

Modellbasierte Entwicklung eines sicherheitskritischen (SIL-4) Projekts unter Verwendung eines zertifizierten Codegenerators

Dr. Thomas Liedtke, Bernd Holzmüller

ICS AG

Vortrag beim 8. SafeTRANS Industrial Day am 5. Mai 2010

Agenda



ICS AG - Kurzvorstellung

Modellbasierte Entwicklung mit SCADE

Entwicklung eines digitalen, sicheren Übertragungssystems mit SCADE

Aufwandsbetrachtungen

Lessons Learned - Zusammenfassung



Philosophie

Partner für den gesamten Produkt Life Cycle

Vordenken

- Machbarkeitsstudien, Evaluierungen

Beraten

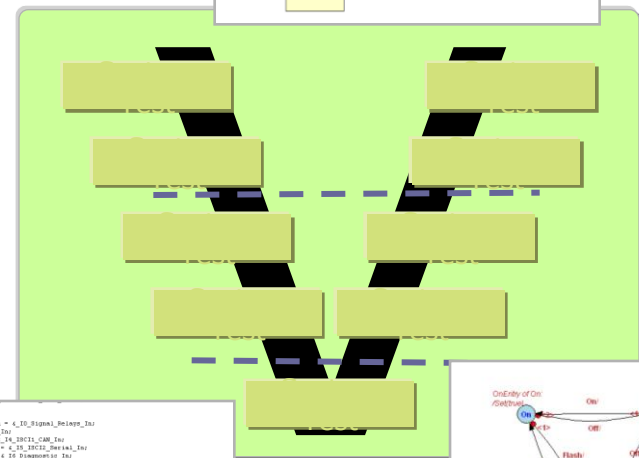
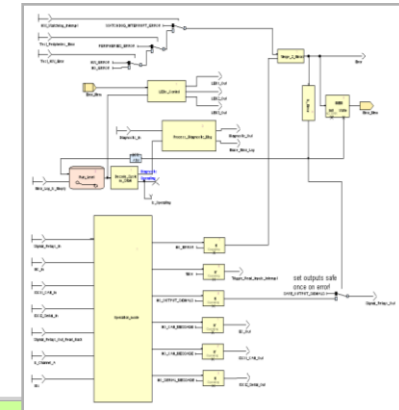
- Technologien
- Methoden
- Werkzeuge
- Standards
- Prozesse

Umsetzen

- Systems und Safety Engineering
- Produkt Design und Entwicklung
- Kapazitäts- und Kompetenzergänzung
- Test und Validierung
- RAM und Safety Management
- Assessment

Betreuen

- Maintenance von Software und Systemen



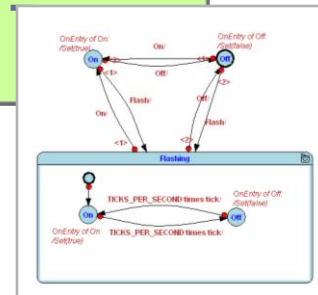
```

/* Init channelContext */
channelContext_10_Signal_Relay_In = 4_10_Signal_Relay_In;
channelContext_10_IC_10 = 4_10_IC_10;
channelContext_10_IC11_CAM_In = 4_10_IC11_CAM_In;
channelContext_10_IC12_Period_In = 4_10_IC12_Period_In;
channelContext_10_Singapore_In = 4_10_Singapore_In;
channelContext_10_Signal_Relay_Out_Back = 4_10_Signal_Relay_Out_Back;
channelContext_11_10 = 1_11_10;

/* Initiate SCADNode, not global channelContext */
Channel_10Init (channelContext 1);

/* Make some basic tests for the SCADNode */
/* Check, whether Channel A or Channel B */
/* ID: In Channel A ChannelID: In = 0 */
#E ID_CHANNEL_A == g_uChannelID;
{
  channelContext_10_In_Channel_A = true; /* Binary: 01 -> Channel A */
  channelContext_11_10=RepeatID = 0x1;
}
else
{
  channelContext_10_In_Channel_A = false;
  channelContext_11_10=RepeatID = 0x2; /* Binary: 10 -> Channel B */
}

/* 11_10 */
channelContext_11_In=RepeatID = (Repeat) g_uRepeatID;
channelContext_11_10=RepeatID = (Repeat) g_uRepeatID;
  
```



Standorte



Geschäftsstelle Braunschweig

Salzdahlumer Straße 196
38126 Braunschweig
Tel.: +49 531 26306 - 33
Fax: +49 531 28503 - 46

Hauptsitz Stuttgart

Sonnenbergstraße 13
70184 Stuttgart
Tel.: +49 711 21037 - 00
Fax: +49 711 21037 - 53

Geschäftsstelle Immenstaad

Ziegelei 5
88090 Immenstad
Tel.: +49 7545 94996 - 0
Fax: +49 7545 94996 - 29

Geschäftsstelle Ulm

Sedanstraße 14
89077 Ulm
Tel.: +49 731 93579 - 325
Fax: +49 731 93579 - 327

Geschäftsstelle Berlin

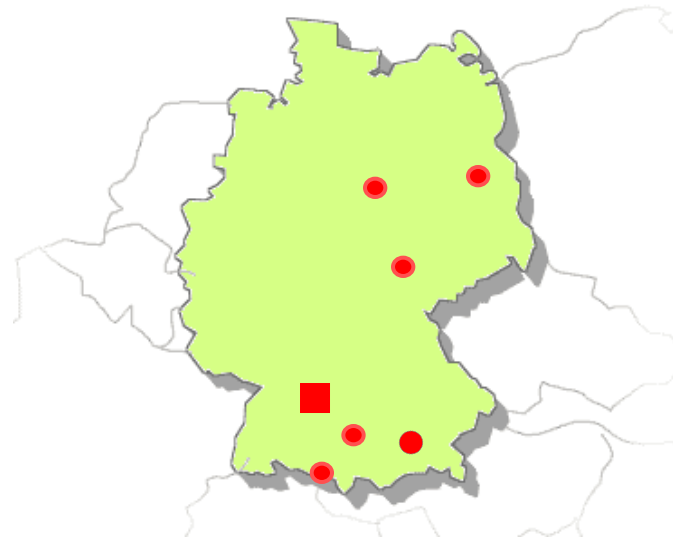
Rother Straße 18
10245 Berlin
Tel.: +49 30 20687 - 262
Fax: +49 30 20687 - 375

Geschäftsstelle Leipzig

Marschnerstraße 3
04109 Leipzig
Tel.: +49 341 98250 - 10
Fax: +49 341 98250 - 11

Geschäftsstelle München

Lise-Meitner-Straße 1
38126 Unterschleißheim
Tel.: +49 89 203504 - 70
Fax: +49 89 203504 - 69



Competence Centers

Systems Engineering
Software Engineering



Operation Support Systems

- Bodenstationen
- Betriebs-Überwachungssysteme
- Simulatorsysteme
- Führungs- und Führungseinsatzsysteme
- digitale Kartensysteme
- Auswertesysteme
- Flugsicherungssysteme

Onboard Systems

- Missionsplanungs- und Managementsysteme
- Aufklärungssysteme
- Selbstschutzsysteme
- Steuerungssysteme
- Radarsysteme

Competence Centers

Fertigungslogistik
Infrastruktur
ERP
QM Industrielle Prozesse



Supply-Chain-Management

- Beschaffungslogistik
- Produktionslogistik
- Lagerlogistik
- Containermanagement
- Zollabwicklung

IT-Infrastruktur

- Maintenance (Betriebsbereitschaft 7*24)
- mobile Datenfunksysteme

ERP-Systeme

- Entwicklung und Integration von branchen- und kundenspezifischen Lösungen
- Definition und Implementierung von Systemschnittstellen

Transportation



Competence Centers

Systems Engineering
Software Development
Verification & Test
Validation
RAMS
Quality Assurance
Assessment

Applikationen:

- Leit- und Sicherungstechnik
- Schienenfahrzeugtechnik

Engineering

- Systems & Safety Engineering
- Software Design und Development

Verification & Validation

- Analytische Verifikation
- Testmanagement und -processing
- Testautomatisierung, Test-Tools
- Soft- und Hardware-Validierung



RAM und Safety Management

- RAMS und funktionale Sicherheit
- System- und Fehleranalysen (FMEA, FTA, HAZOP)
- Sicherheitsnachweise, Cross Acceptance
- Kunden- und Realisierungsprojekte (Customer Order Fulfillment)

