EMES: Eigenschaften mobiler und eingebetteter Systeme

AUTOSAR

Dr. Felix Salfner, Dr. Siegmar Sommer

Wintersemester 2010/2011

AUTOSAR

- AUTomotive Open System ARchitecture
- Offene und standardisierte Softwarearchitektur für den Automobilbau
- Gemeinsam entwickelt von Automobilherstellern, Zulieferern und Werkzeugherstellern.
- AUTOSAR existiert seit 2003
- Gründungsmitglieder: BMW, DaimlerChrysler, Bosch, Continental, Volkswagen, und Siemens VDO.
- Heute
  - 9 "Core partners": Gründungsmitglieder plus Toyota, Ford, General Motors, und PSA (Peugeot + Citroen), (Siemens VDO gehört jetzt zu Continental)
  - 46 "Premium members": Weitere Autohersteller, IBM, ARM, T-Systems, etc.
  - 24 "Associate Members"

Motivation

- Mechatronische Systeme in Autos werden immer komplexer
- Produktion wird flexibler (On-demand), auch die mechatronischen Systeme müssen flexibler werden
- Lösungen / Produkte sollten skalierbar sein
- Qualität und Zuverlässigkeit sollten verbessert werden

Motivation II

Vergleich der Komplexität der mechatronischen Syteme in einem einzigen Auto (ohne Infotainment Systeme)

	1994	**2008**
Control Units	40	60
MIPS	45	1150
MHz	85	2000
MCU Storage (Programm / Daten)	1,1MB / 160 kB	19MB / 1,25MB
Transistoren	21 Millionen	340 Millionen

Quelle: Elektronik automotive: Sonderausgabe AUTOSAR

Ziele

- Schaffung von Interoperabilität und Kompatibilität durch standardisierte Schnittstellen
- Flexibilität für Produktmodifikationen
- Skalierbarkeit von Lösungen / Produkten
- Skalierbare Fehlerbehandlung
- Verbesserte Qualität und Zuverlässigkeit
- “Risk containment”
- Verwendung von “Commercial-off-the-shelf” Komponenten
- Verringerung der Kosten

Ziele II

AUTOSAR stellt nicht nur ein Betriebssystem dar sondern umfasst auch:

- Kommunikationsmuster
- Modellierungssprache
- Modellierungswerkzeuge
- Tool-chain

Spezielle Anforderungen für Fahrzeuge

- OS ist statisch konfiguriert und skaliert.
- Ressourcennutzung (Anzahl Tasks, Ressourcen, Dienste) ist statisch vom Nutzer konfiguriert
- Ausführbarkeit aus dem ROM
- Portierbarkeit von Anwendungstasks
- Vorhersagbares und dokumentiertes Verhalten des OS
- Für jede OS-Implementierung müssen Performance-Parameter bekannt sein

Geschichte

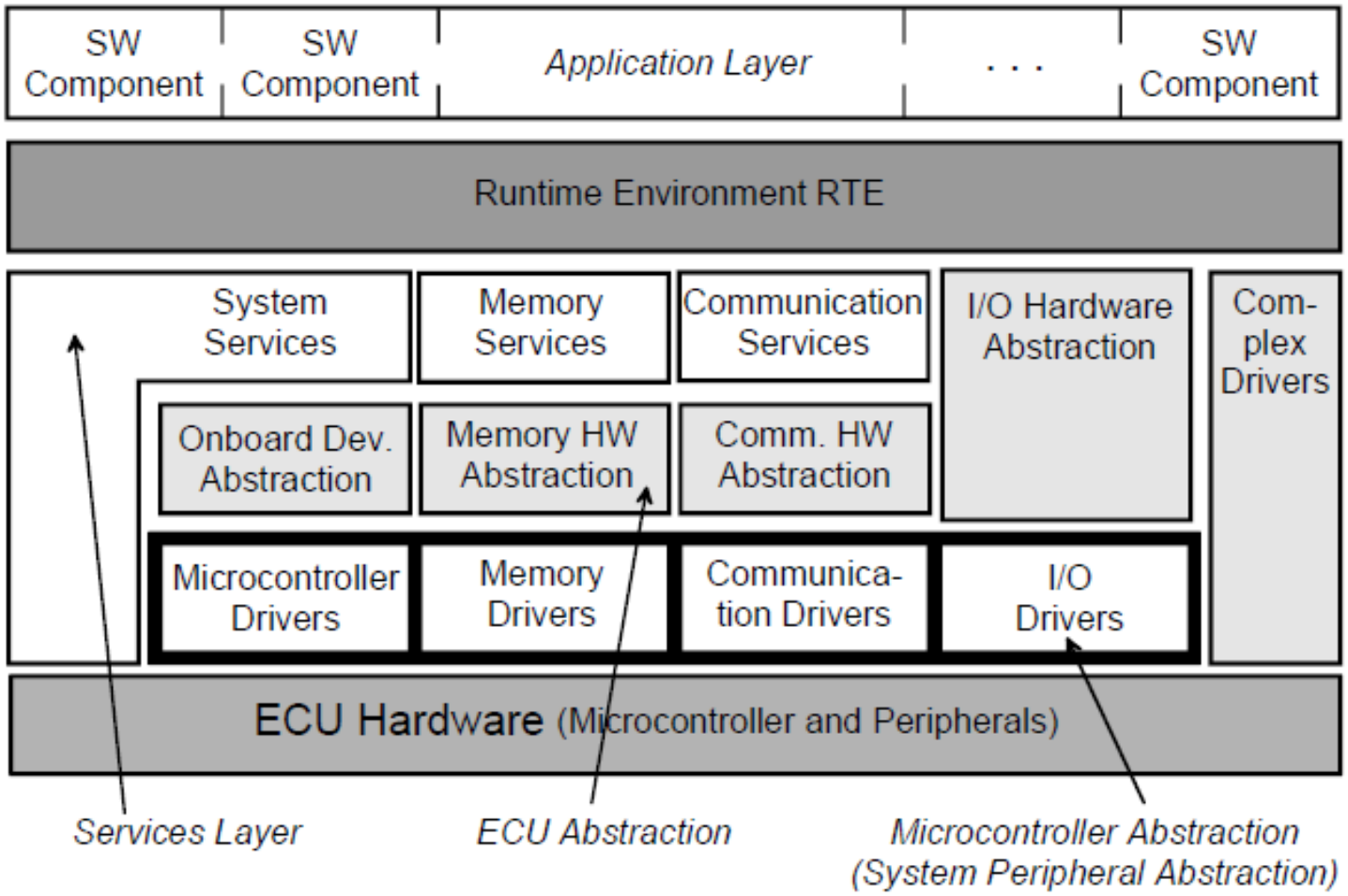
- 1993 gründen BMW, Daimler-Benz, Opel, VW, Bosch und Siemens und die Universität Karlsruhe das Gremium OSEK (Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug)
- 1994 Zusammenschluss mit der 1988 gegründeten französischen Initiative VDX (Vehicle Distributed eXecutive) zu OSEK/VDX
- Erster Standard: OSEK-OS, der seit 1997 in der Fahrzeugentwicklung eingesetzt wird.

