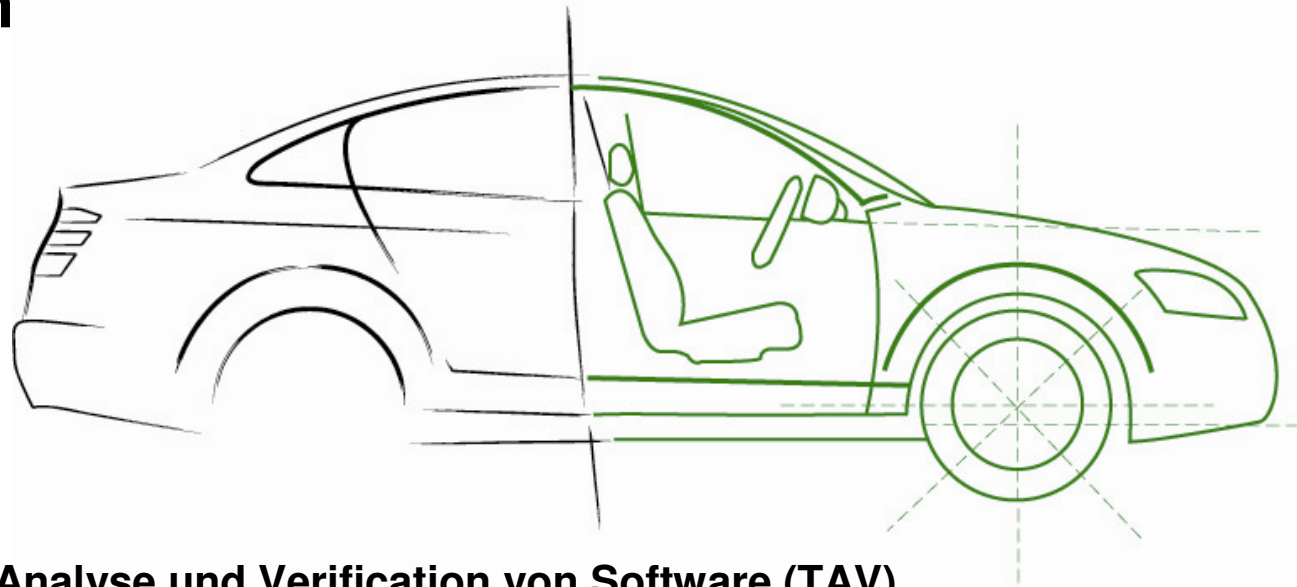




# Entwicklungsbegleitende Verifikation von AUTOSAR Steuergerätefunktionen auf Basis einer Test-RTE und SiL-Simulation



**GI-Fachgruppe Test, Analyse und Verification von Software (TAV)**

**30. Treffen: 17. und 18. Juni, Capgemini sd&m, München**

**Autor: Dipl. Ing. (FH) Michael Müller  
Berner & Mattner Systemtechnik**



# Übersicht

- Einleitung
- Herausforderungen bei der AUTOSAR Funktionsentwicklung
- Verifikation von AUTOSAR Software Komponenten
- Lösungsansatz am Beispiel einer SiL Ausführungsplattform
- Testen auf Basis der SiL Simulation
- Zusammenfassung





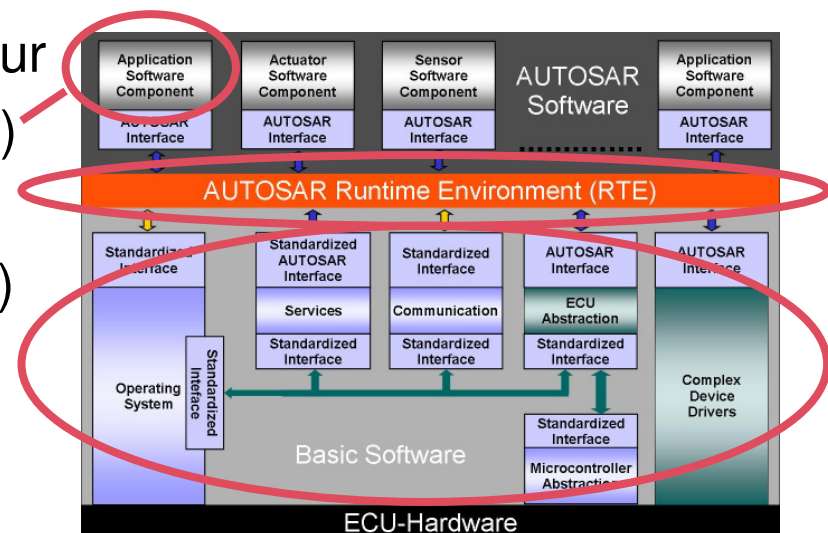
# Einleitung

Der AUTOSAR Standard definiert...

