ENTWICKLUNGSPROZESSE I WERKZEUGE I PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME

ARAMiS II Abschlussveranstaltung
20.09.2019 Stuttgart

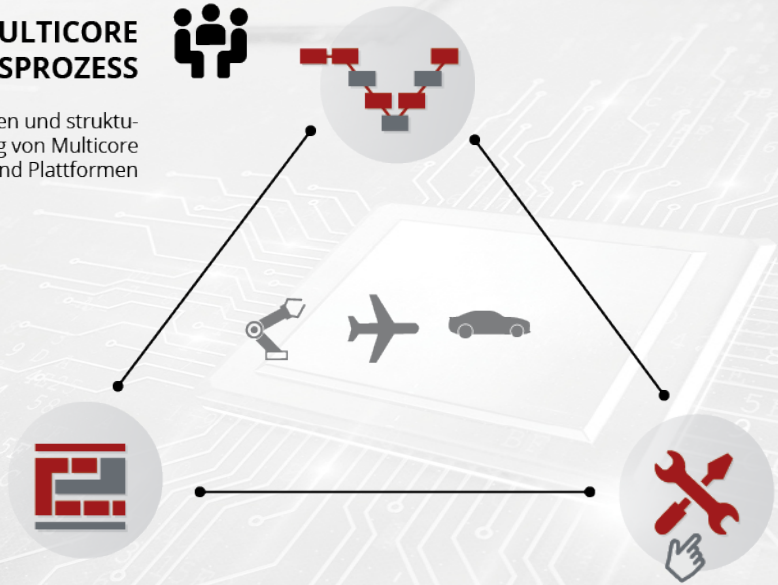
Multicore-Methoden und -Werkzeuge

Patrick Friederich, Vector

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

STRUKTURIERTER MULTICORE ENTWICKLUNGSPROZESS

Bereitstellung eines systematischen und strukturierten Ansatzes zur Entwicklung von Multicore Software und Plattformen



NEUE INDUSTRIELLE PLATTFORMEN

Entwicklung und Erweiterung von etablierten industriellen Plattformen unter Berücksichtigung Multicore spezifischer Anforderungen.

NEUE METHODEN UND WERKZEUGE FÜR DEN ENTWICKLUNGSPROZESS

Entwicklung von Methoden und Werkzeugen, welche den strukturierten Multicore Entwicklungsprozess unterstützen

TP3 Multicore Methoden und Tools

Multicore Methoden und Tools

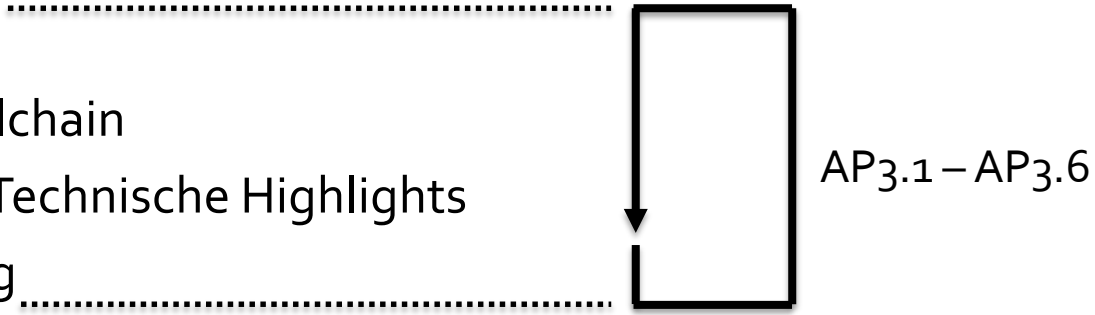
Entwicklung spezifischer Methoden und Werkzeuge zur Unterstützung der Multicore-Entwicklung

Erweiterung der Methoden für alle Schritte im Entwicklungsprozess (z.B. Partitionierung, Deployment, Scheduling)

Höherer Automatisierungsgrad in der Entwicklung durch Werkzeugunterstützung

Agenda

- Motivation und Vorgehen
- Arbeitspakete
  - Ziele
  - Workflow und Toolchain
  - Methodische und Technische Highlights
  - Zusammenfassung