Geschichte II

- OSEK-OS wird ergänzt durch weitere Standards:
  - zur Beschreibung von Betriebssystemobjekten (Tasks, Ressourcen, Interrupts, etc.)
  - zur Kommunikation zwischen Programmteile (auch über Controller-grenzen hinweg)
  - Network Management: Welche Geräte dürfen wann abgeschaltet sein oder müssen an bleiben?

- Später wird die OSEK Norm in die ISO 17356 Norm überführt

AUTOSAR Architektur

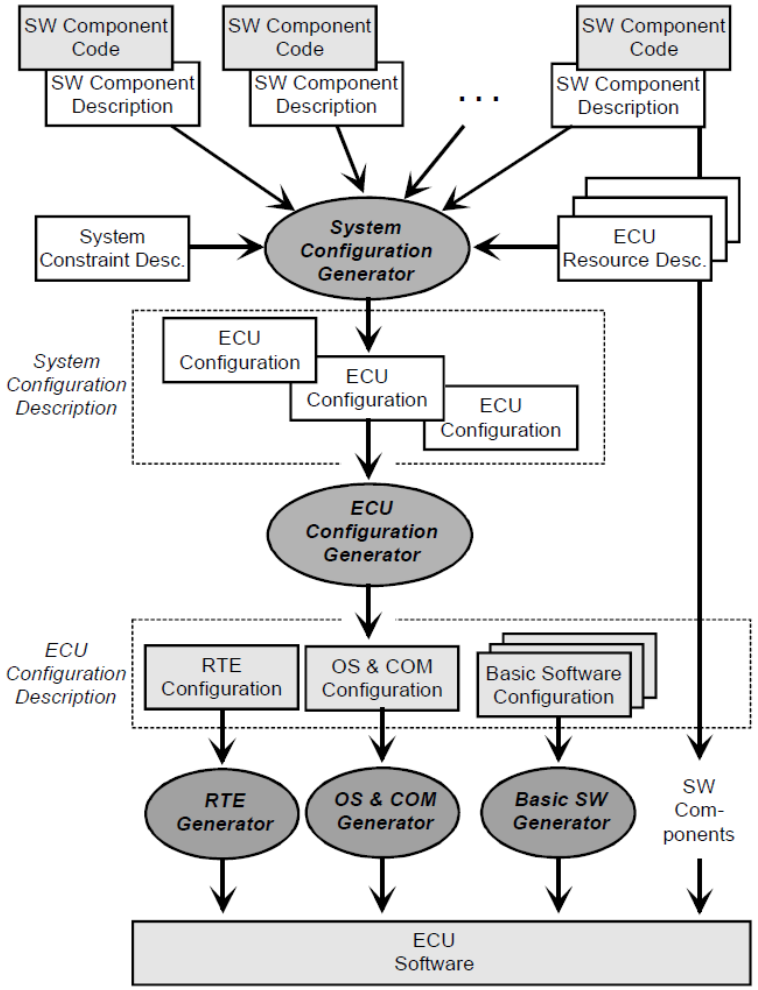


Quelle: *Bussysteme in der Fahrzeugtechnik*, Zimmermann und Schmidgall, Vieweg Teubner Verlag.

Architektur Legende

- Microcontroler Abstraction Layer: Abstrahiert den Zugriff auf den Controller und Steuerung der Microcontroller-internen Peripherie. Wird vom Prozessorhersteller geliefert.
- Electronic Control Unit (ECU) Abstraction Layer: Abstrahiert den Zugriff auf externe Peripherie-ICs. Wird vom Steuergerätehersteller geliefert.
- Services Layer: Bietet Services zur Speicherverwaltung, Kommunikationsprotokolle, etc. an
- Runtime Environment (RTE): Regelt den Datenaustausch, wird oft auch Virtual Function Bus bezeichnet.