- eine technische Infrastruktur mittels einer standardisierten Steuergeräte-Basissoftware
- eine durchgängige Entwicklungsmethodik zur Entwicklung von Steuergeräte Software

Bestandteile der AUTOSAR Infrastruktur

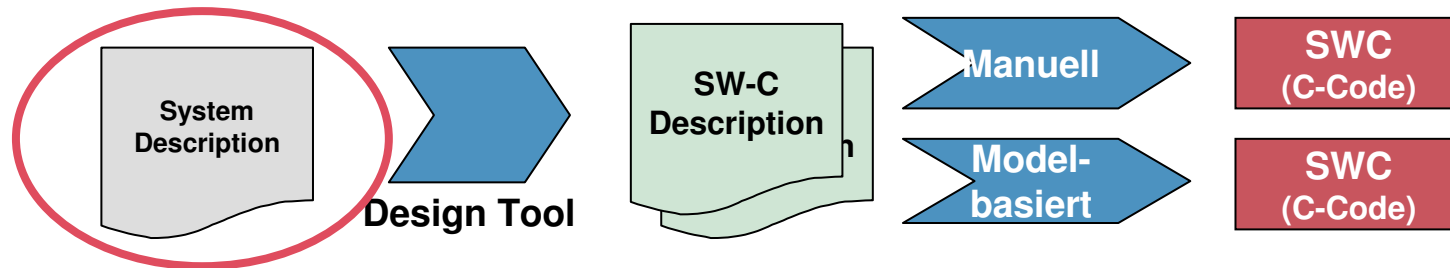
- Software Komponenten (SWC)
- Laufzeitumgebung (RTE)
- Basissoftware (BSW)
- ECU-Hardware (HW)





## Einleitung - Entwicklungsmethodik von AUTOSAR

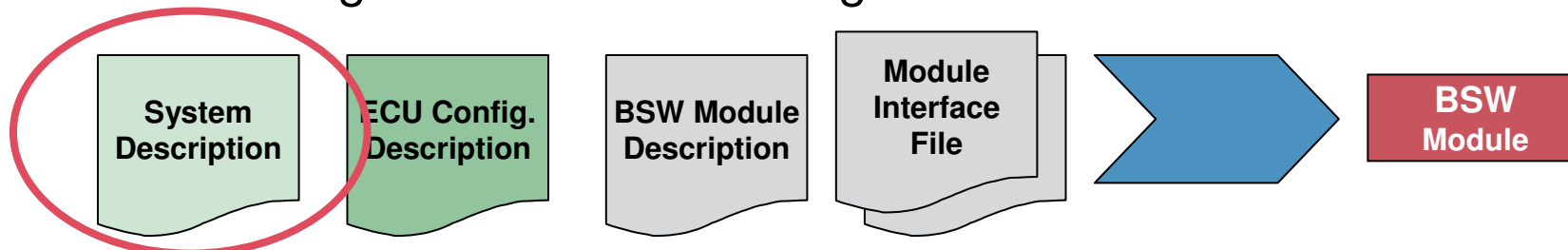
- Entwicklungsschritte für die SW Komponenten



