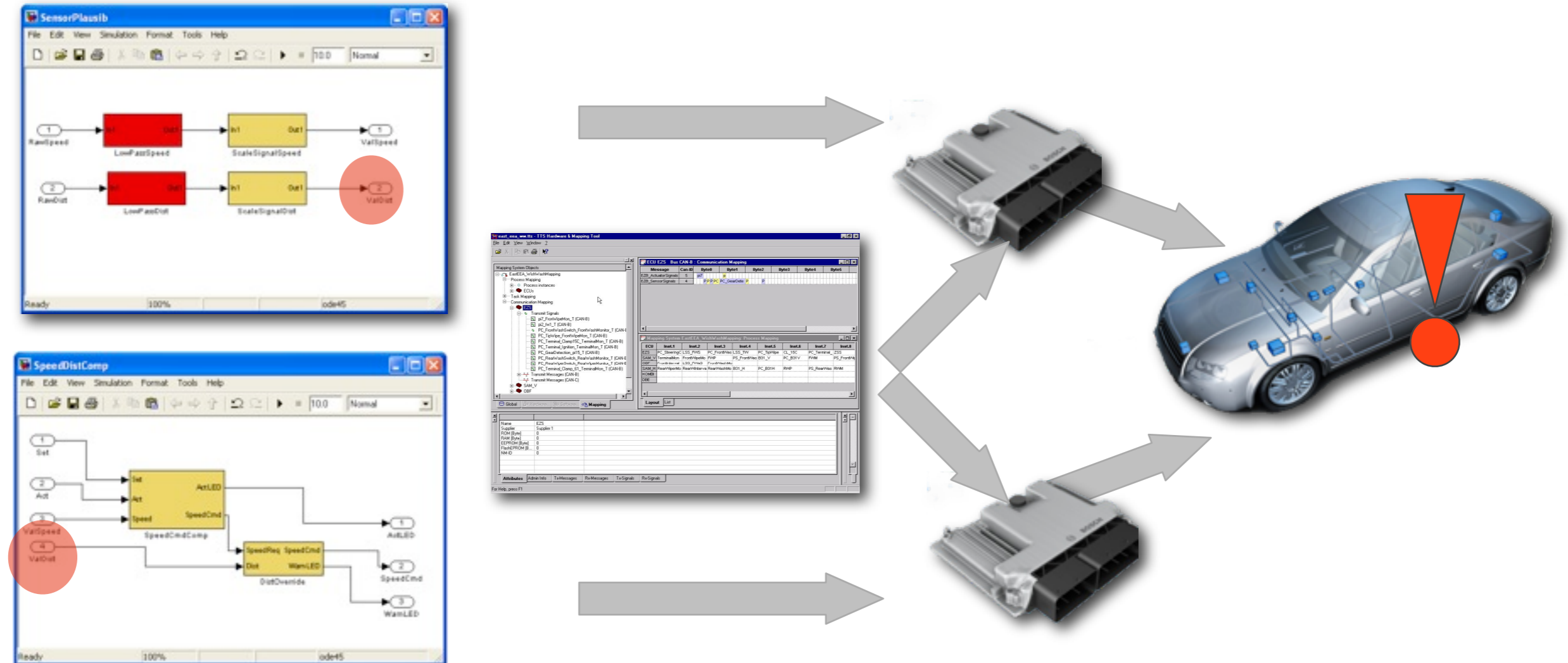
Property	Value
Comment	Check whether the current battery status is low
Name	Check Low Battery
Signal	Signal Bat
Value	Value Lo

Property	Value
Comment	Set the the motor to "No Movement"
Name	Stop Movement
Signal	Signal Mot
Value	Value Zr

Improvement: Explicit models - Structured instead of unstructured information

- Method: Extension of architectural modeling by functional level
- Further Examples: Product lines, timing requirements
- Improvement: More effective modeling and gezieltere Nutzung