Tool-Chain



Quelle: *Bussysteme in der Fahrzeugtechnik*, Zimmermann und Schmidgall, Vieweg Teubner Verlag.

Tool-Chain II

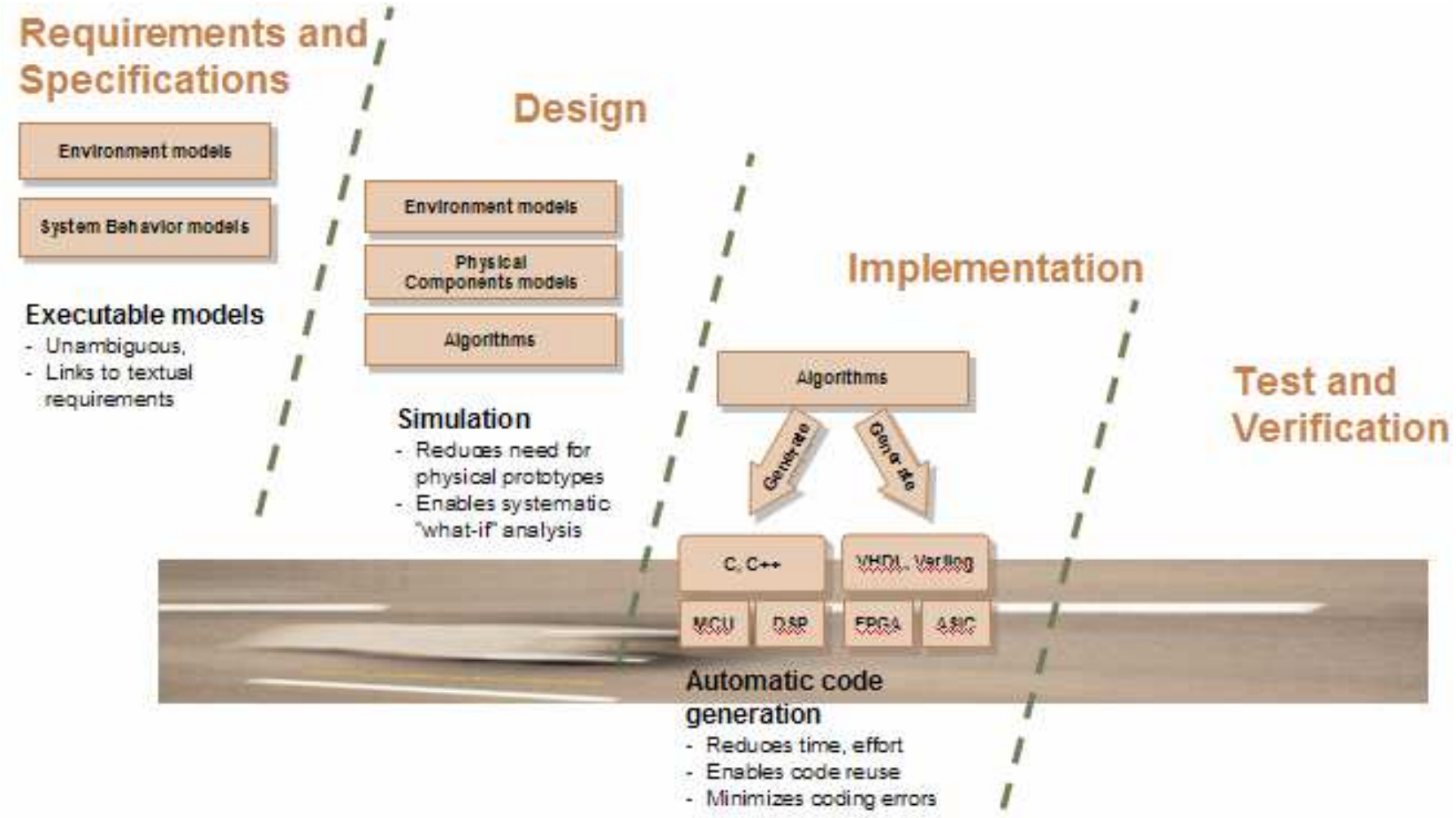
- Verbreitete Fehlerquelle bei großen und verteilten Systemen: Integration von Komponenten in das Gesamtsystem
- AUTOSAR unterstuützt die automatische Softwareintegration für ein konkretes Steuergerät
- Neben der eigentlichen Software-Komponente werden verschiedene Beschreibungsdateien benötigt:
  - SW Component Description: Beschreibt Schnittstellen, RAM/ROM Bedarf, Laufzeiteigenschaften, etc.
  - ECU Resource Description: Steuergerätehersteller spezifizieren Rechenleistung, Speichergrößen, I/O Ports etc.
  - System Constraint Description: geforderte Randbedingungen für das Gesamtsystem