- Entwicklungsschritte für die Generierung der RTE



- Entwicklungsschritte für die Konfiguration der Basissoftware



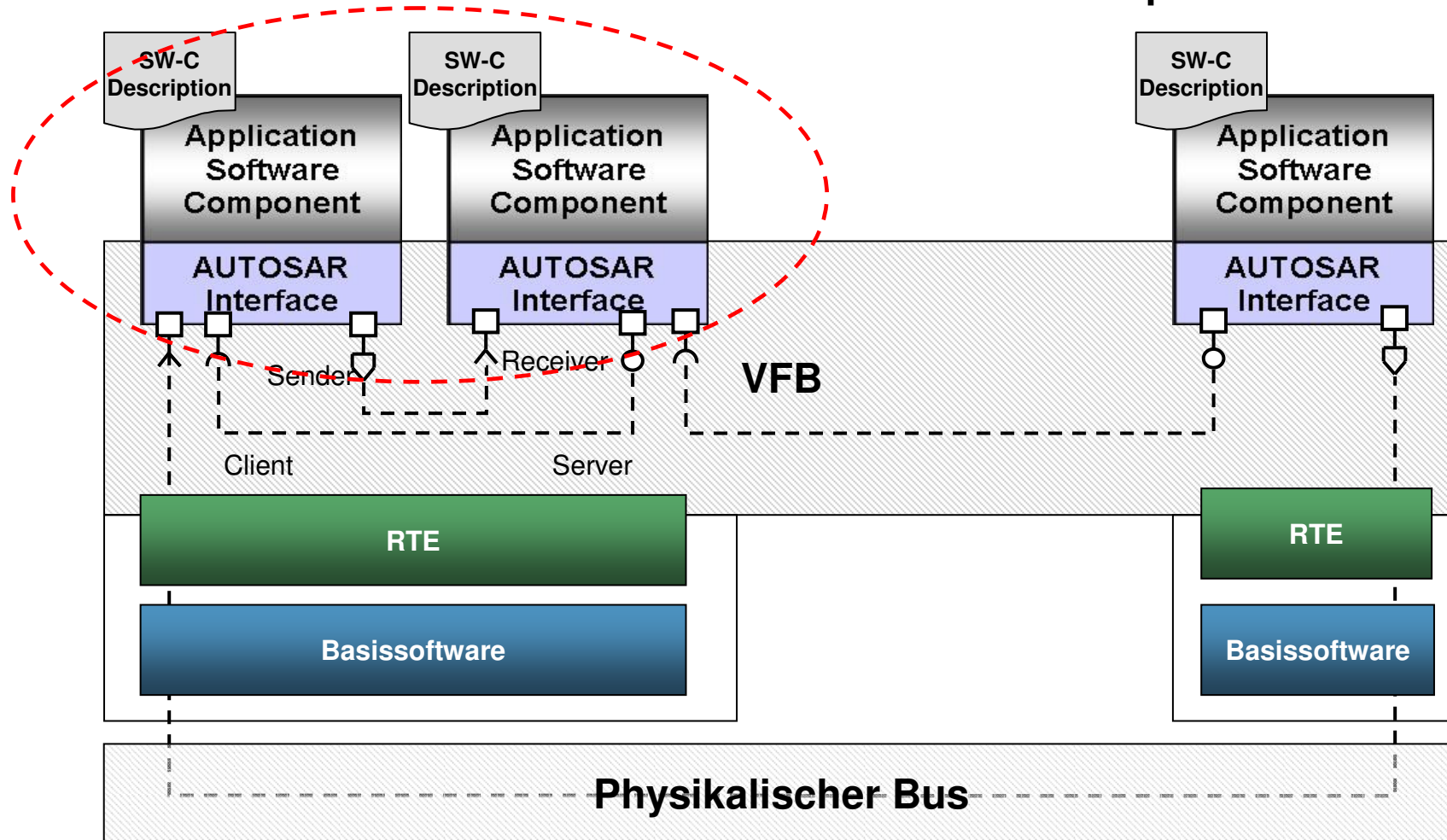


## Herausforderungen AUTOSAR Funktionsentwicklung

- **Beherrschung der iterativen Entwicklung von SWCs**
  - Hoher Anteil an Basissoftware mit umfangreichen Konfigurationen
  - Entwicklungsbegleitende Validierung & Verifikation
- **Beherrschung der eingesetzten SW-Entstehungswerkzeuge**
  - (Teil-) Wiederverwendung von Funktionen (Legacy-Software)
  - Verwendung unterschiedlicher Modellierungswerkzeugen
  - Verteilte Entwicklung und Einsatz von Standards mit zu beachtenden Schnittstellen (oftmals = vertragliche Schnittstelle)
- **Beherrschung des entwicklungsbegleitenden Testes**
  - Einsatz unterschiedlicher und geeigneter Testmethodiken
  - Integration und Test eigener und beigestellter Funktionen
- **Beherrschung steigender Qualitätsanforderungen**
  - Sicherheit, Performanz oder Kosten



# Verifikation von AUTOSAR SW-Komponenten





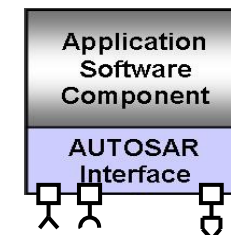
# Verifikation von AUTOSAR SW-Komponenten

- Funktionale Absicherung teilweise erst bei Verfügbarkeit der konfigurierten Basissoftware, Serien RTE und Hardware möglich
  - **späte Verifikation im Entwicklungsprozess**
- Komponentenbasierter Austausch von SW-Versionen im statischen AUTOSAR Modell kaum durchführbar
  - **zeitaufwändige Generierung aller Komponenten notwendig**
- Unit Tests einer SWC auf der RTE Schnittstelle nicht möglich
  - **Integrationstest mit allen Komponenten**
  - **Bei Fehlfunktion schwierige Fehleranalyse & -Ursache**
- Konfiguration und Generierung der Basissoftware zeitaufwändig
  - **Fehlende Fokussierung auf die Funktionsentwicklung**



## Lösungsansatz zur Verifikation von AUTOSAR SWCs

- Verfügbarkeit einer Ausführungsplattform für AUTOSAR SW Komponenten während der Entwicklung
- Dynamischer und einfacher Austausch der SW-Komponenten unabhängig von weiteren Softwarekomponenten
- Umgebungssimulation durch Modelle oder Testfallbeschreibung
- Keine Verwendung einer aufwendig konfigurierbaren AUTOSAR Basissoftware – Konzentration auf die Funktionsentwicklung
- Einsatz und Verwendung unterschiedlicher Testbeschreibungen
- Wiederverwendung von verfügbaren Schnittstellendefinitionen aus den AUTOSAR Designtools