Assessment

- Systembegutachtung
- Beratung und Unterstützung bei Zulassungsfragen
- Schnittstelle zur Zulassungsbehörde

Methods-Processes-Tools (MPT)

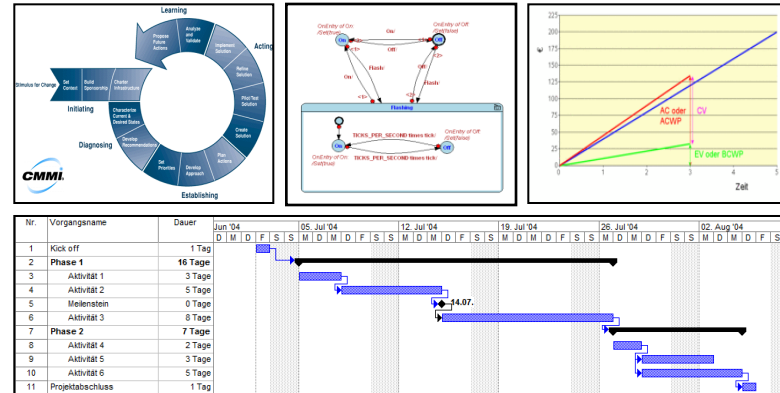


Werkzeuge und Methoden:

- Tool-Evaluierung (Anwendbarkeit, Performance)
- Software Engineering (z. B. Modellbasierte Entwicklung)
- Statische Codeanalysen
- Simulation & Test (z.B. anforderungsbasiertes Testen, Testautomatisierung)

Prozesse:

- Prozessentwicklung von Teilprozess- und Gesamtprozess für softwareintensive und sicherheitskritische Systeme (IEC 61508, EN 5012x, CD 26262, DO-178B)
- Prozesseinführung und -optimierung
- Schwachstellenanalyse nach SPICE, CMMI, ISO 9001
- Entwicklung von Aktionsplänen



Projektmanagement:

- Projektplanung, -steuerung und -fortschrittskontrolle
- Aufwandsabschätzungen
- Risikomanagement
- Auswertungen und Reports
- Tooleinsatz wie MS-Project, Primavera
- Coaching der Stakeholder

Agenda



ICS AG - Kurzvorstellung

Modellbasierte Entwicklung mit SCADE

Entwicklung eines digitalen, sicheren Übertragungssystems mit SCADE

Aufwandsbetrachtungen

Lessons Learned - Zusammenfassung



- 2004 **Gründung der Gruppe Methoden Prozesse Tools (MPT)**
Idee zur Bündelung der Consulting Aktivitäten innerhalb einer eigenständigen Einheit

- 2005 **erste Gehversuche mit SCADE 4**

- 2005 **erstes Kundenpilotprojekt**
Focus auf Erprobung von SCADE im HW-nahen Bereich der Mikrocontroller

- 2006 - **erstes kommerzielles Projekt im Bahnumfeld mit SCADE 5**

- 2007

- 2008 **Einrichtung Business Unit MPT**

- 2009 Machbarkeitsstudie im Transportation-Umfeld mit SCADE 6

- 2010 Einsatz von SCADE in **Luftfahrt**projekten

SCADE Suite

- Kommerzielles modellbasiertes Werkzeug
 - SCADE = SAGA (Verilog) + SAO (Airbus) (Ende 80er Jahre)
 - ⇒ s. <http://www.esterel-technologies.com/company/history/>
 - Seit 2001 durch die frz. Fa. *Esterel Technologies* von Telelogic übernommen, weiterentwickelt und vertrieben
 - Einsatz zunächst hauptsächlich Luftfahrtindustrie, dann auch Nukleartechnik u.a.

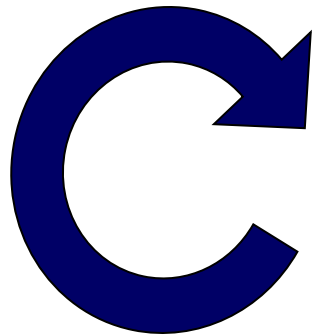


- Basisnotation: synchrone Datenflusssprache *Lustre*

- formal definiert
- streng typisiert
- Logischer Zeitbegriff ("tick")
- Variablen = Sequenzen von Werten (Funktion der Zeit)
- Datenflussgleichungen definieren Abhängigkeiten zwischen Variablen (auch rekursiv)
 - nat: int;**
 - nat = 1 -> (pre(nat) + 1);**
- deterministische Semantik

```
node Average (I : int) : (A : int);
let
  N = 1 -> pre(N) + 1
  A = (I -> (pre(A) + I)) / N
tel;
```

- Zyklisches Ausführungsmodell



```
loop  
  Wait clock/event  
  Read sensors  
  Compute outputs  
  Write actuators  
end loop
```

SCADE Suite (1)



The screenshot displays the SCADE Suite software interface for a project named "BUe.vsw - SCADE Suite - [BUe/eq_BUe_1]". The main workspace shows a block diagram of a vehicle control system. The central block is "Fahrzeugsteuerung" (Vehicle Control), which is connected to several other components:

- Funkstrecke** (Radio Channels): Three blocks labeled "Funkstrecke 2" with instance numbers 8, 7, and 9. They are connected to "Fahrzeugsteuerung" via signals "Einschaltbefehl", "Statusabfrage", and "Ausschaltbefehl".
- Fahrzeugsensor** (Vehicle Sensor): A block with instance number 1, connected to "Fahrzeugsteuerung" via signals "kmStand" and "Stoer_FzSensor_defekt".
- FzSensorDefekt** (Sensor Defect): A block connected to "Fahrzeugsensor" via signals "BUe_frei" and "FzSensorDefekt".
- FBY** (Function Block): A block connected to "Funkstrecke 8" and "Funkstrecke 9".
- Inputs/Outputs:** "Bestaet_mSich" is an input to "Fahrzeugsteuerung". "Anz_mSich" is an output from "Fahrzeugsteuerung".

The left sidebar shows a project tree with folders like "Locals", "Einschalten", "Main", "BSA_Lz", "BSA_Mindestgruenzeit", "BSA_Schranken", "BUe", "BZ", "Countdown", "Fahrzeugsensor", "Fahrzeugsteuerung", "Funkstrecke", "HSchranke", "Lz_Anlage", "Prove", "Timer", "Zug_Bremskurve", and "Zug_kann_noch_halten".

The bottom status bar shows the following messages:

```

x Loading project BUe.etp...
^ Successfully loaded project BUe.etp

```

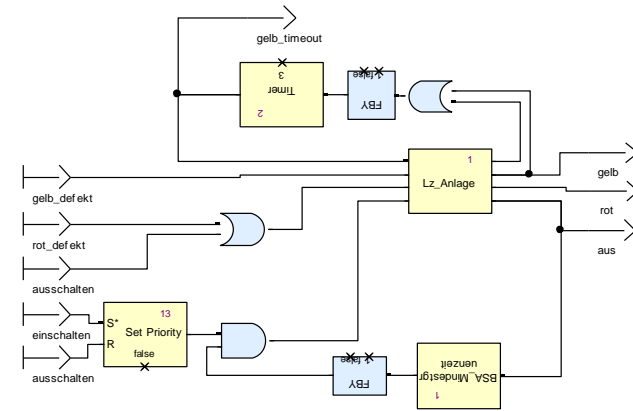
The bottom navigation bar includes "Messages", "MTC", "Dump", "Build", and "Simulator".