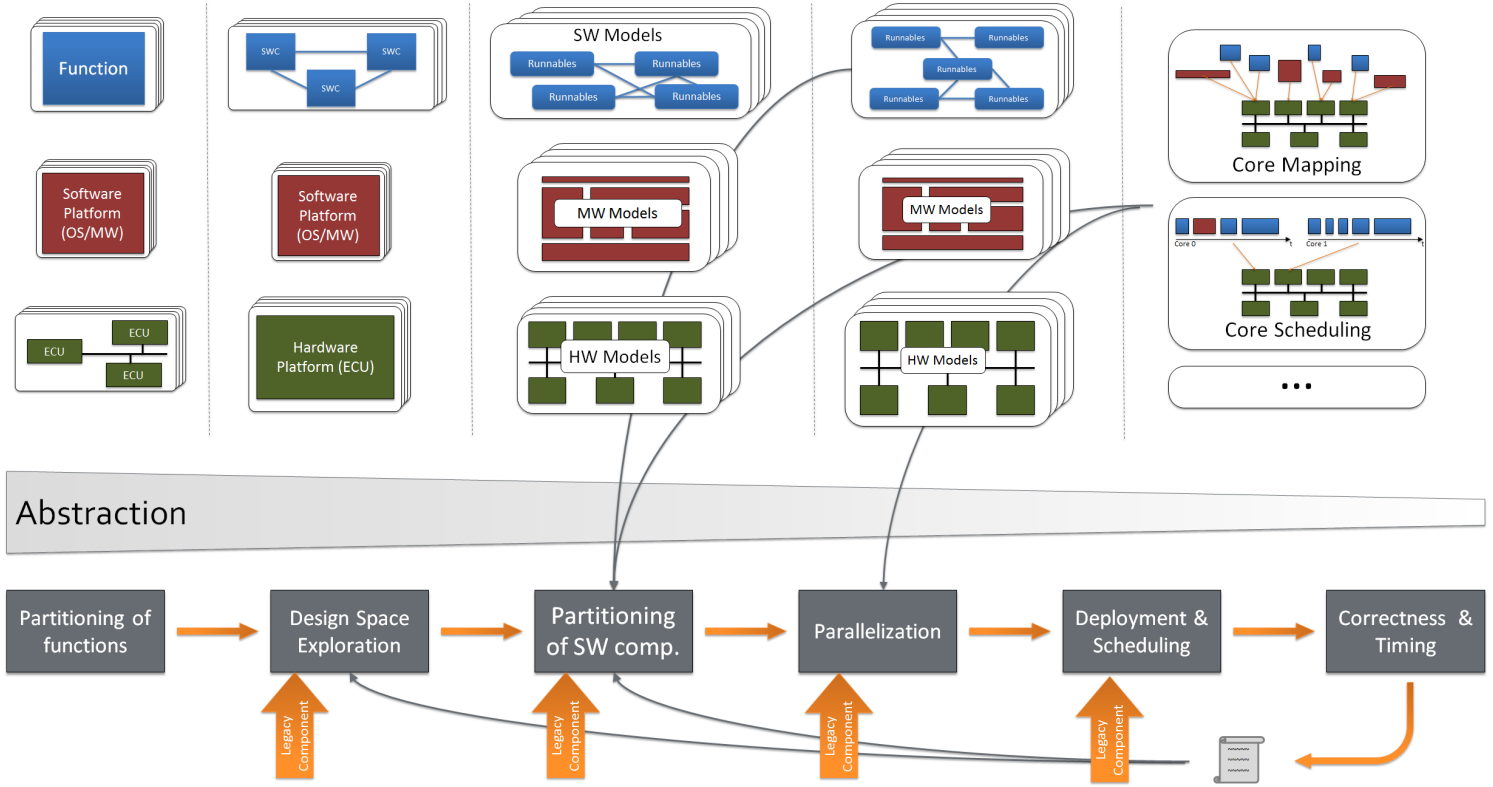


- Ergebnisse TP3

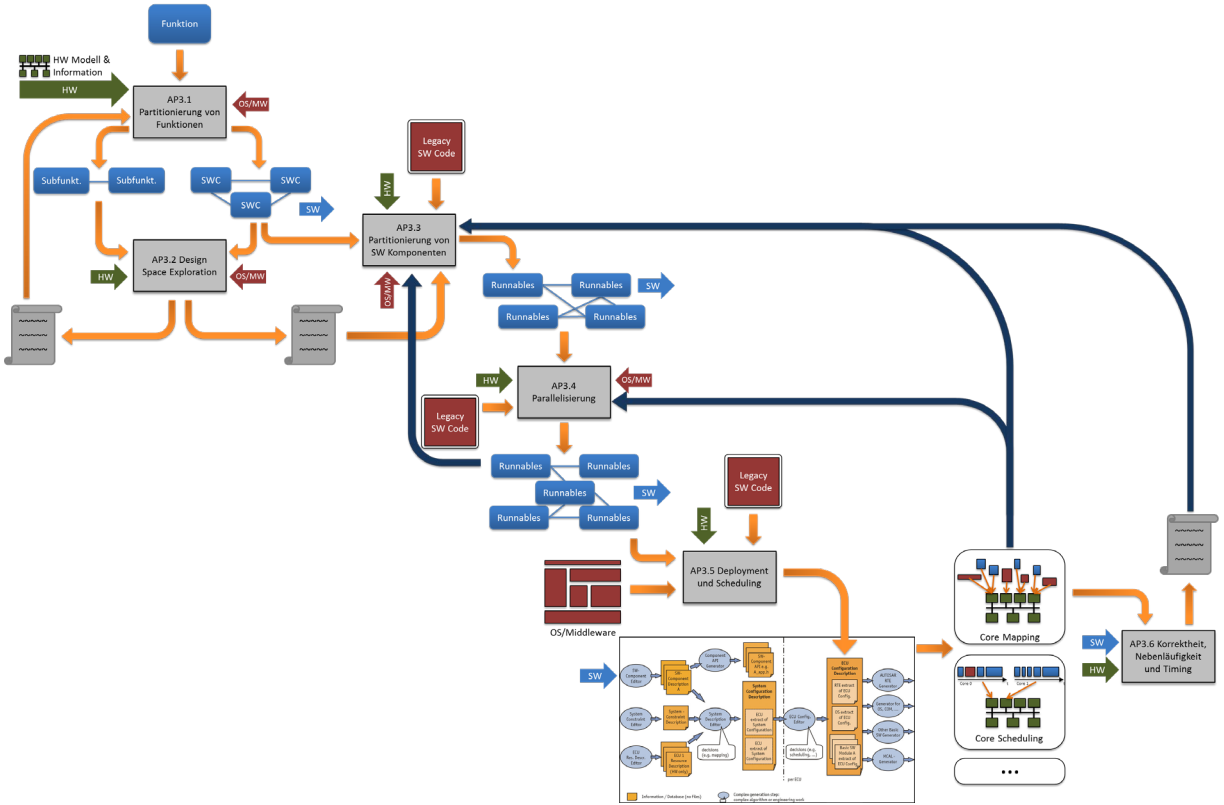
Zahlen, Daten, Fakten für TP3

- 26 Partner
- ~90k h an Aufwände
- >20 Tools
- Leitung von UnA und Vector