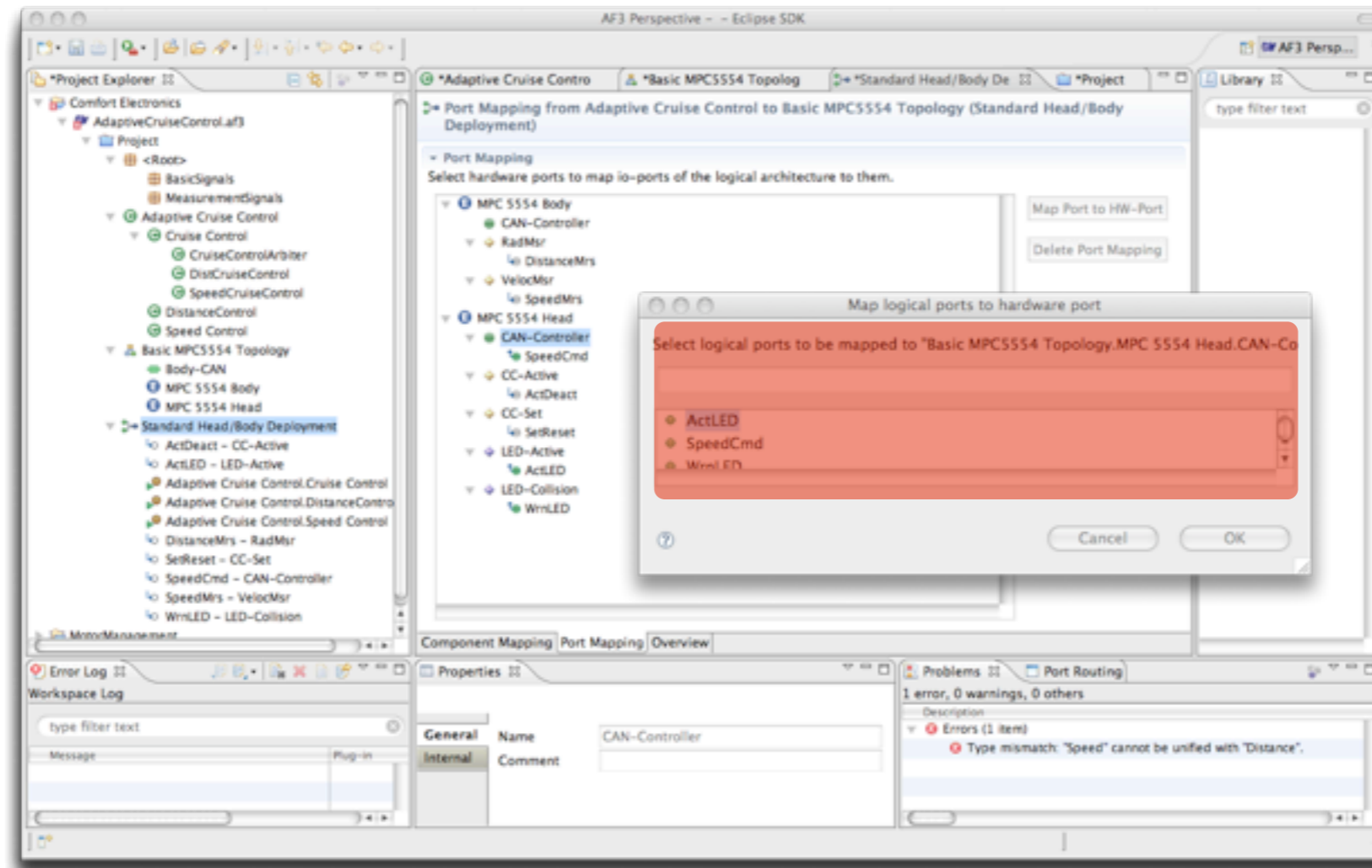
Example 3: Implementation



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

- Problem: Gaps in development process
- Consequence: Defects by contradicting decisions in later phases

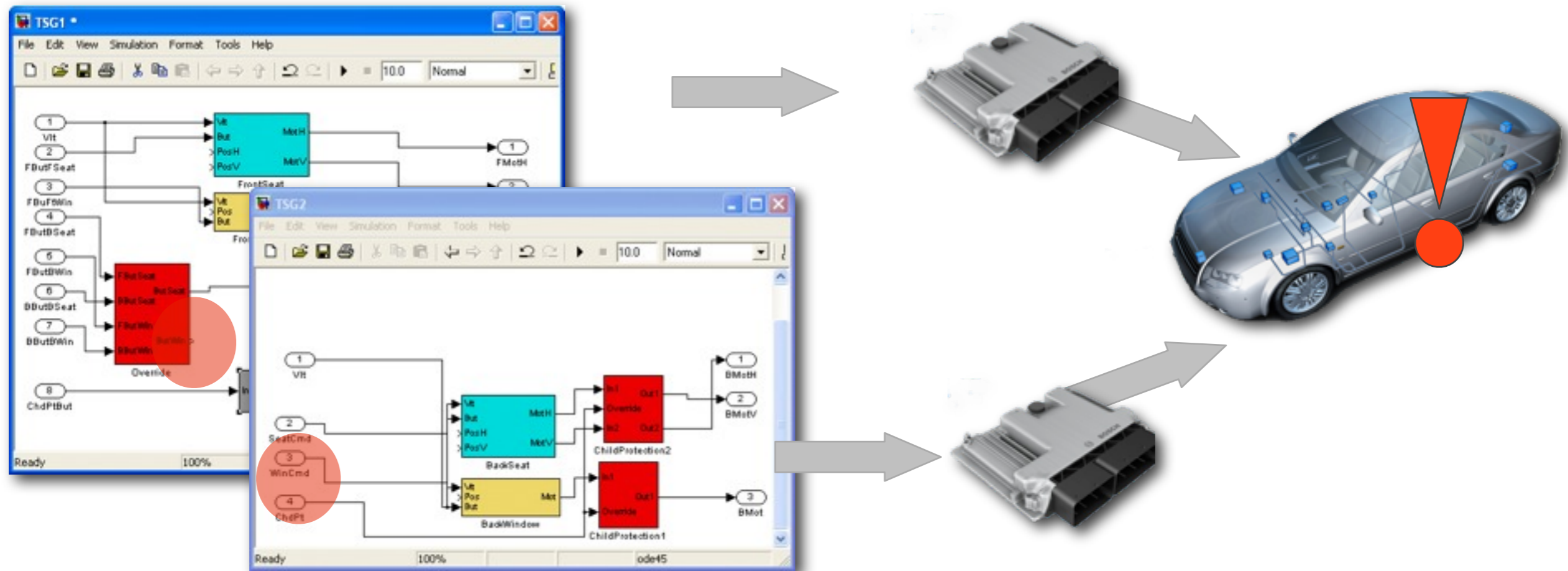
Integrated models: Component-topology-mapping