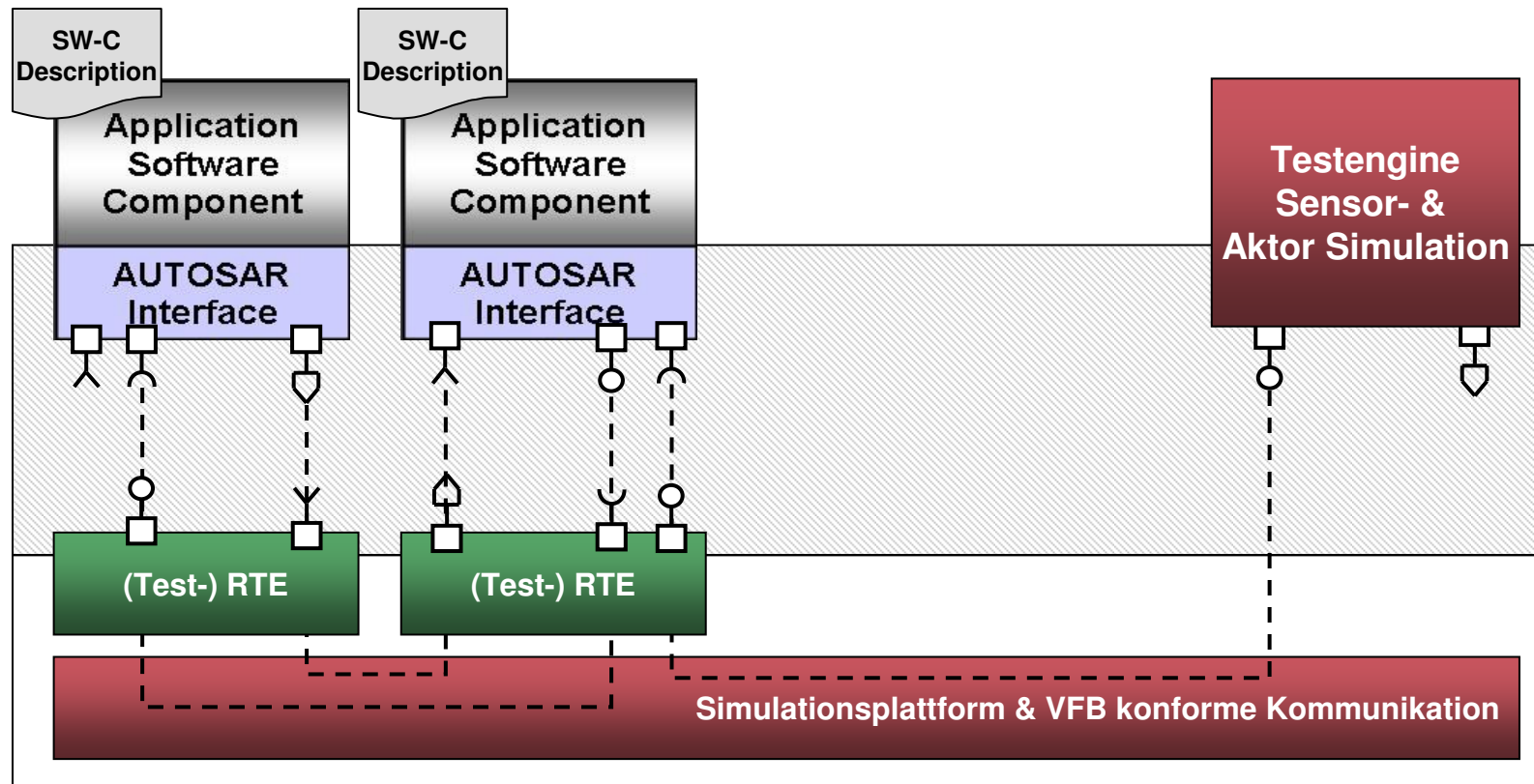


# Lösungsansatz - Virtuelle SiL-Integration

- **SiL Ausführungsplattform**
  - Import der Software Komponenten (SWC) und Generierung einer Test-RTE pro Komponente
  - Ports mit Client/Server und Sender/Receiver
  - Übernahme der Systemkonfiguration für die Kommunikation (VFB)
  - Ausführung v. Runables (zyklisch- oder eventgesteuert)
- **Umgebungssimulation**
  - Einbindung und Verwendung von nicht AUTOSAR konformen Modellen (z.B. Sensor- und Aktormodelle in ML/SL)
  - Einbindung von Legacy Software
- **Testdurchführung**
  - Nachbildung einer Gegenstellensimulation (z.B: Client-Server)
  - Schnittstellen konforme Tests auf Basis der AUTOSAR Beschreibung
  - Zugriff auf Signale der RTE für Teststimulus