SCADE Suite (2)



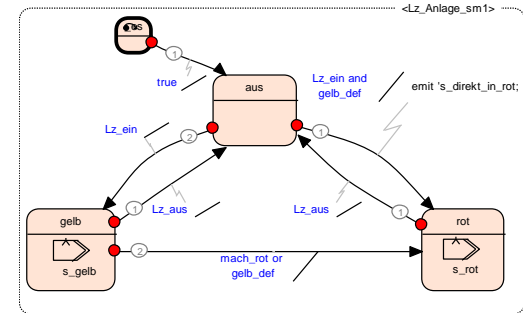
Grafischer Editor

- Datenmodellierung (Strukturen, Arrays, Aufzählungstypen)
- Datenfluss- (Block-) Diagramme
- *Safe State Machines* (Harel State Machines mit Einschränkungen)
- Seit Version 6 sind beide grafische Notationen homogen integriert



Codegenerator

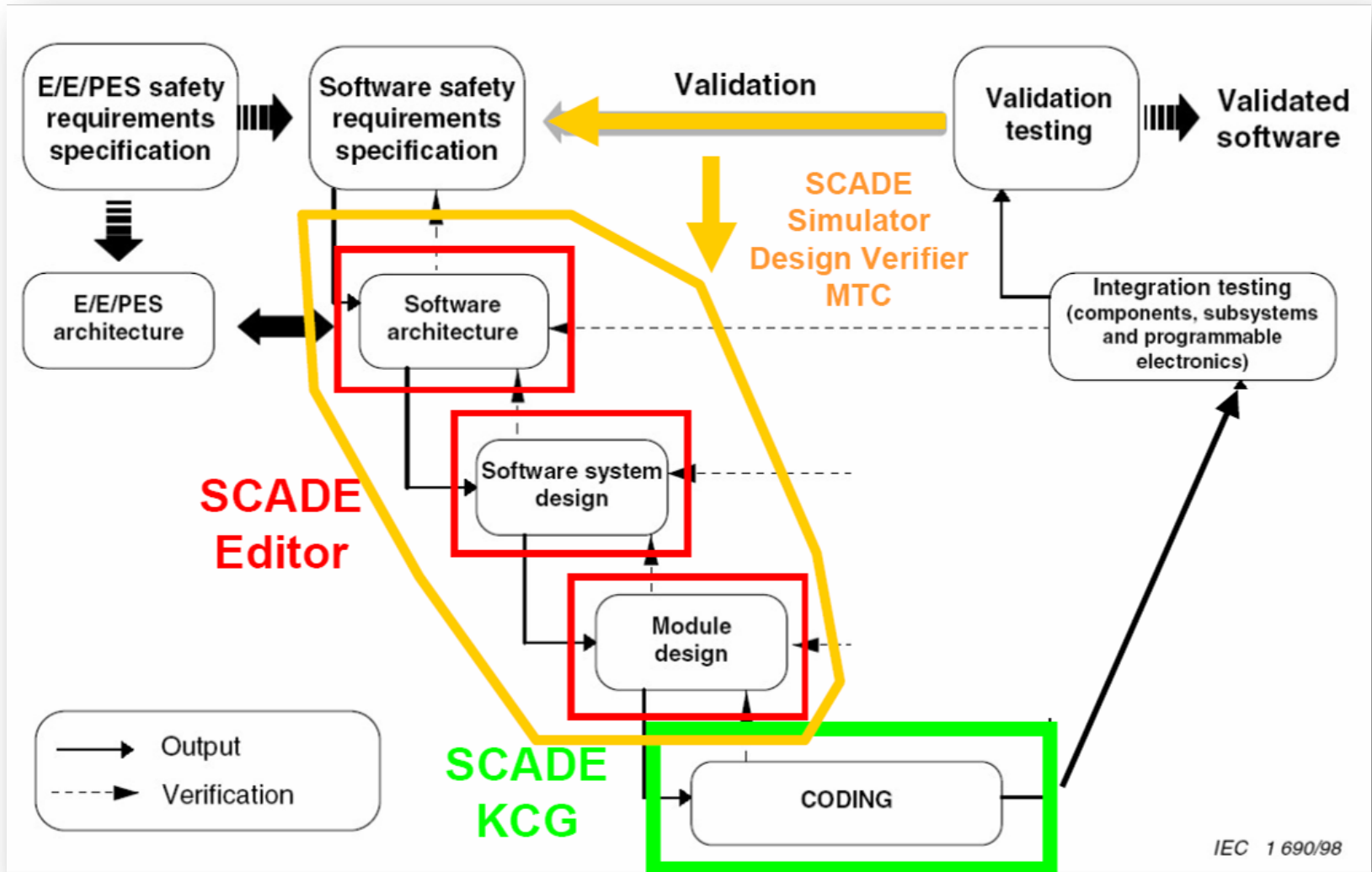
- erzeugt aus SCADE-Modell effizienten und portablen C-Code
- erzeugte C-Funktionen sind flexibel einsetzbar (mit / ohne Betriebssystem, zeit-/ereignisgetrieben)
- qualifiziert / zertifiziert gemäß DO-178B / IEC 61508 / EN50128 / IEC 60880, jeweils höchste Sicherheitsstufe



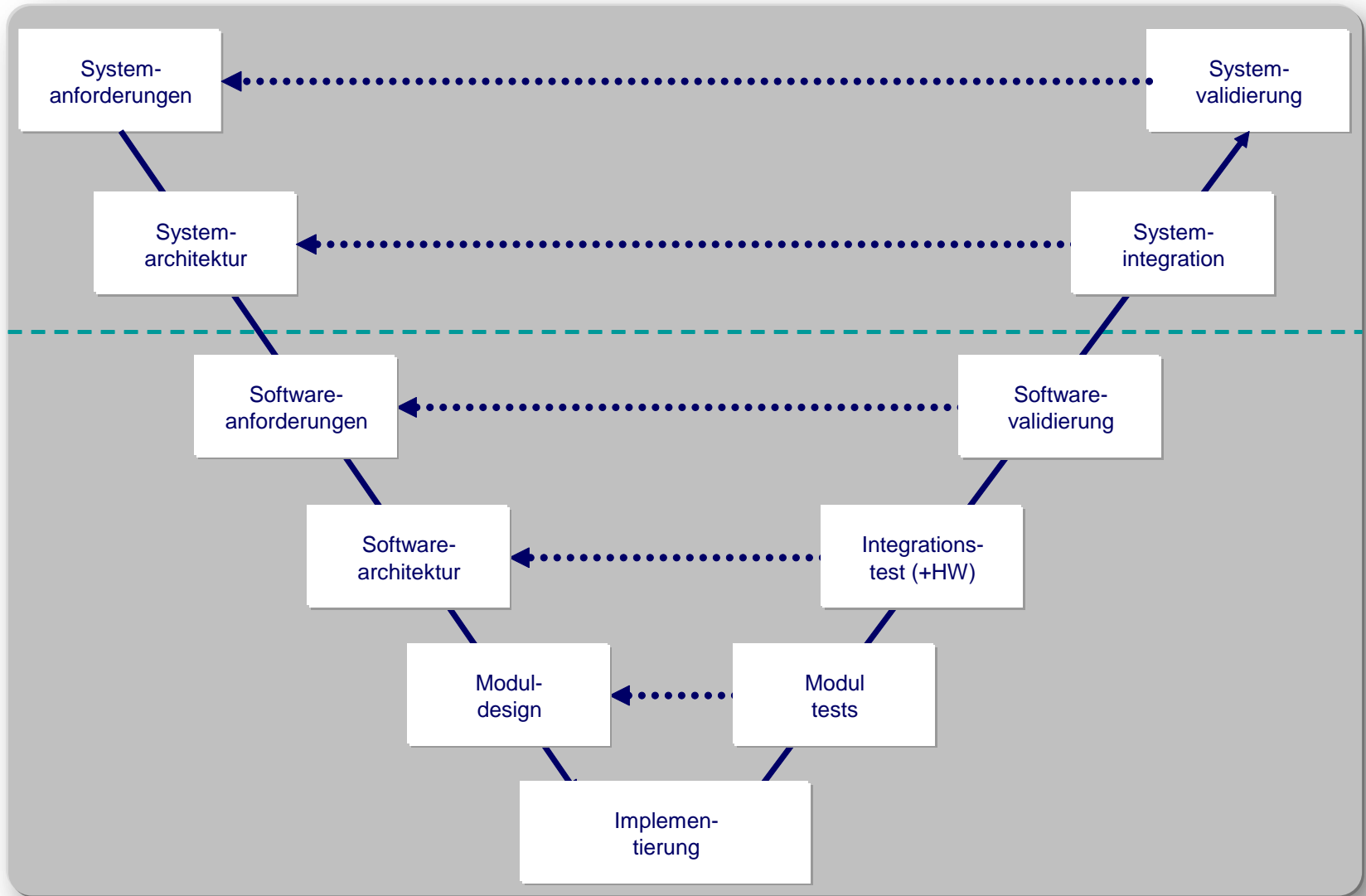
SCADE Suite (3)