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: Checking of models

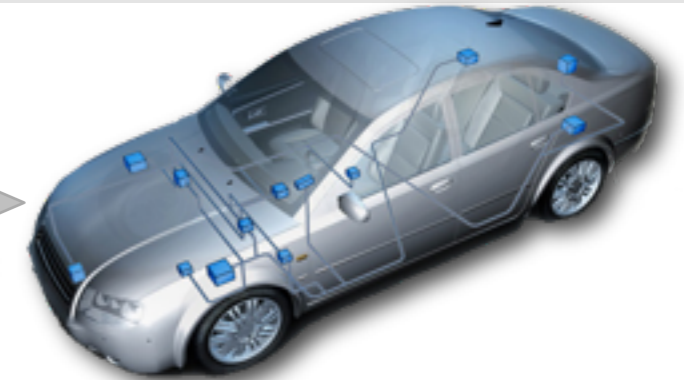
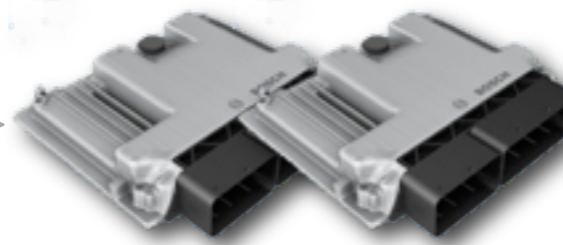
Modeling Guid

The screenshot displays the AutoFocus2 software interface. On the left, a 'Modeling Guid' box is partially visible. The main window shows a project tree for 'Tuersteuergerät' with components like 'TuerComfortfunktion', 'Sitzverstellung', 'Fensterheber', and 'Uebersteuerung'. A 'Counter Examples: Connected Ports' dialog is open, showing a definition: `forall p : Port . exists c : Channel . ((is InChannel(p,c)) or (is OutChannels(p,c)))` and a description: 'Each port is at least connected to one channel.' The dialog also lists counter examples, including `{ (p, BackWinCmd : Port) }` and `{ (p, BackWinCmd : Port) }` with `BackWinCmd : Port` as a sub-example. Other dialog boxes show condition names like 'Connected Ports' and condition descriptions like 'Each port is at least connected to one channel.'

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

- Method: Automatic checks for model structure/model content
- Further examples: Checking of signal periods, checking of determinism
- Improvement: Early defect elimination