## MESSINA SiL – AUTOSAR Simulationsumgebung

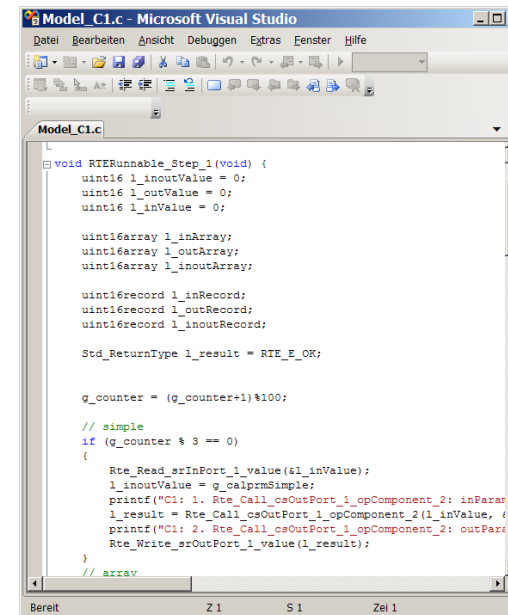
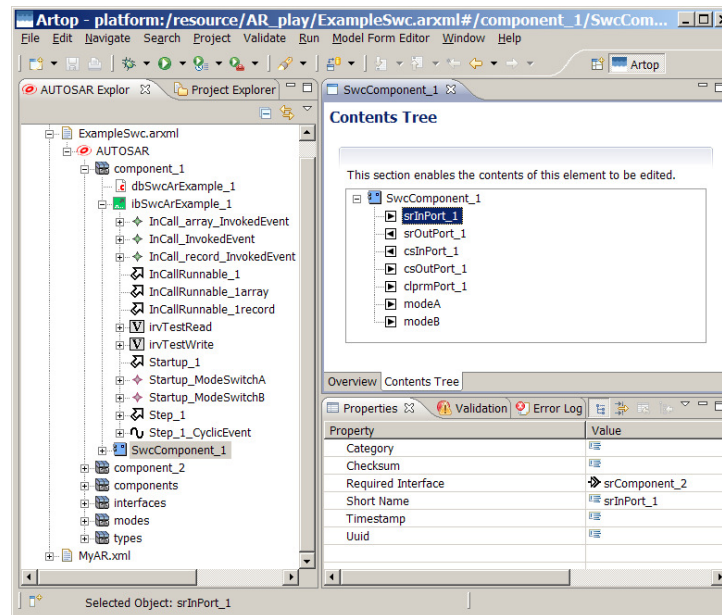


**SiL Simulationsplattform (MESSINA)**



# Beispiel – AUTOSAR Entwicklung

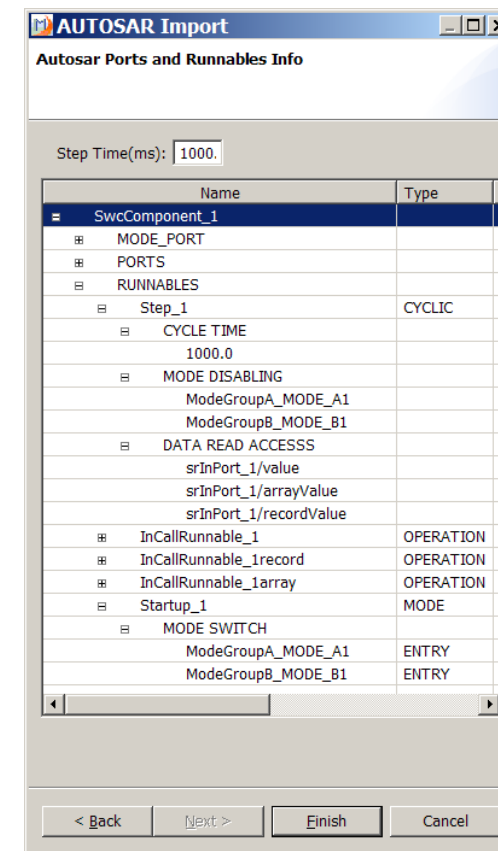
- Festlegung der AUTOSAR Komponenten, Portdefinition, Komposition,...
- Verwendung von Designtools (ARTOP, SystemDesk, DaVinci,...)
- Funktionsimplementierung (ML/SL, Visual Studio, Rhapsody,..)





## MESSINA SiL – Import von AUTOSAR Komponenten

- Einzelne SWCs werden auf Basis der SWC-Description eingelesen
  - Ports (Client/Server, Sender/Receiver)
  - Runables (Zyklisch, Event-gesteuert)
- Einlesen des generierten Codes (C-Code)
  - MATLAB/Simulink, ASCET...
  - Handgeschriebener C-Code
- Generierung ausführbarer Komponenten
  - Object Code
- Generierung der (Test-) RTE
- Hinzufügen zur Model Library
  - Dateiablage, Versionierung





# MESSINA SiL – AUTOSAR Beispiel (Demo)

The screenshot displays the MESSINA SiL development environment with several tool windows:

- Configuration Manager:** A table listing configurations for CAM and CLM.
 

Name	Type	Configuration	Module Location
CAM	AUTOSAR	CAM.xml	C:\Programme\MESSINA\workspace\Library\Models\AUTOSAR
CLM	AUTOSAR	CLM.xml	C:\Programme\MESSINA\workspace\Library\Models\AUTOSAR
- AUTOSAR Import:** A table showing ports and runables for the SwcCaMaster component.
 