Tool-Chain III

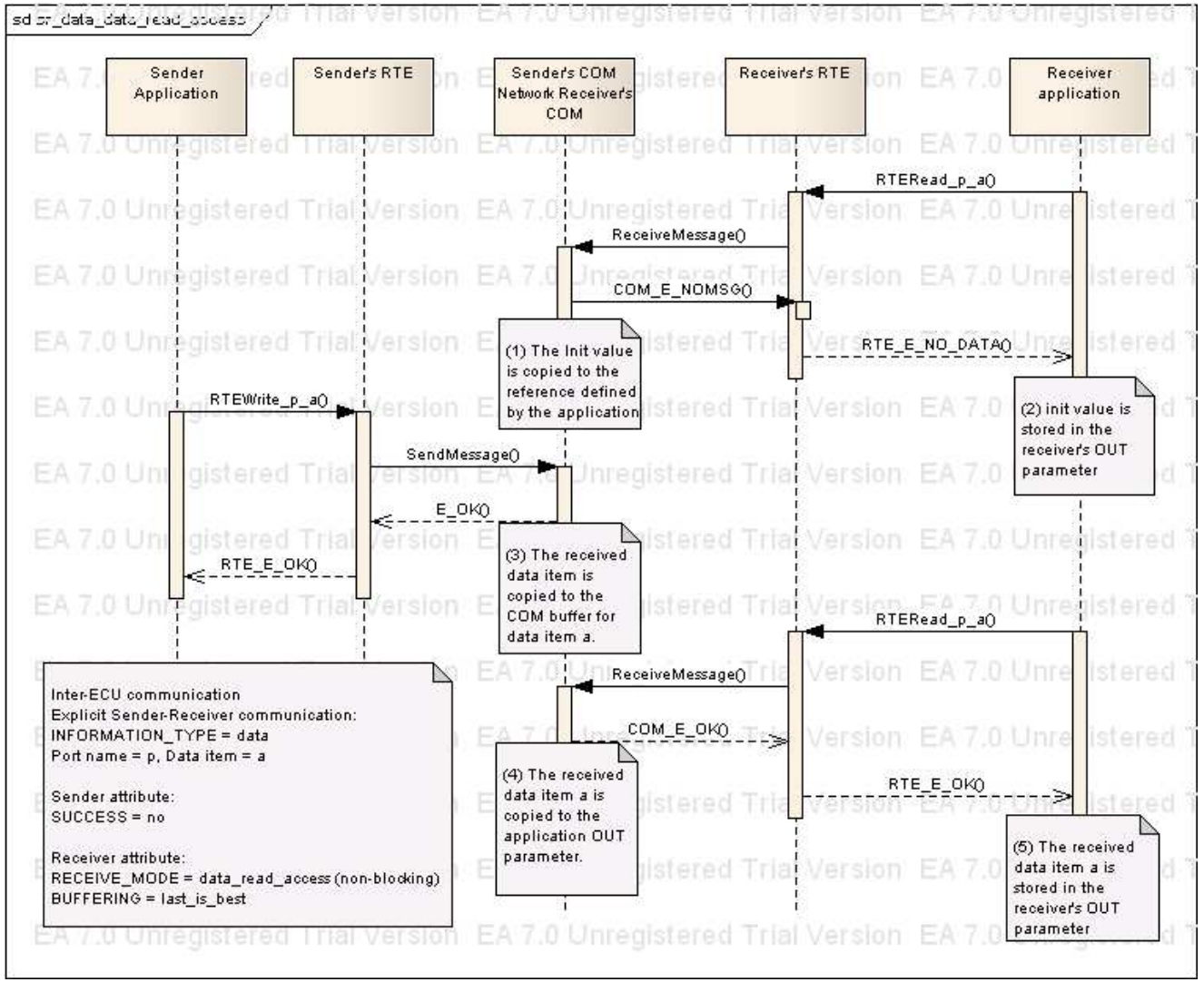
- Zunächst wird die Funktionalität auf verschiedene Steuergeräte aufgeteilt (System Configuration)
- Dann werden die Ressourcen zugeordnet (ECU Configuration)
- Zum Schluss wird der ausführbare Code für jede ECU generiert.