Example 5: Release planing



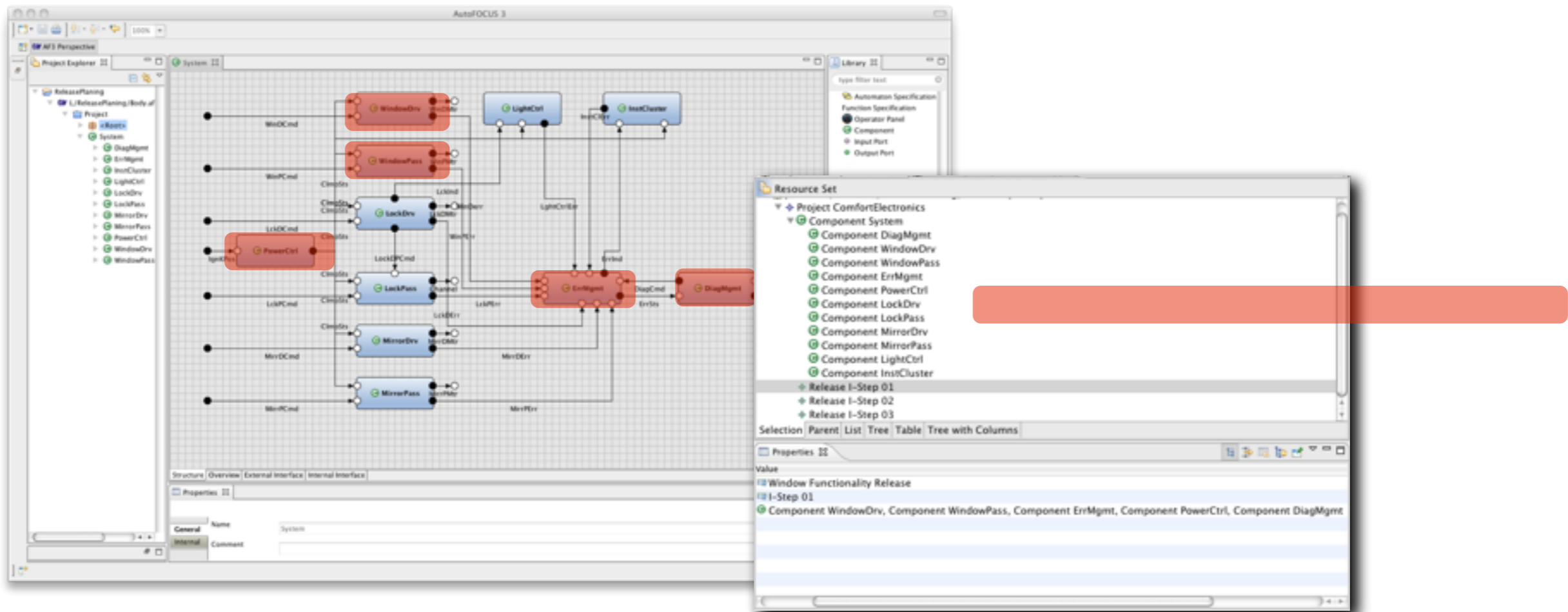
Funktion	Dependab.			Matrix			Integration Points			
	PowerCtrl	ErrMgmt	DiagMgmt	LockDrv	LockPass	LightCtrl	InstCluster	I-Step 02	I-Step 02	I-Step 03
PowerCtrl								x	x	x
ErrMgmt			X					X	X	X
DiagMgmt		X						x	x	x
WindowDrv	X							X	X	X
WindowPass	X					X		X	X	X
LockDrv	X					X			x	x
LockPass	X			X					X	X
MirrorDrv	X									X
MirrorPass	X									X
LightCtrl	X	X					X		x	x
InstCluster	X								x	x



State-of-art: Manual release planing - Construction of complete sub-architectures

- Problem: Lacking automatization of release construction
- Consequence: Complication of release planing

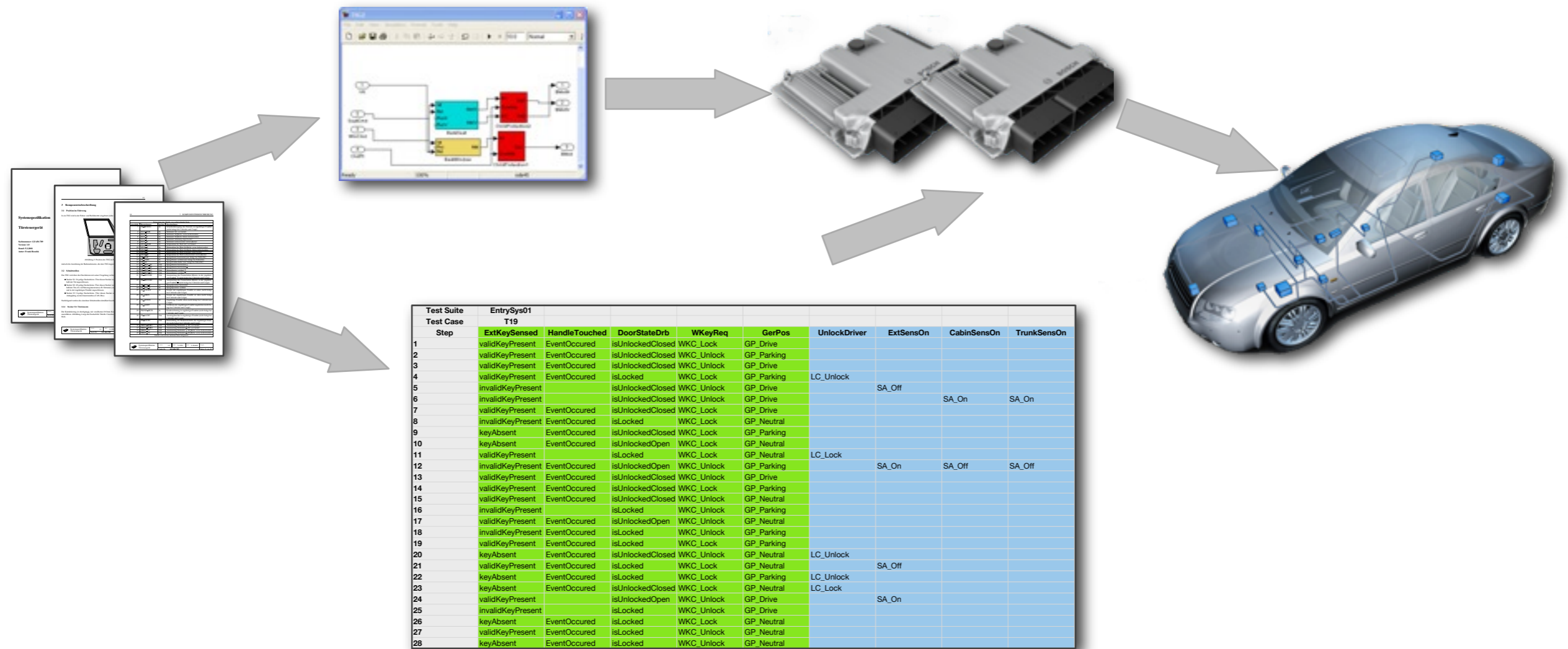
Synthesis method: Release plan generation