Name	Type	Default V.	Direction	Can
MODE_PORT				
currentMode	SENDER_RECEIVER	uint8		
currentMode		0		
caCheckPowerUp	CLIENT_SERVER			
RunCa_InitCheckPowerUp_invoke	bool		IN	
RunCa_CheckPowerUp_invoke	bool		IN	
RunCa_InitHighPower_invoke	bool		IN	
ctrlFmPw	CLIENT_SERVER			
op	CLIENT_SERVER			
lockScreen	SENDER_RECEIVER			
diagWarnState	SENDER_RECEIVER			
ctrlFmPwMasterBr	SENDER_RECEIVER			
ctrlFmPwMasterCut	SENDER_RECEIVER			
trigStateAuth	SENDER_RECEIVER			
pinRequestCa	SENDER_RECEIVER			
ctrlState	SENDER_RECEIVER			
- Testfile\_1.java:** A code editor showing Java code for a test case, including comments in German and assertions.
 

```

// Fahrerkrit über Comfort-Access öffnen
SwcCaMaster_doorHandleSensorStates_caStateDoorHandleDataAuthDrd.setValue(State.DoorOpened);

// aktuellen Zeitstempel holen
long startTime=timer.getValue();
AccessRequested.setValue(true);

// Schlüsselsuche starten wird vom CAM angefordert
// Die Schlüsselsuche ist erfolgreich; Rückgabewert StartKeySearch == OK
SwcCaMaster_caStartKeySearch_caStartKeySearch_StartKeySearchStatus.setValue(abort(2));
ASSERT( SwcCaMaster_caStartKeySearch_caStartKeySearch_invoked_waitValue(trm, timeout), "Keine Schlüsselsuche" );
ASSERT( SwcCaMaster_caStartKeySearch_caStartKeySearch_idgNr.getValue() == 0, "IdgNr" );
ASSERT( SwcCaMaster_caStartKeySearch_caStartKeySearch_BlockedIDGs.getValue() == 0, "BlockedIDGs" );
ASSERT( SwcCaMaster_caStartKeySearch_caStartKeySearch_KeySearchMode.getValue() == 3, "KeySearchMode" );
      
```
- Result Manager:** A table showing test results.
 

Name	Result	Date	Duration
*AR_ComfortAccess_Sil_VirtualIntegration_10051022235	PASSED	10.05.2010 22:23:50	0:00:08.183
*AR_ComfortAccess_Sil_VirtualIntegration_100510222404	PASSED	10.05.2010 22:24:10	0:00:06.711
VirtualIntegration	PASSED	10.05.2010 22:24:10	0:00:09.711
CA_Coding.java (<default>)	PASSED	10.05.2010 22:24:10	0:00:00.110
UnlockedDurationMax = 250	PASSED	10.05.2010 22:24:11	0:00:00.093
UnlockedDurationMax = 250	PASSED	10.05.2010 22:24:11	0:00:06.304
- Signalpool Manager:** A table listing signal pools.
 

Name	Id	Length	Type	Unit	Path
Signalpool					
SwcCaMaster_currentMode_currentMode	1	1	uint8		
SwcCaMaster_caCheckPowerUp_RunCa_InitCheckPowerUp_invoke	2	1	bool		
RunCa_InitCheckPowerUp_invoke					
SwcCaMaster_caCheckPowerUp_RunCa_CheckPowerUp_invoke	3	1	bool		CAM/caCheckPowerUp
RunCa_CheckPowerUp_invoke					
SwcCaMaster_caCheckPowerUp_RunCa_InitHighPower_invoke	4	1	bool		CAM/caCheckPowerUp
RunCa_InitHighPower_invoke					
SwcCaMaster_ctrlFmPw_requestCheckPowerUp_invoke_d	5	1	bool		CAM/ctrlFmPw
requestCheckPowerUp_invoke_d					
SwcCaMaster_ctrlFmPw_requestHighPower_wakeupAliveReason	6	1	uint8		CAM/ctrlFmPw
requestHighPower/wakeupAliveReason					



# Testen auf Basis der SiL Simulation

- Vollständiger Zugriff auf die Ports der RTE (S/R und C/S)
- Testfallbeschreibung in Echtzeit Java (RTSJ Standard)
  - Nachbildung eines Servers oder Clients im Testfall als Gegenstelle
- RTE Modes Zugriff
- Methodenzähler
- ...

```

srInPort_1_value.setValue(1);
clprmPort_1_value.setValue(2);

// perform mode switch
// set mode port A
modeA_modeA.setValue((short) 1);
sleep(2000);
modeA_modeA.setValue((short) 3);

// set return values
csOutPort_1_opComponent_2_paOut.setValue(7);
csOutPort_1_opComponent_2_paInOut_OUT.setValue(88);
csOutPort_1_opComponent_2_status.setValue((short) 42);
// wait for operation call
ASSERT(csOutPort_1_opComponent_2_invoked.waitValue(invokeCounter)
// check results
ASSERT_EQUALS(42, srOutPort_1_value.getValue());
// input parameter
ASSERT_EQUALS(1, csOutPort_1_opComponent_2_paIn.getValue());
// inout in parameter
ASSERT_EQUALS(2, csOutPort_1_opComponent_2_paInOut_IN.getValue());

return 0; //0 means PASSED
    
```