Wissenschaftlicher und technischer Ansatz

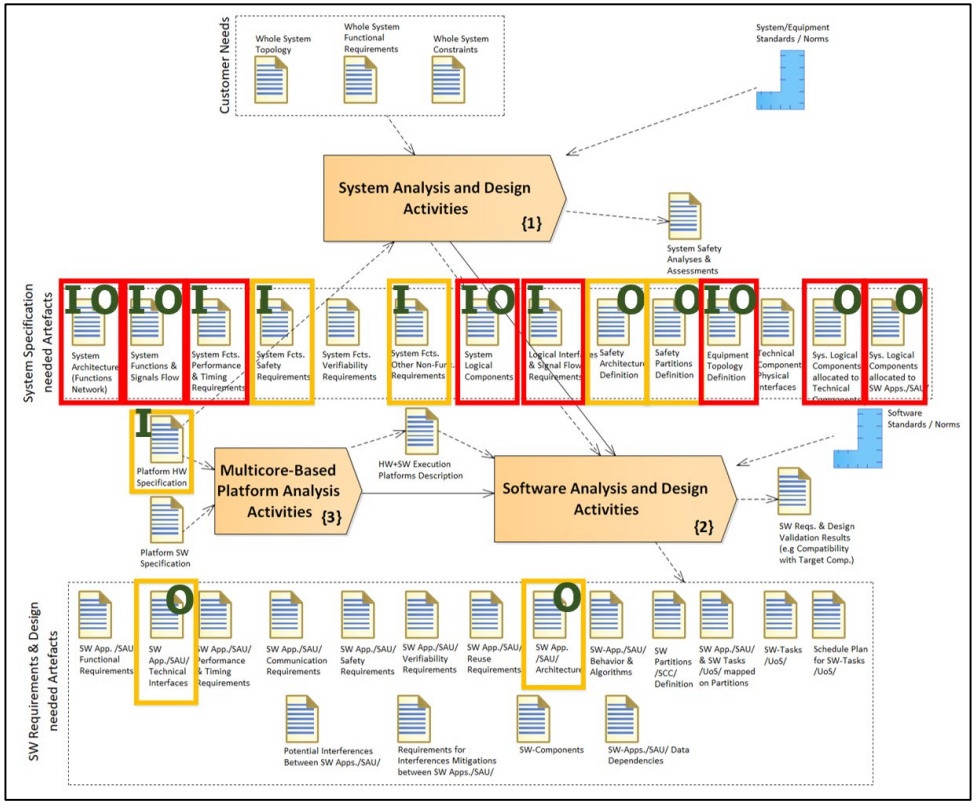


TP3 Entwicklungsprozess



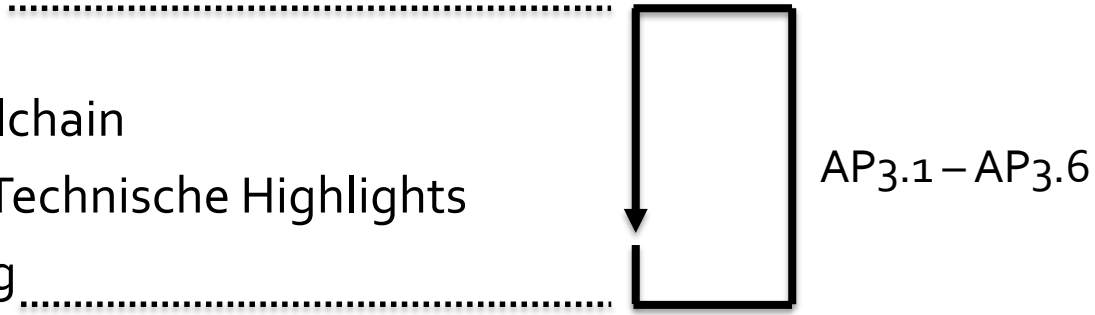
Mapping auf generischen Entwicklungsprozess

AP3.1