Improvement: Generation of release plans - Systematic incremental development

- Method: Automated model construction by generative search
- Further examples: Cable harness construction
- Improvement: Avoidance of miss-planing

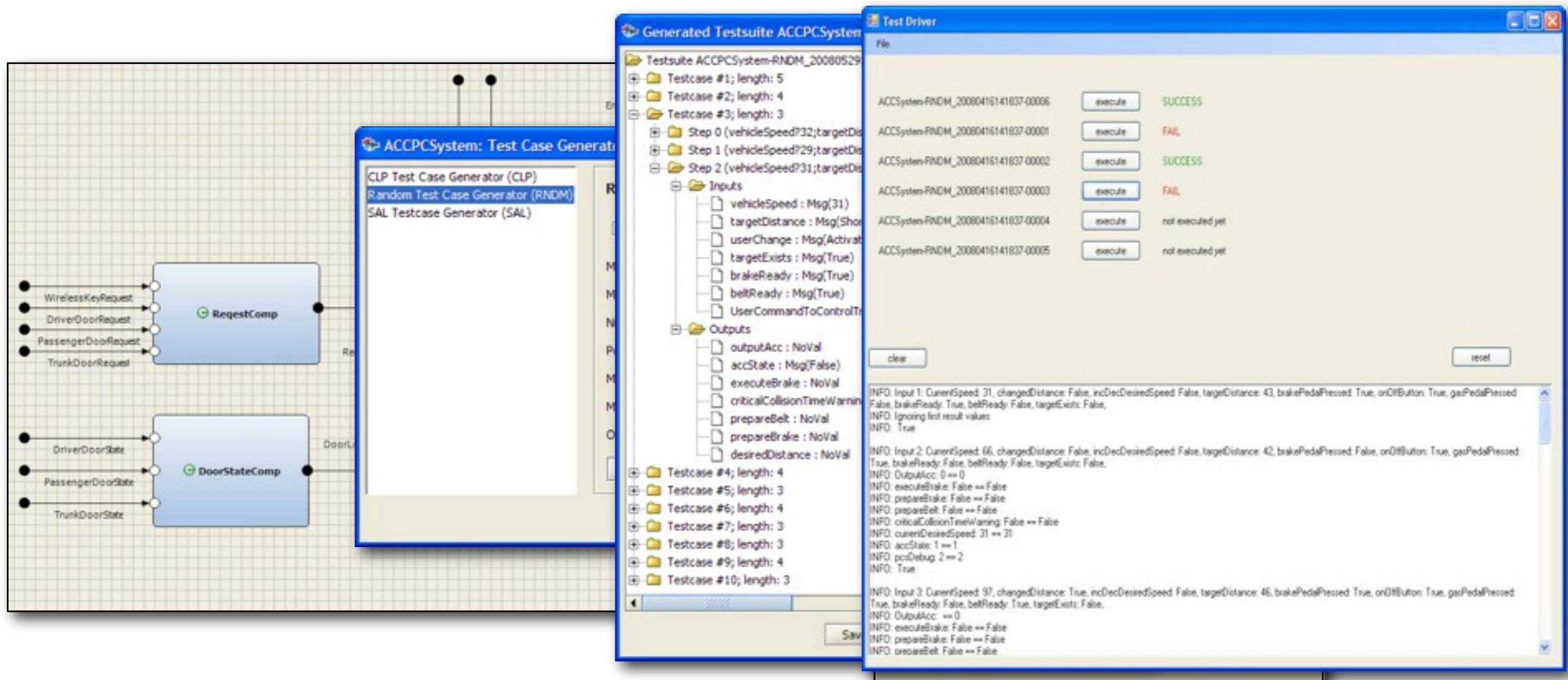
Example 6: Verification



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

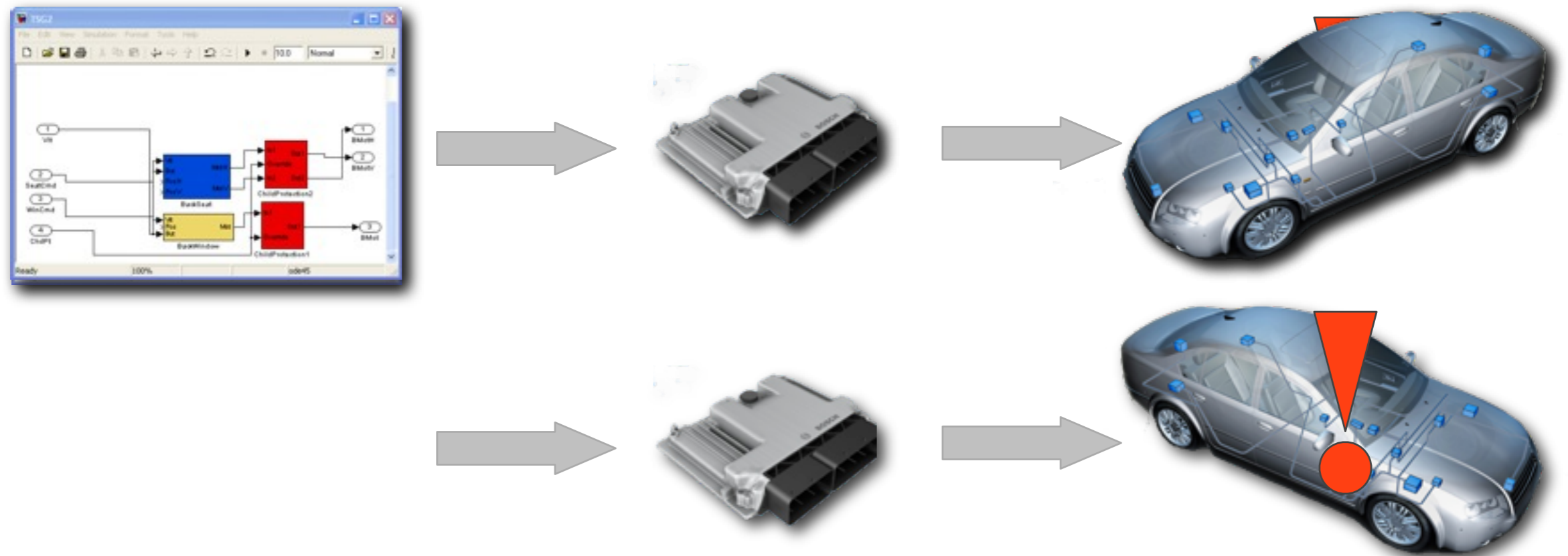
Synthesis method: Test case generation



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

Example 7: System maintenance



State-of-art: Unsystematic use of identical parts - Introduction of change defects

- Problem: Lacking analyse support for maintenance
- Consequence: Complicated and expensive maintenance

Analysis method: Clone detection

Clone table (Main)
Table giving all clones

Clone	ID	Name	Occurrences	Size	Weight	Details
Clone 126	202	140	2	385	70	additionsglied TCUStromglag
Clone 51	32	68	2	36	24	CCoeffizLUstV
Clone 18	54	66	2	27	23	motorstartstop
Clone 45	35	45	5	7	9	auffanggliedskurve
Clone 95	39	45	3	13	15	additionsglied TCUStromglag motorstartstop
Clone 9	42	42	2	25	25	motorstartstop
Clone 19	40	40	2	20	20	motorstartstop
Clone 36	40	40	4	18	18	auffanggliedskurve motorstartstop glauertAG
Clone 44	36	40	2	18	20	klammy
Clone 53	32	40	2	16	20	auffanggliedskurve
Clone 57	32	40	4	8	10	additionsglied CCoeffizLUstV TCUStromglag motorstartstop
Clone 62	32	40	2	16	20	auffanggliedskurve
Clone 20	36	36	2	18	18	auffanggliedskurve
Clone 42	36	36	4			auffanggliedskurve
Clone 54	36	36	2			
Clone 63	28	36	4			
Clone 72	36	36	4			
Clone 97	36	36	3			
Clone 130	48	36	2			
Clone 9	30	34	2			
Clone 49	42	34	2			
Clone 105	26	34	2			
Clone 16	33	33	3			