Name	Data Type	Port Type
EXAMPLE_C1		
srInPort_1	SENDER_RECEIVER	IMPORT
value	uint16	IMPORT
recordValue/RecordElem1	uint16	IMPORT
recordValue/RecordElem2	uint16	IMPORT
arrayValue/0	uint16	IMPORT
arrayValue/1	uint16	IMPORT
arrayValue/2	uint16	IMPORT
srOutPort_1	SENDER_RECEIVER	OUTPUT
csInPort_1	CLIENT_SERVER	IMPORT
csOutPort_1	CLIENT_SERVER	OUTPUT
clprmPort_1	CALPRM	IMPORT
modeA	SENDER_RECEIVER	IMPORT
modeB	SENDER_RECEIVER	IMPORT



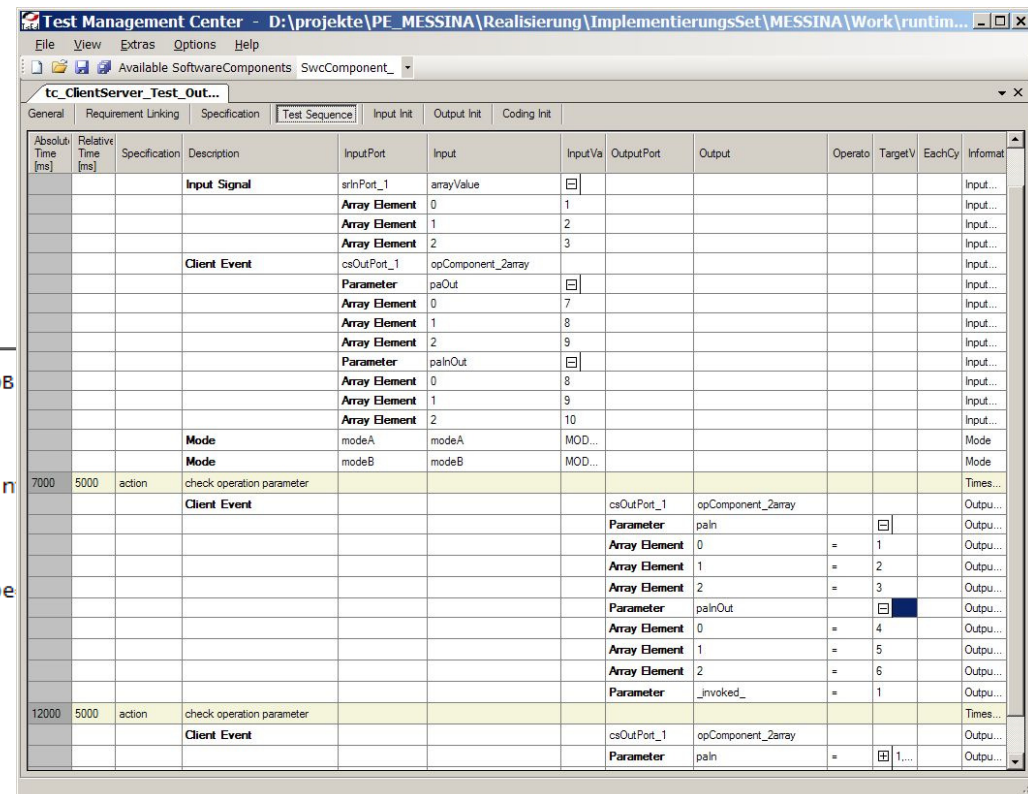
# Testen auf Basis der SiL Simulation

- Tabellenbasierte Testschrittnotation für AUTOSAR
- Testformat in XML direkt ausführbar
- Unterstützung für Ports, RTE Modes,...

```

</signal>
<Mode name="modeB" port="modeB" group="ModeGroupB
  <operation type="ASSIGN">
    <operand type="VALUE" value="MODE_B1" />
  </operation>
</Mode>
<Calprm name="value" port="clprmPort_1" type="uint16"
  <operation type="ASSIGN">
    <operand type="VALUE" value="2" />
  </operation>
</Calprm>
<Calprm name="arrayvalue" port="clprmPort_1" type="array"
  <Elements>
    <ArrayElement pos="0" type="uint16">
      <operation type="ASSIGN">
        <operand type="VALUE" value="0" />
      </operation>
    </ArrayElement>
    <ArrayElement pos="1" type="uint16">
      <operation type="ASSIGN">
        <operand type="VALUE" value="0" />
      </operation>
    </ArrayElement>
  </Elements>
</Calprm>

```



The screenshot shows the 'Test Management Center' application window. The active tab is 'Test Sequence'. The table below represents the data shown in the 'Test Sequence' tab.