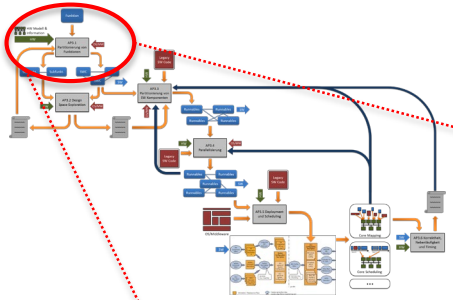
Agenda

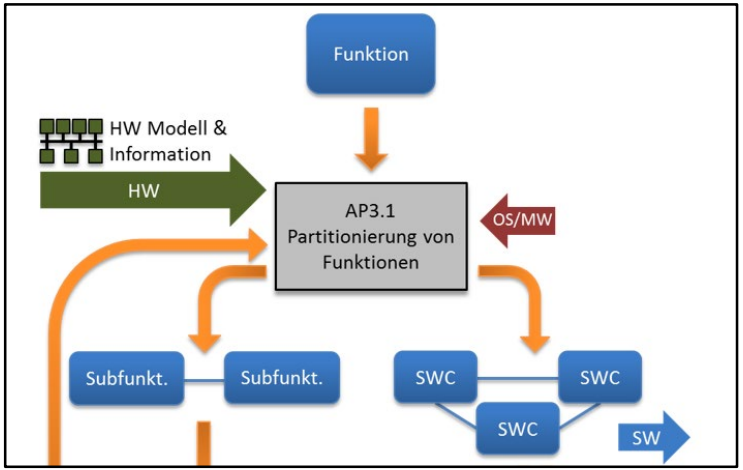
- Motivation und Vorgehen
- **Arbeitspakete**
  - Ziele
  - Workflow und Toolchain
  - Methodische und Technische Highlights
  - Zusammenfassung