- Modellsimulation
- Model Test Coverage (MTC)
- Design Verifier
 - Formaler Nachweis von (Sicherheits-) Aussagen im Modell
 - Plugin der Fa. *Prover Technologies*
- Report-Generator
- Statische WCET-Analyse
 - Plugin der Fa. *AbsInt*
- Diverse weitere Anbindungen
 - DOORS
 - Reqtify
 - Matlab / Simulink
 - Rhapsody
 - Artisan

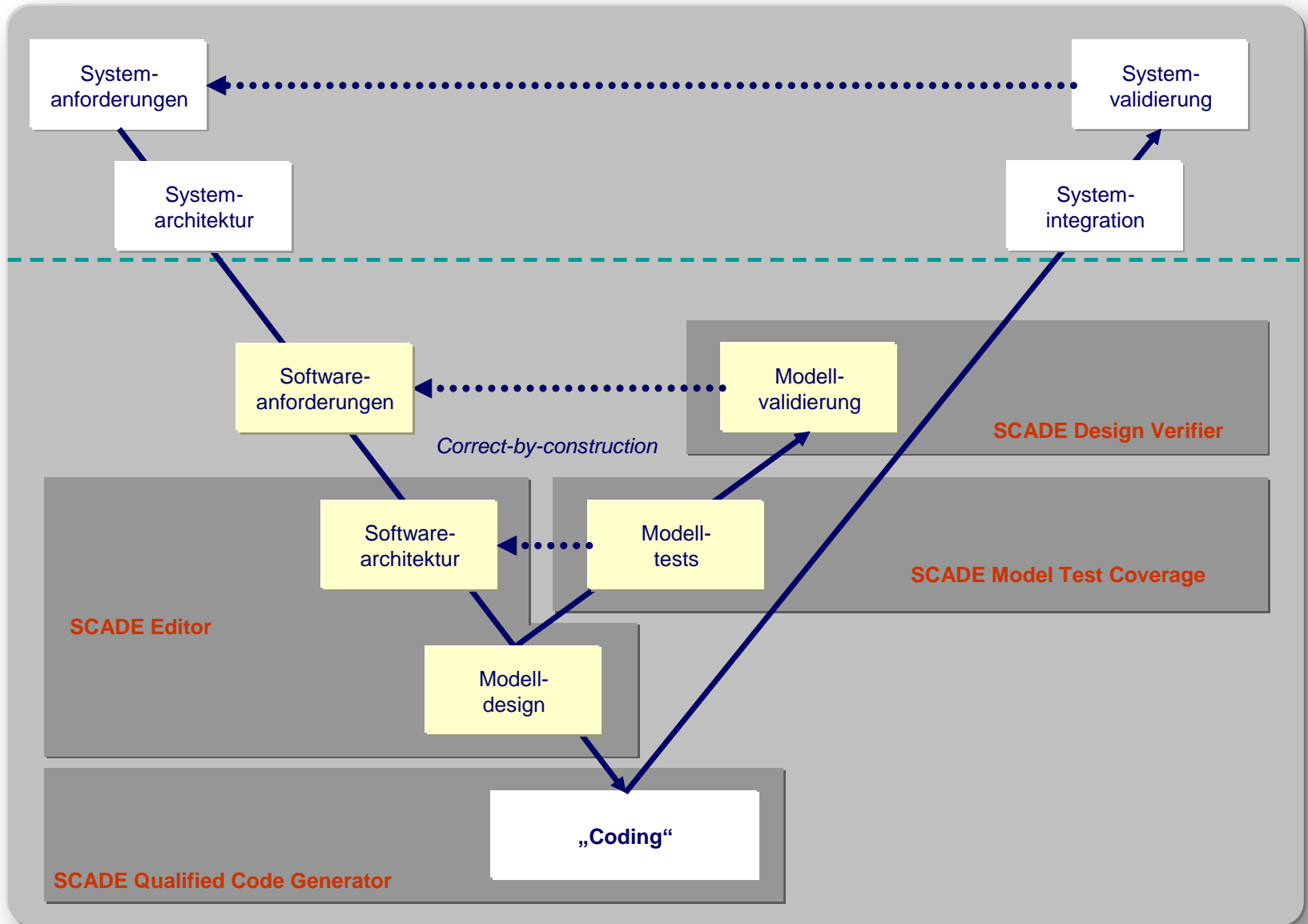
Entwicklungsprozess mit SCADE



V-Modell nach EN 50128



V-Modell nach EN 50128 mit SCADE



Einsparpotenziale bei Verwendung von SCADE (1)



- Einsparungen bei Verifikationsaktivitäten
 - verschiedene **Abstraktionsebenen** (Designstufen, Code) **stimmen stets überein** (Schnittstellenkonformität durch Werkzeug erzwungen)
 - ⇒ vor allem auch nach Änderungen!
 - Design enthält in sich **keine Widersprüche** (vgl. Design mit UML oder Text!)
 - ⇒ Konsequenz der formalen Notation (eindeutige Semantik)
 - **frühe Verifizierung** durch Simulation ("Entwicklertests") und Design Verifier
 - Einsatz des **Design Verifiers** (z.B. für Sicherheitseigenschaften) erspart u.U. erhebliche Testaufwände
 - ⇒ soweit durch vorgegebene Normen unterstützt
 - **Design, Designdokumentation** und **Code** stimmen stets überein
 - **Keine Modultests** nötig ("low level testing activities")
 - ⇒ SCADE-Modelltests setzen auf dem gesamten Modell auf (Stimulation des Modells durch Eingaben in den Hauptknoten)
 - ⇒ Testen von Knoten der unter(st)en Ebene nur als Entwicklertests
 - ⇒ Kein aufwändiges Stubbing nötig (SCADE-Knoten viel leichter isolierbar und einzeln testbar)

Einsparpotenziale bei Verwendung von SCADE (2)



- Einsparungen bei Dokumentation
 - Weitgehende **Generierung von Designdokumenten** aus annotiertem Designmodell
 - Einfacher **Nachweis der Traceability** (Design ↔ Code)
- Einsparungen bei Codierung von nebenläufigem Verhalten
 - Keine Aufwände für Synchronisierung
 - Keine Probleme mit „race conditions“, Nichtdeterminismus, Verhungern von Threads
 - Deterministisches Verhalten erleichtert Korrektheitsnachweis erheblich

Agenda



ICS AG - Kurzvorstellung

Modellbasierte Entwicklung mit SCADE

Entwicklung eines digitalen, sicheren Übertragungssystems mit SCADE

Aufwandsbetrachtungen

Lessons Learned - Zusammenfassung



- Entwicklung eines Systems zur Übertragung sicherheitsrelevanter digitaler Daten der Leit- und Sicherungstechnik (LST)
 - gemäß Lastenheft DB
 - Sicherheitseinstufung: SIL 4
 - Geschlossenes Übertragungssystem gemäß EN 50159-1
 - ⇒ „eine festgelegte Anzahl oder festgelegte maximale Anzahl von Teilnehmern, die durch ein Übertragungssystem mit wohlbekanntem und festgelegten Eigenschaften miteinander verbunden sind, und bei dem das Risiko von nichtautorisiertem Zugriff als vernachlässigbar betrachtet wurde“
 - Verwendung von Sicherheitscodes zur Identifizierung von Übertragungsfehlern

Aufgaben ICS

- Erstellung Planungsdokumente gemäß EN 50128
- Durchführung von RAMS-Aktivitäten
- Modellbasierte SW-Entwicklung (SCADE 5.1) auf Basis der vom Kunden gelieferten Spezifikationen
- SW / HW Integration
- Modellverifikation (MTC)
- Produkttest für das gesamte Produkt gegen die Systemanforderungen
- Validierung des Systems entsprechend EN 50128 und EN 50129
- Begutachtung der Software auf Basis der EN 50128
- Zulassungsunterstützung für die Zulassung durch das EBA

Warum SCADE?