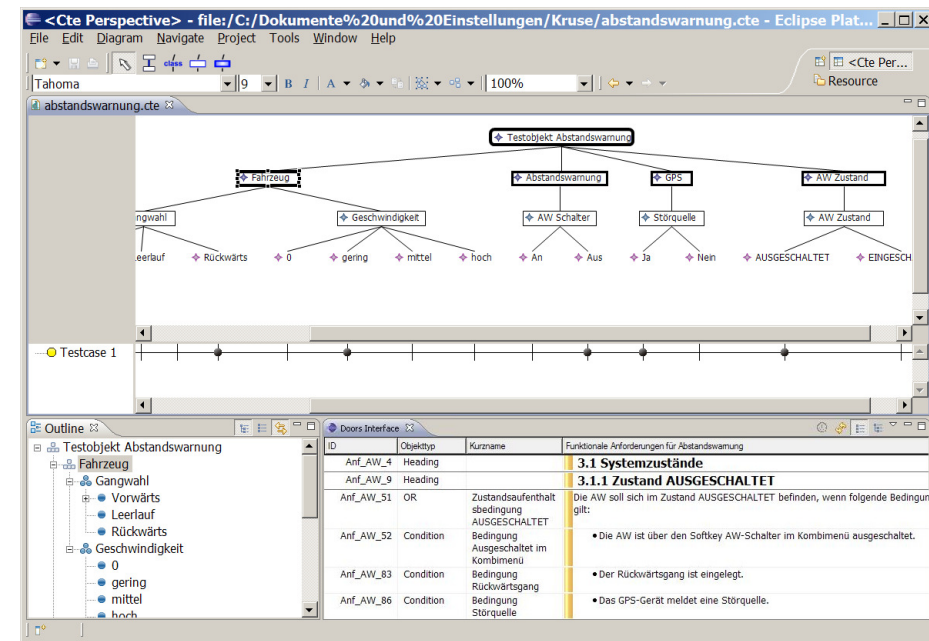
Absolute Time [ms]	Relative Time [ms]	Specification	Description	InputPort	Input	InputVa	OutputPort	Output	Operato	TargetV	EachCy	Informat
			<b>Input Signal</b>	srInPort_1	arrayValue	<input type="checkbox"/>						Input...
				Array Element	0	1						Input...
				Array Element	1	2						Input...
				Array Element	2	3						Input...
			<b>Client Event</b>	csOutPort_1	opComponent_2array							Input...
				Parameter	paOut	<input type="checkbox"/>						Input...
				Array Element	0	7						Input...
				Array Element	1	8						Input...
				Array Element	2	9						Input...
				Parameter	paInOut	<input type="checkbox"/>						Input...
				Array Element	0	8						Input...
				Array Element	1	9						Input...
				Array Element	2	10						Input...
			<b>Mode</b>	modeA	modeA	MOD...						Mode
			<b>Mode</b>	modeB	modeB	MOD...						Mode
7000	5000	action	check operation parameter									Times...
			<b>Client Event</b>				csOutPort_1	opComponent_2array				Output...
				Parameter	paIn	<input type="checkbox"/>						Output...
				Array Element	0	= 1						Output...
				Array Element	1	= 2						Output...
				Array Element	2	= 3						Output...
				Parameter	paInOut	<input type="checkbox"/>						Output...
				Array Element	0	= 4						Output...
				Array Element	1	= 5						Output...
				Array Element	2	= 6						Output...
				Parameter	_invoked_	= 1						Output...
12000	5000	action	check operation parameter									Times...
			<b>Client Event</b>				csOutPort_1	opComponent_2array				Output...
				Parameter	paIn	=					1...	Output...



# Testen auf Basis der SiL Simulation

Parametrisierung der Testfälle (z.B. Diagnosedaten) mit CTE XL

- Definition der Parameter (evtl. abgeleitet aus den Testfällen)
- Festlegung der möglichen Parameterwerte
- Automatische Generierung der Varianten von ausführbaren Testkampagnen in MESSINA





## Zusammenfassung

- Reduktion der Komplexität durch bewusste spätere Verwendung der Basissoftware im AUTOSAR Entwicklungsprozess
- Schnelle virtuelle Integration und Tests einer oder mehrerer AUTOSAR SW Komponenten während der Entwicklung
- Komplexe Gegenstellensimulation durch Einsatz einer Programmiersprache für Client/Server Kommunikation möglich
- Qualitätssteigerung durch Simulation mehrerer Funktionen im virtuellen Integrationstest und damit verbesserte Spezifikationen
- Fehlervermeidung durch Datenübernahme aus den Designtools
- Automatisierte Regressionstests der AUTOSAR SWC bei neuen Softwareständen



# Diskussion

Vielen Dank für ihre Aufmerksamkeit...

**Fragen, Anmerkungen und Diskussion**

*„Program testing can be used to show the presence of bugs, but never to show their absence.“ (E. Dijkstra)*