- Ergebnisse TP3

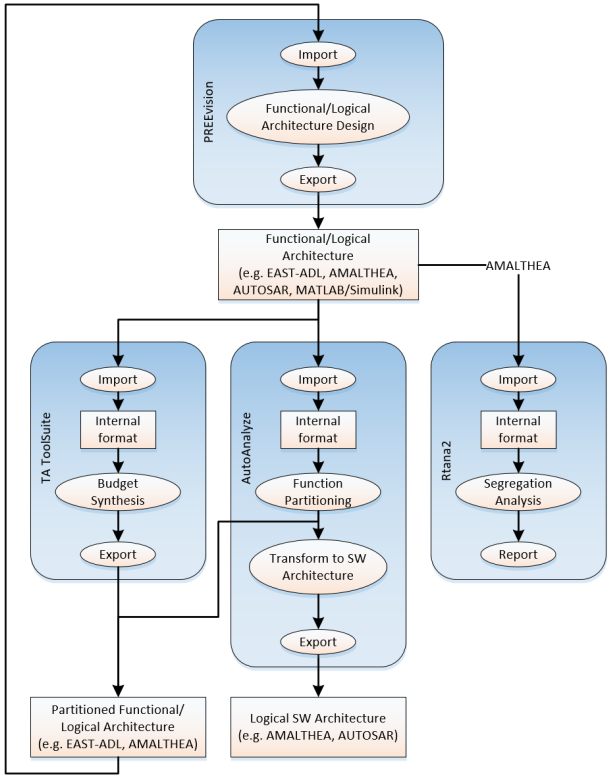
AP3.1 – Partitionierung von Funktionen





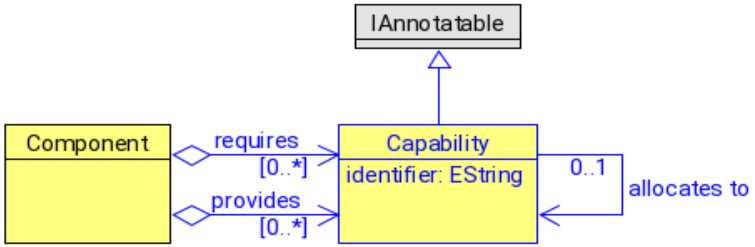
- Design Pattern für Partitionierung
- Funktions-partitionierung
- Software-Allokation
- Partitionierung auf unterschiedlichen Abstraktionsebenen