■ Chancen

- Nutzung des zertifizierten SCADE-Codegenerators
 - ⇒ Spart Kosten im SW Lifecycle ein, da die Verifikation bei einem zugelassenen Codegenerator auf Modellebene erfolgen kann und nicht auf Codeebene erfolgen muss
 - ⇒ Ausgerichtet auf die Entwicklung sicherheitskritischer Systeme
 - ⇒ Codebezogene Dokumentation und Prüfungen für Modell entfallen
- Schnelle Erstellung eines funktionsfähigen Modells
- Frühe Verifikation im Modell mit durchgängigem Werkzeug

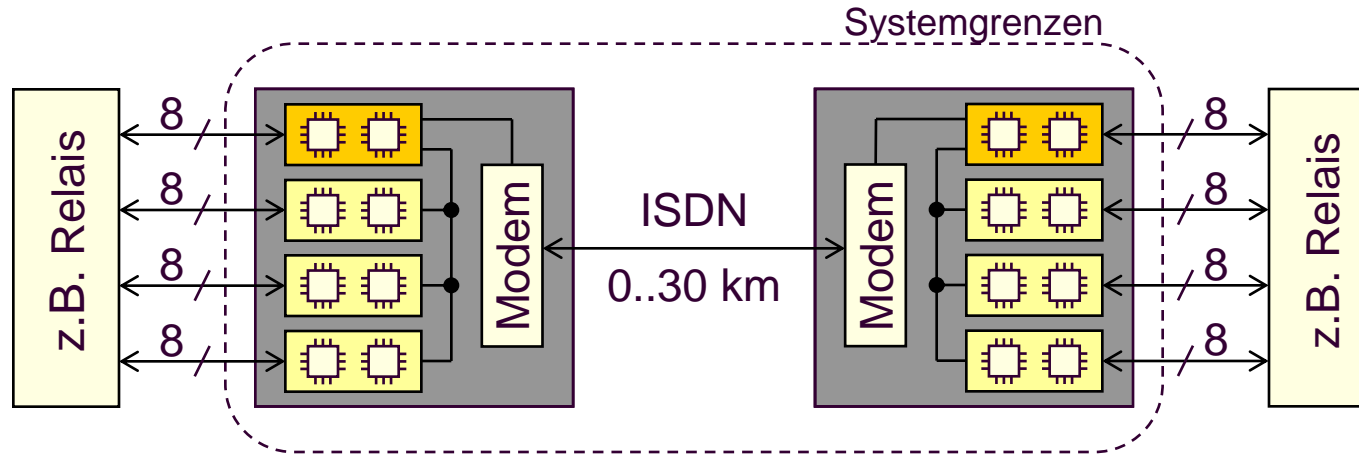
■ Risiken

- Keine offensichtlichen Risiken

■ Ziele

- möglichst großen Anteil der Software mit SCADE entwickeln
- Verkürzung der Entwicklungszeit
- Höhere Software-Qualität

Zielsetzung: Softwareentwicklung für ein sicheres Übertragungssystem für Stellwerkssignale (SIL 4)



Systemarchitektur

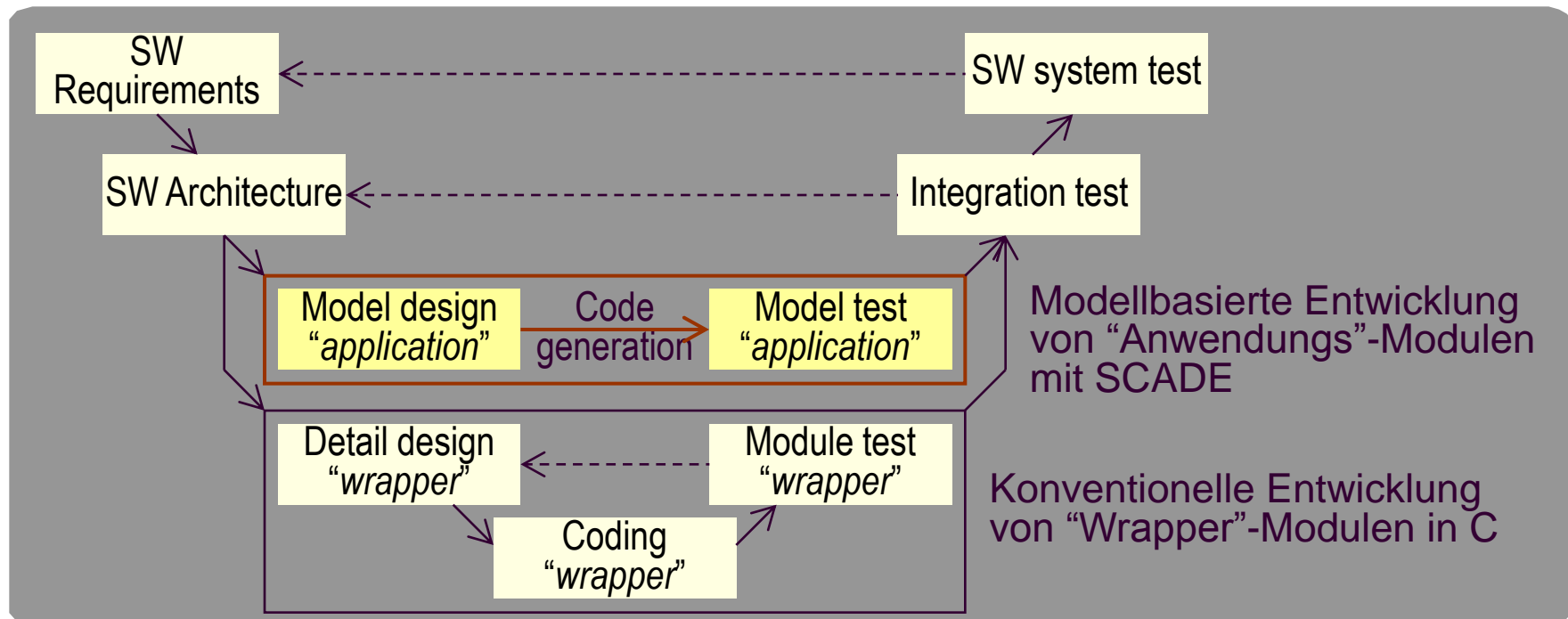
- 2 verbundene Einheiten (Duplexübertragung)
- Jede Einheit mit 1..4 redundanten Subsystemen
- 16 bit Mikrocontroller

Software-Funktionalität

- Sichere Übertragung zw. verbundenen Einheiten
- Sichere Übertragung zw. Subsystemen jeder Einheit und μ Cs jedes Subsystems
- Zyklische HW-Überwachung inkl. Online-Tests von CPU, RAM, ROM ...

Ansatz zur Anpassung des Entwicklungsprozesses

- Zielsetzung: Modellbasierte Entwicklung mit SCADE 5.1
- Einschränkung: handgeschriebene Basisfunktionen (“Wrapper”) benötigt (kein Betriebssystem verwendet!)
- Ansatz: V-Modell aufteilen in modellbasierten Zweig und konventionellen Zweig