Model-based Development

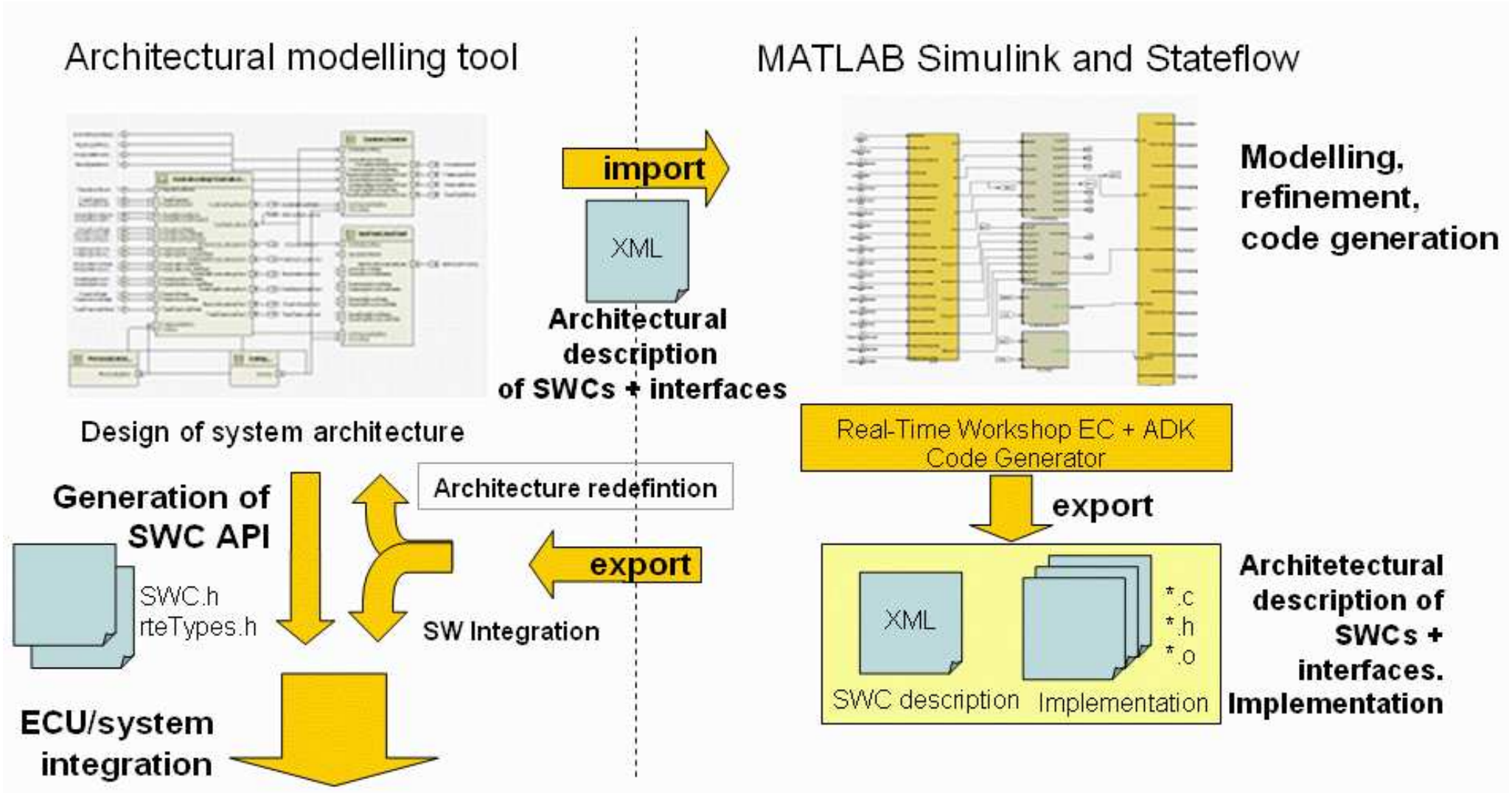
- AUTOSAR unterstützt modell-basierte Softwareentwicklung



Design mit UML (embedded profile)



Simulation with Simulink



Echtzeit in AUTOSAR

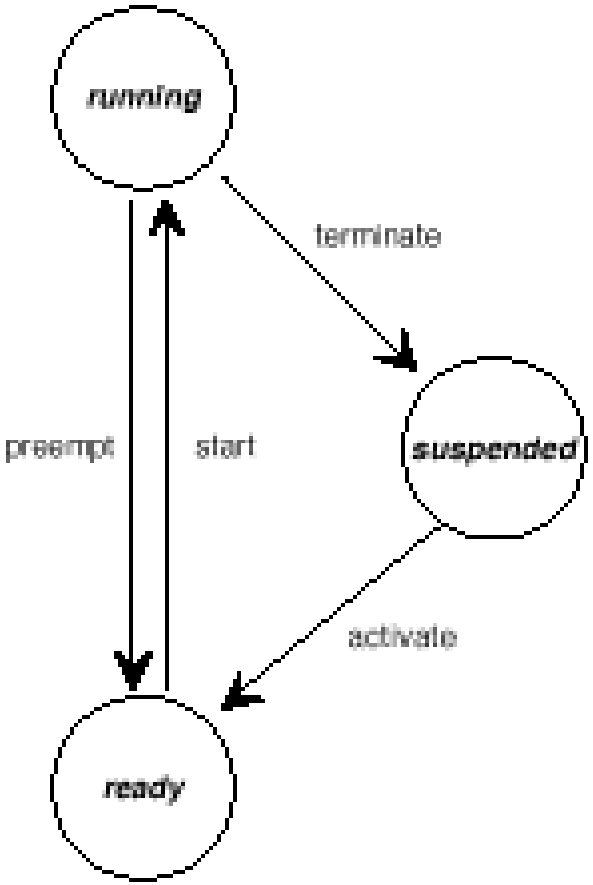
- Das Taskmodell und Scheduling von AUTOSAR basieren im wesentlichen auf dem OSEK Standard
- OSEK sah ursprünglich nur ereignisgesteuertes Scheduling vor
- Daher kommen einige Erweiterungen zum Einsatz
- Andere Erweiterungen betreffen Speicherschutz und Überwachung

$\Rightarrow$ Es wird zunächst das OSEK Modell vorgestellt und anschließend die Erweiterungen