Table 1: Klon 1

Name	Wert
Gravanz	400
Gravanz	400
Intervall	3
Volumen	144
Gravanz Sum	104
Beschreibung	bremsenregler, tempomatregler

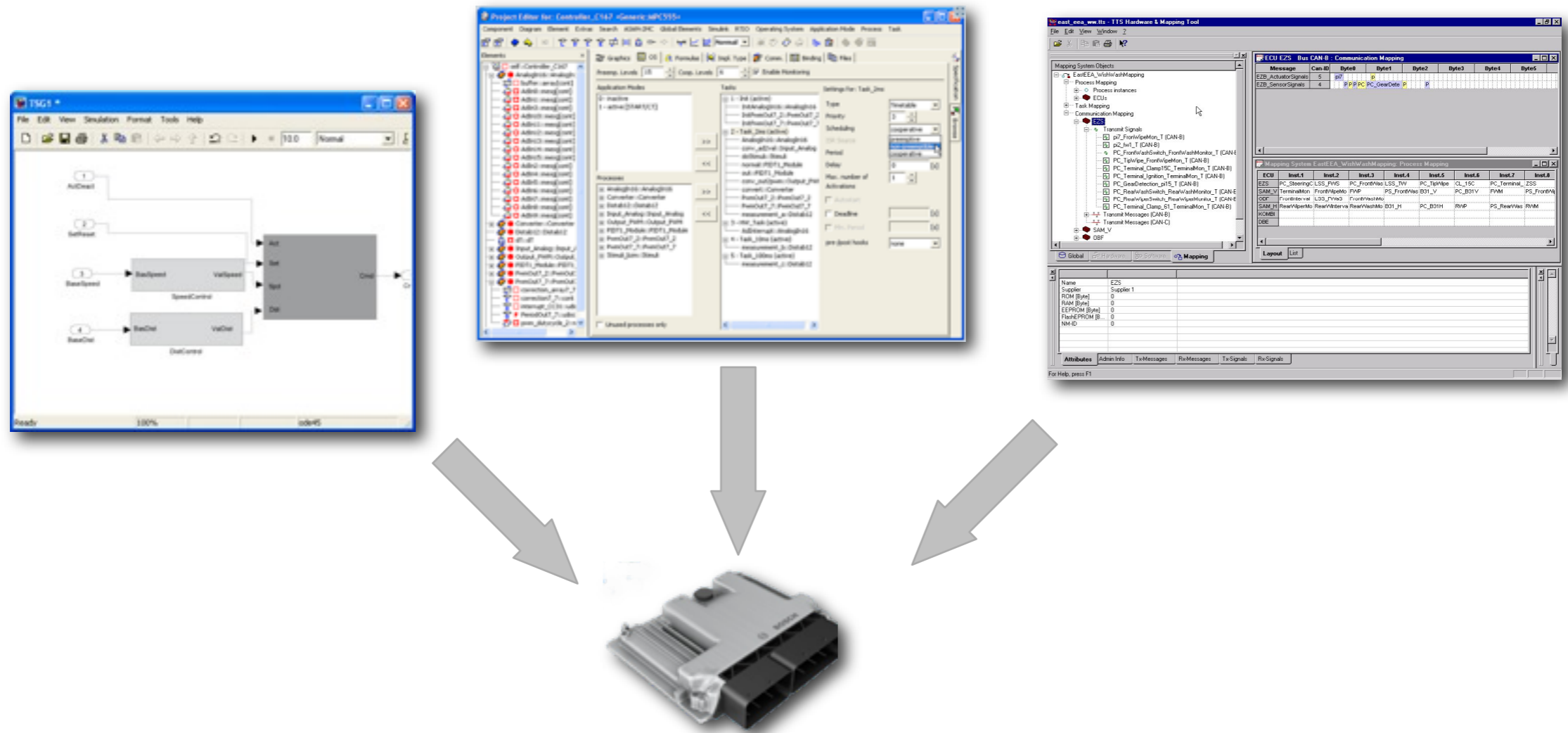
Table 2: Klon 18

Name	Wert
Gravanz	74
Gravanz	74
Intervall	3
Gravanz Sum	148
Beschreibung	tempomatregler

Improvement: Identification of identical parts - Avoiding change anomalies

- Method: Automated checks of architectural structure
- Further examples: Requirements clones, IP-libraries, product lines
- Improvement: Simplification maintenance