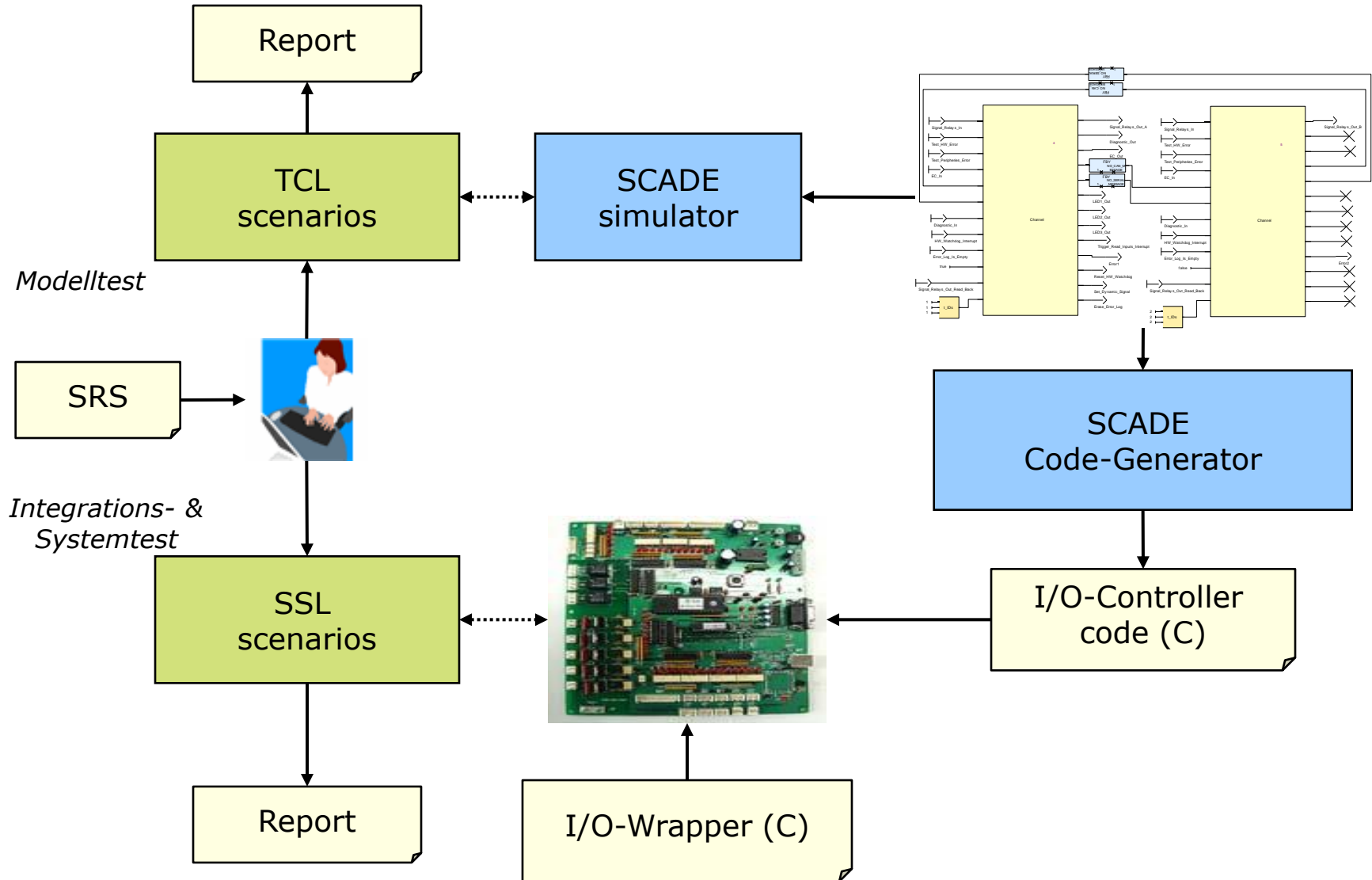


Anpassung des Entwicklungsprozesses – Features



- SW-Entwicklung: Modellentwurf der Anwendung mit SCADE (kein Coding!)
 - SW-Funktionalität: ~80% modellbasierte Module, ~20% Wrappermodule
- SW-Verifikation: Modellbasierte Techniken für Anwendungsmodule
 - Testen des Modells in SCADE-Simulationsumgebung
 - MTC (Model Test Coverage) als Testüberdeckungskriterium
 - Statische Analyse der Modellkonsistenz
 - Review auf Übereinstimmung mit definierten Modellierungsregeln
 - Wrapper- und Integrationstests konventionell
- Unterstützende Prozesse: Verwendung etablierter Prozesse
 - Projektmanagement, CM, DM, Validation, QAM, RAMS, Assessment
 - Basis: Klare Abgrenzung zwischen modellbasierter und konventioneller Entwicklung!

Testarchitektur



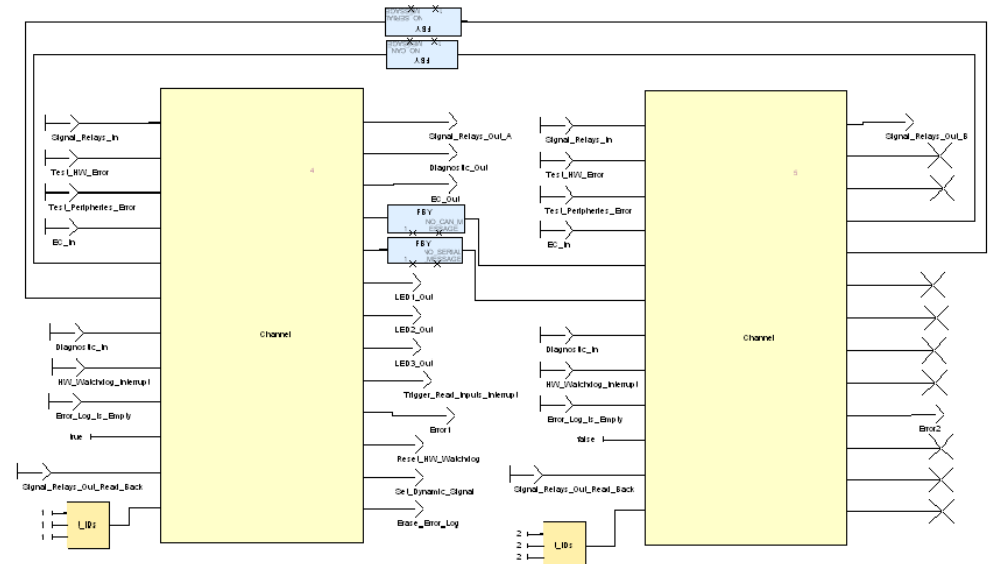
Modelltest

- TCL Szenarios
 - automatisierte Testdurchführung und -auswertung

```

reset "life tick seqNr after reconnect"
source TL_Go_Operational.tcl
cycle
checkVars Msg_Out [Create_Connect 0]
for {set i 1} {$i <= 2} {incr i} {
    cycle 50
    checkVars Msg_Out [Create_Connect $i]
}
cycle 20
setForNextCycle Msg_In [Create_ConnAck 1 2]
cycle 3
checkVars Msg_Out [Create_Indication 3 ...]
    
```

- Nachbarkanalkommunikation
 - Simulation aufwändig
 - Ansatz: Definiere Controller-Knoten durch Rückkopplung zweier Channel-Instanzen



Integrations- / Systemtest

- Anforderungsbasierter Black-Box-Test
 - Schnittstellen: 2 x Seriell, 4 x Digital-I/O, 8 x CAN
- Einsatz des ICS-Produkts SYMTES / SSL
 - Integrationstests in mehreren Stufen (Bottom-Up)
 - Automatisierte Ausführung und Auswertung der SSL-Szenarien
 - Regressionstests
 - Einsatz des SYMTES Scenario Managers

→ System zugelassen vom *Eisenbahnbundesamt!*

Herausforderungen (1)

- Nachträgliches Auslagern performancekritischer Teile
 - Unterschiedliche Anforderungen an Antwortzeitverhalten / Zykluszeit
 - ⇒ In Summe wurden 5 SCADE Modelle und 7 Wrapper-Teile implementiert
 - generierter Code für CRC Berechnung war aufwändig
 - ⇒ einige 100 Variablen im Stack benötigt
 - ⇒ CRC Berechnung musste in Wrapper ausgelagert werden
 - Beim Sortieren der CAN-Frames in SCADE werden Variablen kopiert, statt Zeiger zu versetzen
 - ⇒ Sortierung wurde in Wrapper ausgelagert
 - ⇒ SCADE wurde nur zum Dekodieren benutzt

Herausforderungen (2)

- Einschränkungen in SCADE 5
 - Verwendung von Arrays umständlich
 - ⇒ keine dynamische Indizierung, kein Slicing
 - Codegenerierung für Zustandsautomaten in Version 5 nicht zertifiziert
 - ⇒ Modellierung durch Datenflüsse etwas umständlicher
- besondere Aufwände für Modellierung mit SCADE
 - Datenflussansatz erfordert Umdenken
 - grafische Modellierung z.T. umständlicher als textuelle Programmierung
 - ⇒ Ausdrücke ($x*y + 3 > z$)
 - ⇒ Layout

Herausforderungen (3)

- SCADA 6 wäre sehr hilfreich gewesen:
 - Vollständig integrierte State Machines mit zertifizierter Codegenerierung
 - Arrayverarbeitung mit Funktionalen (map, fold)
 - dynamische Array-Indizierung mit Defaultwerten
 - Textuelle Knoten / textuelle Ausdrücke in Diagrammen