OSEK — Taskmodell

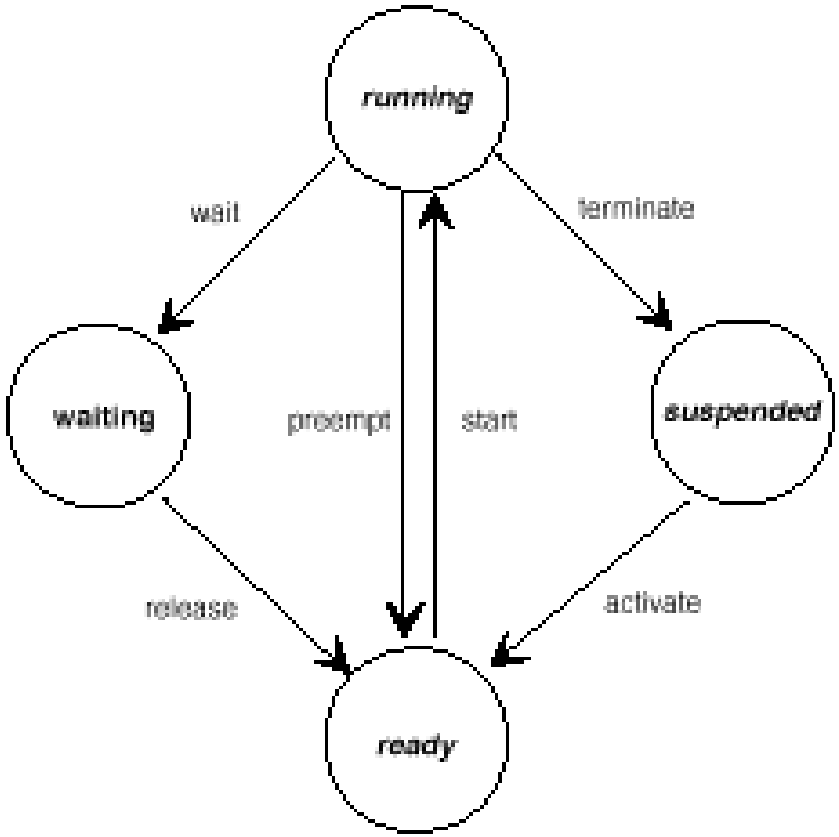
- Das OSEK-Modell unterscheidet verschiedene Konformitätsklassen, die über das unterstützte Taskmodell unterschieden werden
- Taskmodelle:
  - Basic Tasks
    Haben keinen “waiting” Zustand
  - Extended Tasks
    Können auf Events warten
- Hintergrund
  - Adaptierbarkeit in beiden Richtungen
    - Sehr einfache Systeme, die weder Synchronisation, noch kompliziertes I/O benötigen
    - Große und komplexe Systeme mit umfangreichen Tasksets, gemeinsamen Ressourcen-Zugriffen und Kommunikation zwischen den Tasks

OSEK — Basic Tasks



- running
  - CPU ist der Task zugewiesen
  - Instruktionen werden ausgeführt
  - Nur eine Task in diesem Zustand
- ready
  - Alle funktionalen Voraussetzungen für "ready" sind erfüllt
  - Andere Task ist der CPU zugewiesen
  - Warten auf Scheduler-Entscheidung
- suspended
  - Task ist passiv
  - Kann aktiviert werden

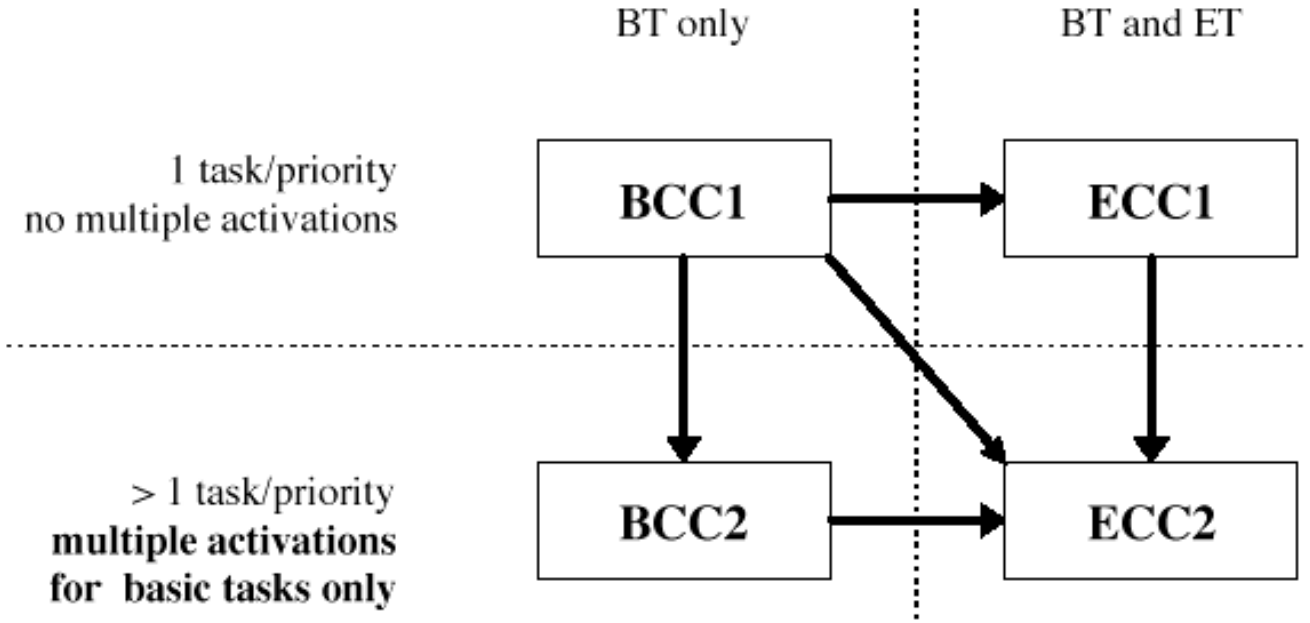
OSEK — Extended Tasks



- running
  - CPU ist der Task zugewiesen
  - Instruktionen werden ausgeführt
  - Nur eine Task in diesem Zustand
- ready
  - Alle funktionalen Voraussetzungen für "running" sind erfüllt
  - Andere Task ist der CPU zugewiesen
  - Warten auf Scheduler-Entscheidung
- waiting
  - Task wartet auf wenigstens ein Ereignis
- suspended
  - Task ist passiv
  - Kann aktiviert werden