Example 8: System integration



State-of-art: Manual integration steps - Constructed models

- Problem: Lacking extensive automatization of design steps
- Consequence: Limited support for design space exploration

Synthesis method: Schedule generation

The image displays two software interfaces. On the left is the Eclipse IDE showing a project named 'AdaptiveCruiseControl.af3'. The project structure includes components like 'CruiseControlArbiter', 'DistanceControl', and 'SpeedControl'. A diagram in the center shows a CAN bus topology with components like 'CAN-MPC554 Head', 'CAN-MPC554 Body', and 'CAN-Controller'. A red box highlights a set of equations:

$$\begin{aligned} E_{DstCtrl} &= S_{DstCtrl} + 10, \dots \\ E_{ValSpdMsr} &= S_{ValSpdMsr} + 5 \\ E_{DstCtrl} &\leq S_{ValSpdMsr}, E_{ValSpdMsr} \\ E_{DstCtrl} &\leq S_{SpdCtrl} \vee E_{SpdCtrl} \\ E_{ValSpdMsr} &\leq S_{ValDstMsr} \vee E_{ValDstMsr} \end{aligned}$$

On the right is the 'TTS Hardware & Mapping Tool' window. It shows a 'Mapping System Objects' tree with various CAN bus components. Below it is a table titled 'ECU EZS Bus CAN-B : Communication Mapping'.

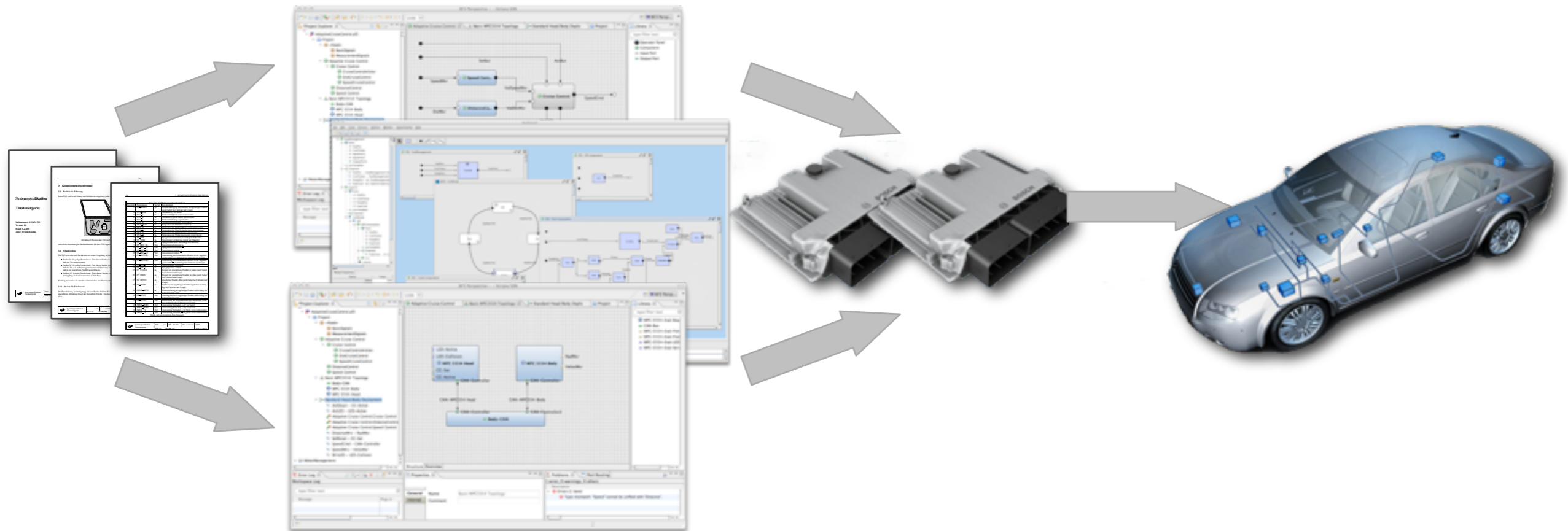
Message	Can-ID	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5
EZB_ActuatorSignals	5	p17	p	p	p	p	p
EZB_SensorSignals	4	p	p	p	p	p	p

Below the table is a 'Process Mapping' section with columns for ECU, Inst.1 through Inst.8, and various signal names like 'PC_Steering', 'PC_FrontWash', etc.

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

Models in the development process: Principles



Improvement of effectivity and efficiency: Models support

1. Adequate modeling
2. Phase-integrating development
3. Front loading analyse methods
4. Synthesis of models

➔ **Implementation of tools for an integrated development process**

Models in application: Examples



Test via application development:

- Driver-Assistance: Adaptive cruise control, parking assistant
- Infotainment: Use of entertainment system
- Body electronics: Power window, keyless entry system