Agenda



ICS AG - Kurzvorstellung

Modellbasierte Entwicklung mit SCADE

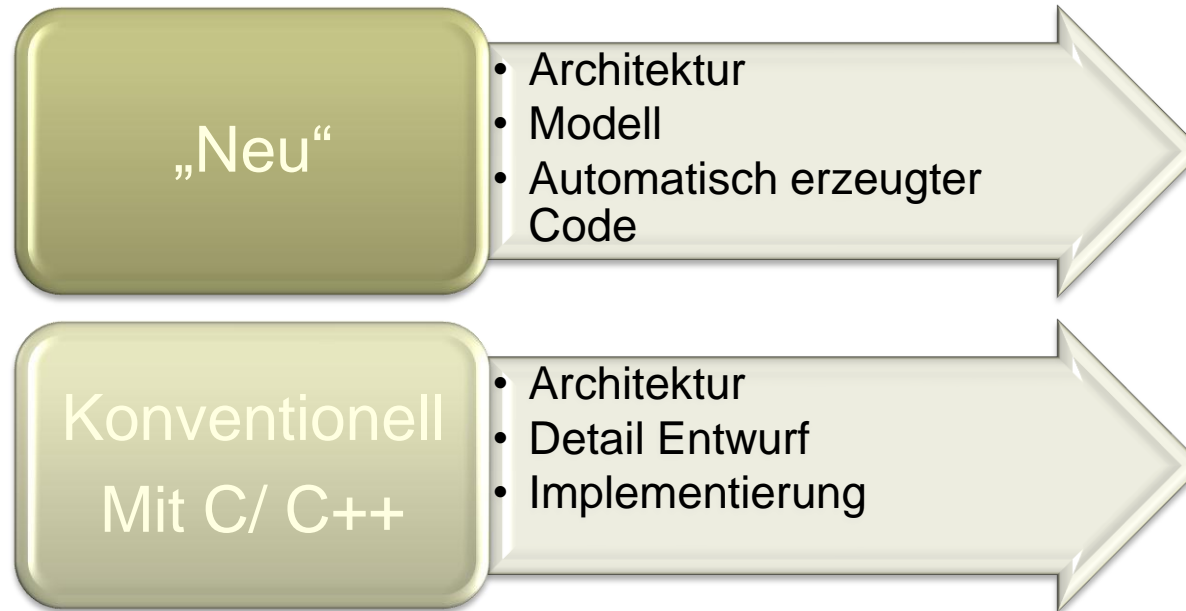
Entwicklung eines digitalen, sicheren Übertragungssystems mit SCADE