OSEK — Konformitätsklassen I

- Zwei Gruppen von Konformitätsklassen (CC — Conformance Classes):
  - Basic (BCC) erlaubt nur Basic Tasks
  - Extended (ECC) erlaubt auch Extended Tasks
- CCs sind aufwärtskompatibel:
  - BCC1, BCC2, ECC1, ECC2

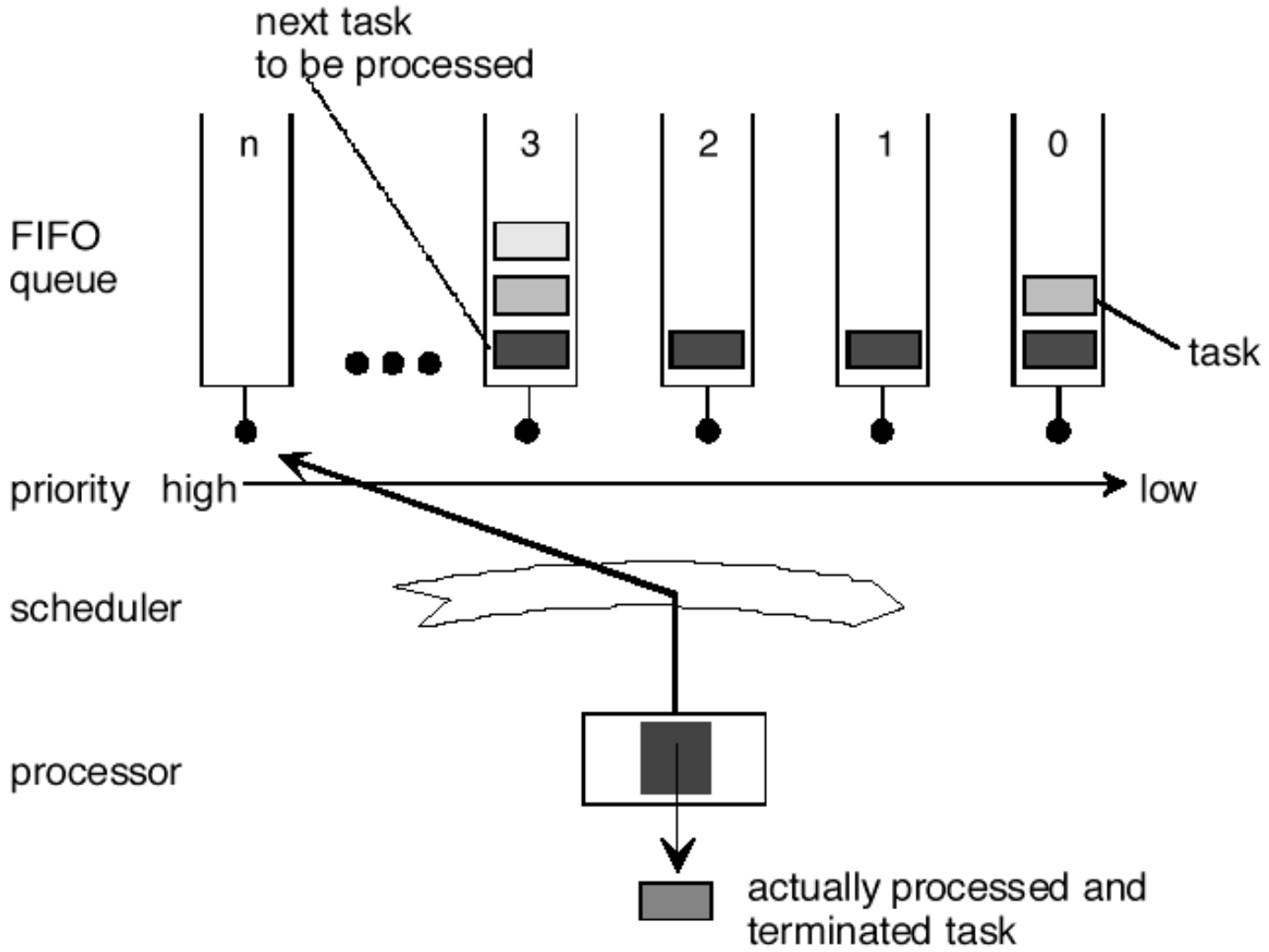


OSEK — Konformitätsklassen II

Minimale Parameter der Implementation

	BCC1	BCC2	ECC1	ECC2
Multiple requesting of task activation	no	yes	BT[9]: no ET: no	BT: yes ET: no
Number of tasks which are not in the *suspended* state	≥ 8		≥ 16 (any combination of BT/ET)	
Number of tasks per priority	1	> 1	1 (both BT/ET)	> 1 (both BT/ET)
Number of events per task	—		≥ 8	
Number of priority classes	≥ 8			
Resources	only scheduler	≥ 8 (including scheduler)		
Alarm	≥ 1 (single or cyclic alarm)			
Application Mode	≥ 1			