Aufwandsbetrachtungen

Lessons Learned - Zusammenfassung

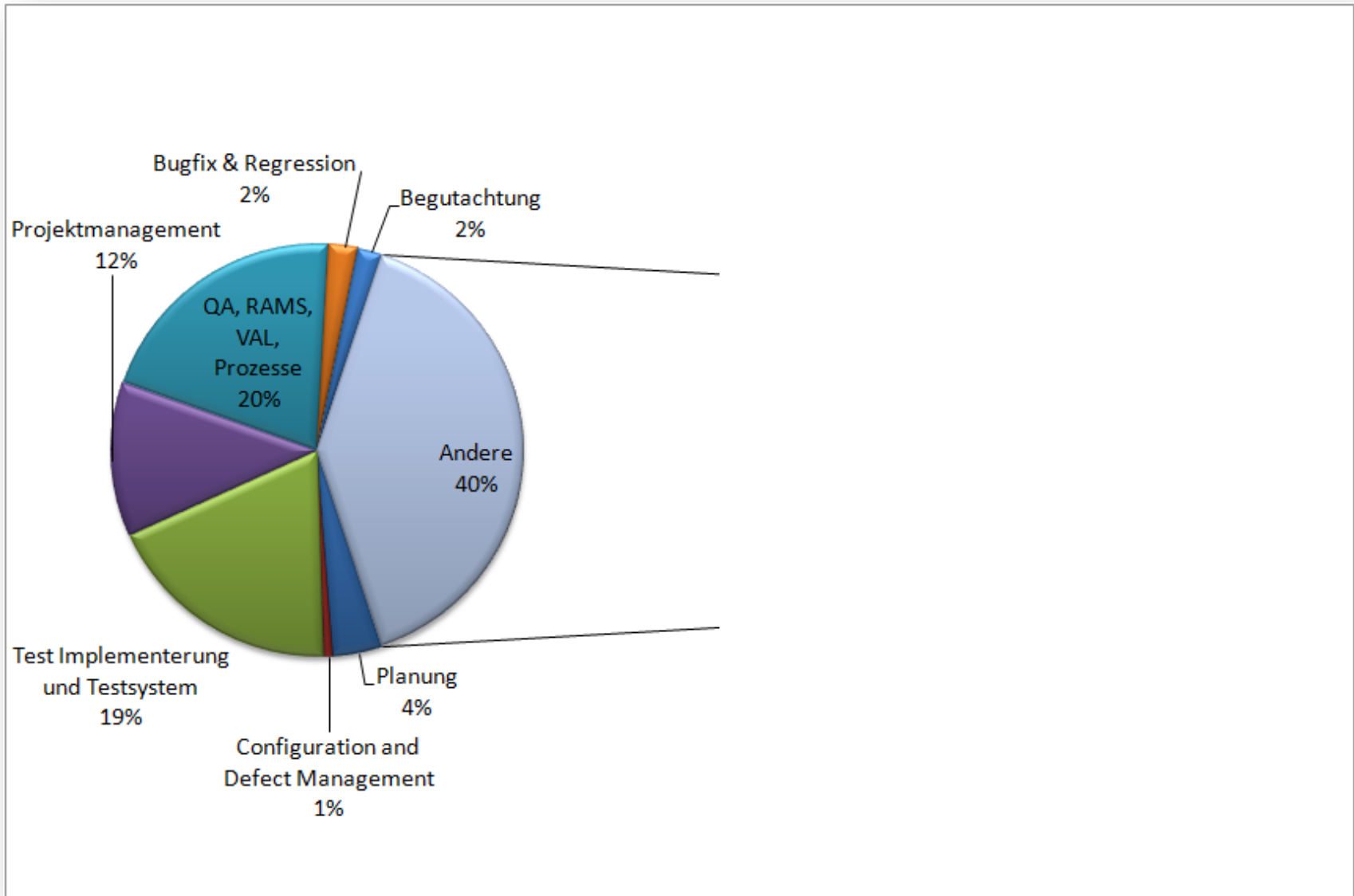


Vergleich zu konventioneller Entwicklung

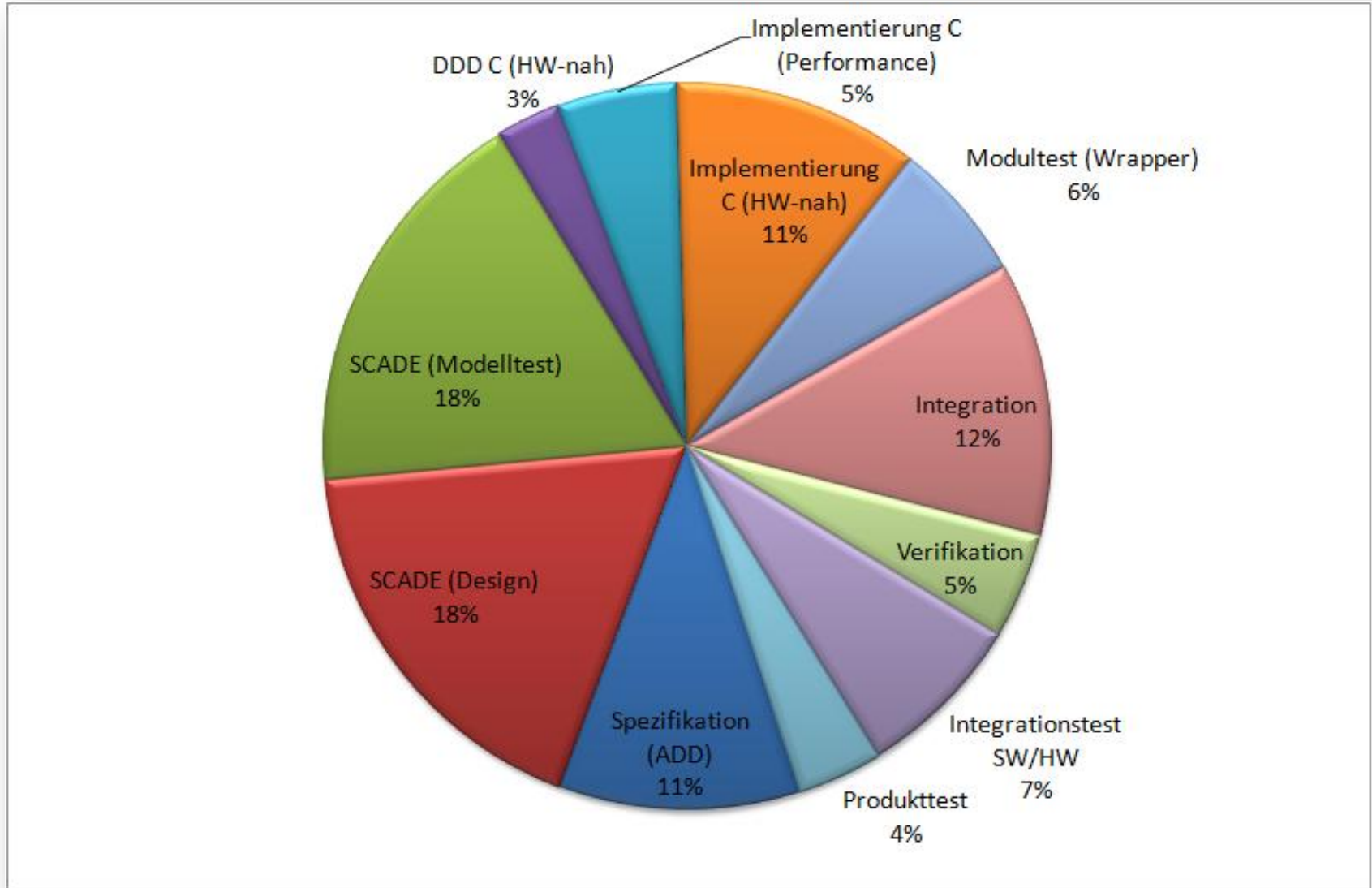


- **Laufzeit-Verhalten** verhältnismäßig einfach analysierbar aufgrund zeitsynchronen Ansatz
 - ⇒ **Geringerer Testaufwand in der Validierung**
- **Re- und Umstrukturierungen** einfacher als bei konventionellem Vorgehen
- Bei neuen Anforderungen sind Änderungen leichter zuordenbar

Aufwandsverteilung - Gesamt



Aufwandsverteilung - Realisierung



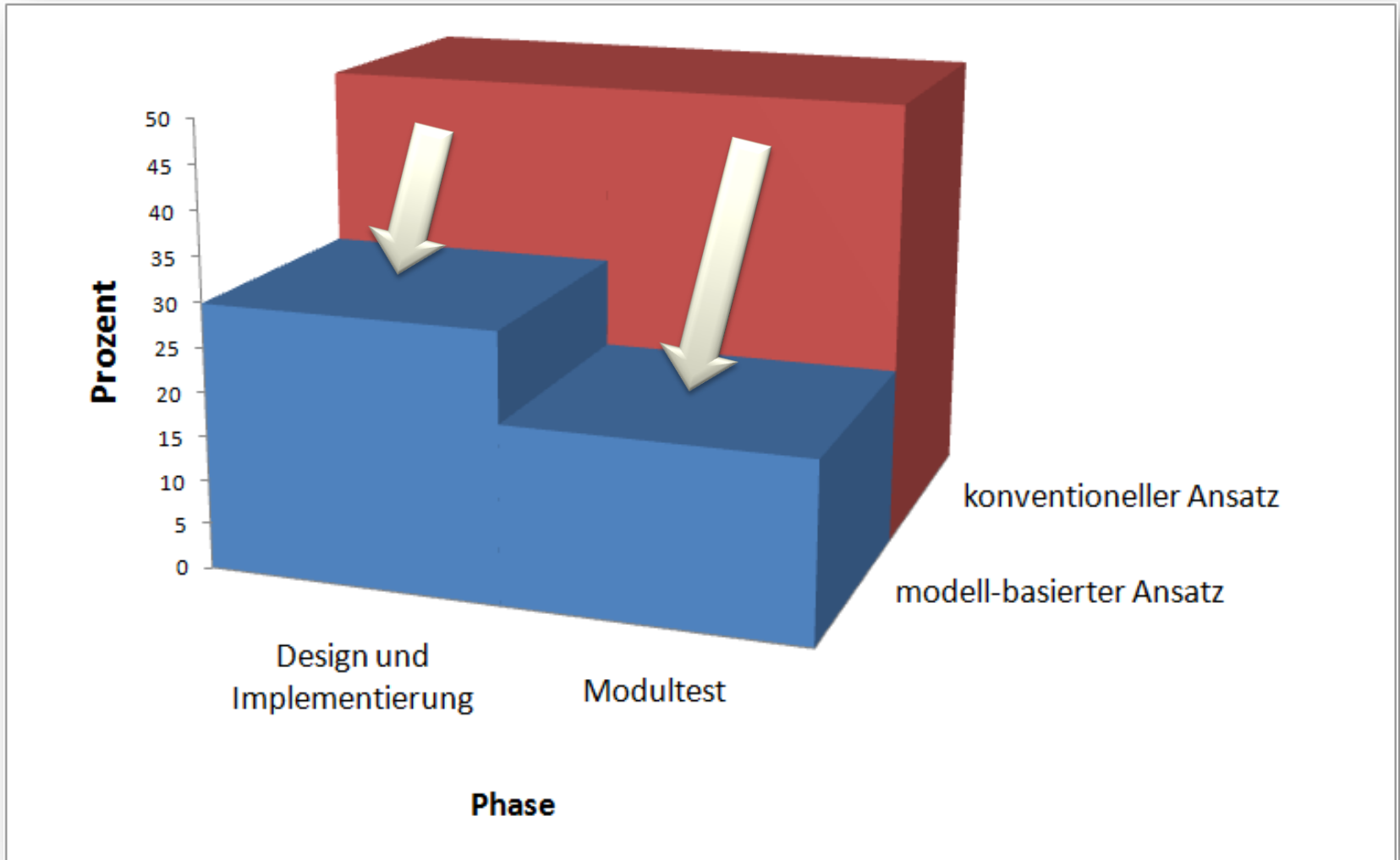
Anweisungsüberdeckung WRAPPER

- Gemäß **EN 50128** muss Anweisungsüberdeckung des Codes nachgewiesen werden. Konkret wurde für das Projekt folgendes erreicht:

MUT	Successful test cases	Failed test cases	Postponed Test Cases	Cancelled Test Cases	Test Coverage
Wrapper A	63	1	0	0	90,0%
Wrapper B	21	0	0	8	80,7%
Wrapper C	15	0	0	4	69,8%
Wrapper D	11	0	0	2	77,3%
Wrapper E	14	1	0	0	92,5%
Total	125	2	0	14	86,2%

- 13,8% mussten durch manuelles Code-Reading geprüft werden
 - Einzeln nicht abgedeckte Bereiche mussten einzeln argumentiert und protokolliert werden (warum nicht testbar, warum korrekt)

Aufwandsvergleich der Entwicklungsansätze



Agenda



ICS AG - Kurzvorstellung

Modellbasierte Entwicklung mit SCADE

Entwicklung eines digitalen, sicheren Übertragungssystems mit SCADE

Aufwandsbetrachtungen

Lessons Learned - Zusammenfassung



- **Weniger Aufwand und bessere Qualität für die Modul-Entwicklung**
 - Aufwand für Modellierung und Modell-Test \approx 50 % Aufwand von Detail Design, Codierung und Modultest

- **Weniger Aufwand für Modultest und Source Verifikation**
 - Hoher Automatisierungsgrad des Modultests und der Modell-Analyse
 - Bemerkungen:
 - ⇒ die Spezifikation der Modell-Tests anspruchsvoller als Modultest
 - ⇒ für die Verifikationsstrategie umfassende Abgrenzung des Bereiches Modellbasierte Entwicklung vs. konventioneller Entwicklung notwendig

- **Hohe Qualität der modell-basierten Software erhöht die Projekteffizienz**
 - Sehr geringe Fehlerrate während Systemtest und Feldtest
 - sehr geringe Zahl kostenintensiver Bug fixes

- **SCADE ist ausgereiftes Werkzeug**
 - Schnelle Einarbeitung
 - Intuitive Bedienbarkeit
 - Gute Stabilität
 - Guter Support

- **Eignung auch im μ C-Umfeld**
 - schlankes Code-Image
 - einfache Handhabung, da kein Multithreading im generierten C-Code

- **Einsparpotenziale**
 - Schnelle Erstellung eines funktionsfähigen Modells
 - Code-bezogene Dokumentation und Prüfungen entfallen
 - Frühe Verifikation im Modell mit durchgängigem Werkzeug
 - Frühe Validation auf Modellebene
 - Schnelle, zuverlässige Korrekturzyklen
 - Design, Code und Dokumentation sind immer konsistent
 - kein Bruch zwischen Anforderungen, Design und Coding

**Herzlichen Dank für Ihre
Aufmerksamkeit!**