OSEK — Scheduling



OSEK — Event Mechanismus

- Mittel der Synchronisation
- Nur für extended Tasks
  - festgelegte Anzahl von Events pro Task
- Bewirkt Zustandsänderungen zum und vom “waiting” Zustand
- Systemrufe zum: Erzeugen, Warten, Rücksetzen und Definieren von/auf Events
- Alarme als Form von Events

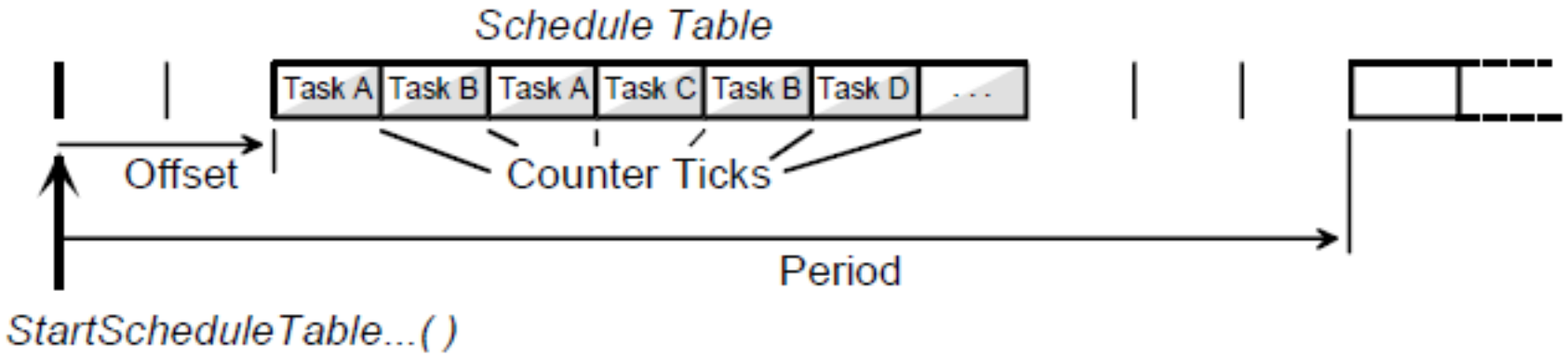
OSEK — Ressourcenverwaltung

- Obligatorisch für alle Konformitätsklassen
- Verantwortlich für prioritätengesteuerte Zugriffsverwaltung auf Ressourcen:
  - Management Entities (z.B. Scheduler)
  - Programmstücke (kritische Sektionen)
  - Speicher
  - Hardware
- Gewährleistet:
  - Gegenseitigen Ausschluß
  - Verhinderung von Prioritäteninvertierung (durch Priority Ceiling)
  - Deadlock-Vermeidung
  - Zugriff auf Ressourcen führt nicht zum Zustand “waiting”

AUTOSAR — Erweiterung von OSEK

- OSEK erlaubt nur ereignisgesteuertes Multitasking Konzept: Der Scheduler wird nur angeworfen, wenn
  - ein Task kooperativ die Kontrolle abgibt
  - durch den Aufruf von `wait` die Kontrolle abgibt
- Keine Unterstützung von zeitgesteuerten Abläufen
- Mussten in OSEK über Alarme nachgebildet werden
- wird schnell komplex und unübersichtlich
- Lösung: Scheduletabellen
  - Festlegung eines Zählers
  - Spezifikation welche Task bei welchem Zählerstand ausgeführt werden soll.

Scheduletabellen



Quelle: *Bussysteme in der Fahrzeugtechnik*, Zimmermann und Schmidgall, Vieweg Teubner Verlag.

AUTOSAR — Erweiterungen II

- AUTOSAR hat verschiedene Überwachungsmechanismen (abhängig von der Ausbaustufe)
  - Speicherüberwachung
  - Laufzeitüberwachung
- Wird ein Constraint verletzt, können die Reaktionsmechanismen konfiguriert / programmiert werden, z.B.
  - Fehlerhafte Task stoppen
  - Fehlerhafte Task neu starten
  - Ganzes System stoppen
  - Ganzes System neu starten
- Überwachung kann pro Applikation an oder abgeschaltet, werden (sogar zur Laufzeit)

Fehlerbehandlung I

- Fehlerbehandlung durch Hook-Routinen (nutzerdefinierte Aktionen im internen Ablauf des OS)
- Hook-Routinen:
  - Vom OS aufgerufen in einem speziellen Kontext
  - Höhere Priorität als alle Tasks
  - Implementationsabhängiges Interface
  - Teil des OS, aber vom Nutzer definiert und implementiert
  - Interface von OSEK standardisiert
  - Funktionalität nicht von OSEK standardisiert (und meist umgebungsabhängig und nicht portabel)
  - Dürfen nur eine Teilmenge der API-Funktionen nutzen
  - sind optional
  - Konventionen für Hooks müssen in einer OSEK-Implementierung beschrieben sein

Fehlerbehandlung II

Beispiele für Hook-Routinen:

- Systemstart
  Entsprechende Hook-Routine wird nach Start des OS und vor Start des Schedulers ausgeführt
- System-Shutdown
  Herunterfahren des Systems durch Anwendung oder im Falle eines Fehlers
- Debugging
  Anwendungsabhängiges Tracing oder Hooks zur Erweiterung von Kontext-Wechseln
- Fehler-Behandlung
  Wird gerufen, wenn ein Systemruf kein E_OK zurückgegeben hat