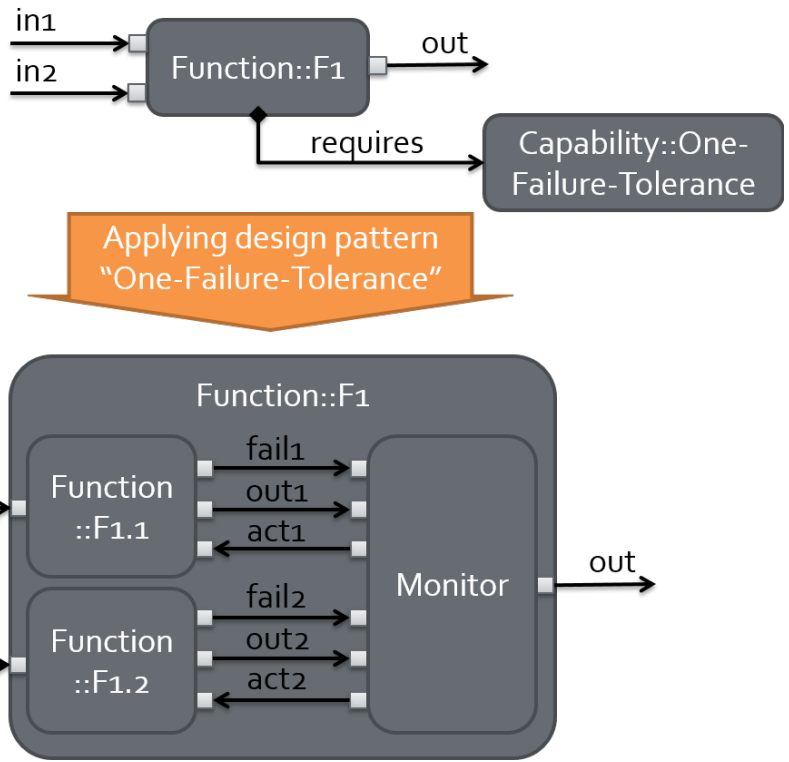
AP3.1 – Workflow und Toolchain



AP3.1 – Methodische und Technische Highlights

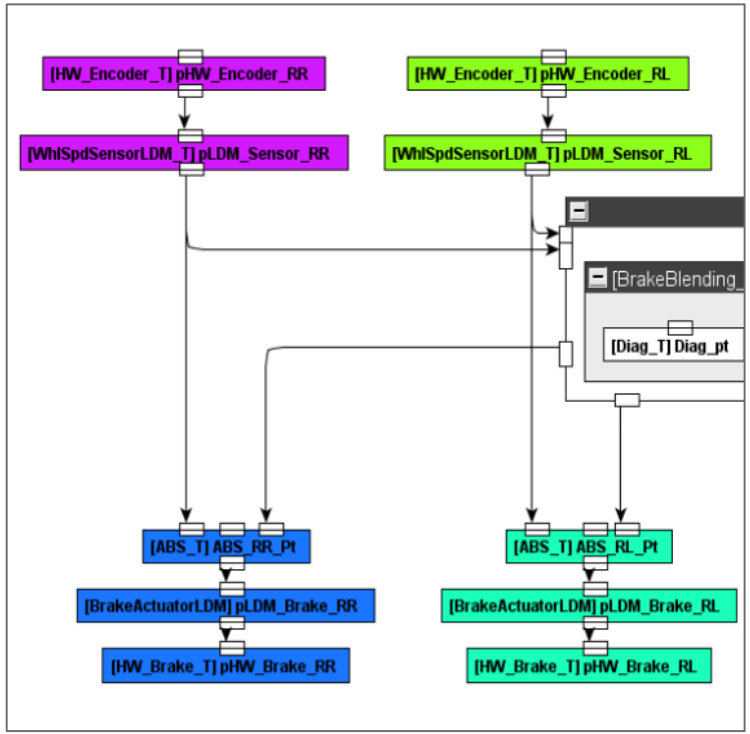
- Design Pattern zur Partitionierung
  - Safety-Decomposition Pattern
  - Allocation Pattern
  - Spezielle AUTOSAR Pattern
  - Anwendung im Design Prozess





AP3.1 – Methodische und Technische Highlights

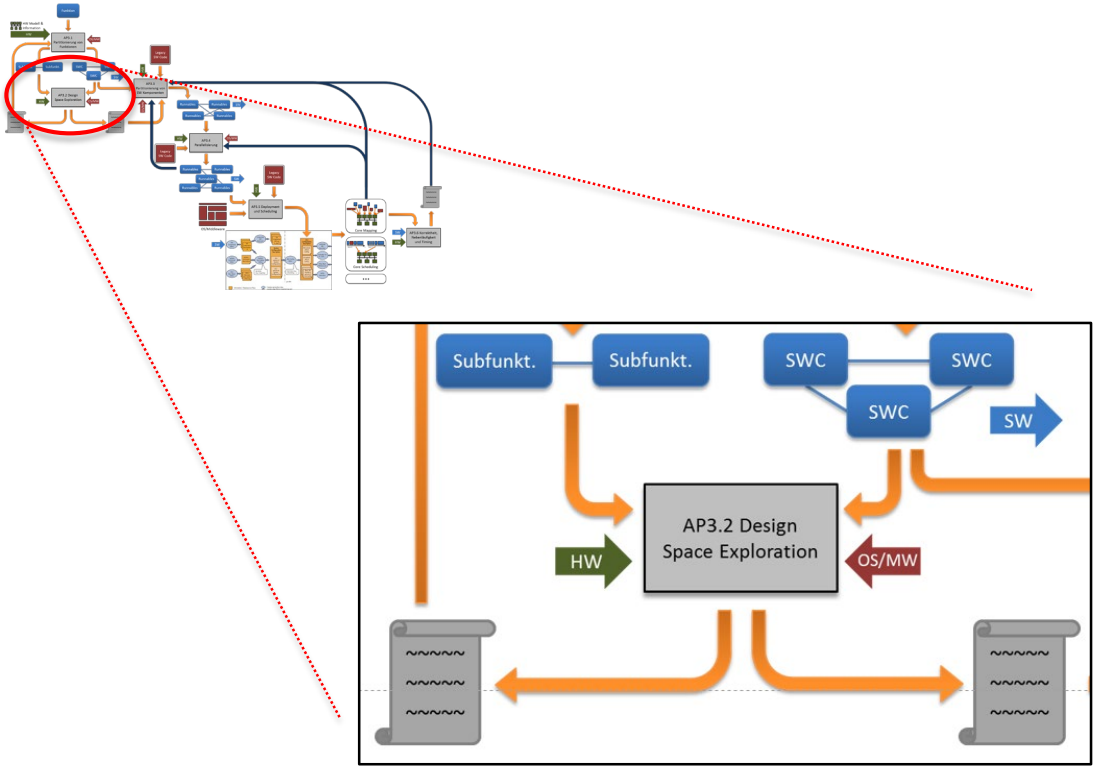
- Partitionierung Funktionaler Architekturen
  - Erweiterung von EAST-ADL um für Partitionierung relevante Elemente
  - Verschiedene Algorithmen zur automatischen Partitionssuche
  - Einbettung in Entwicklungsmethodik



AP3.1 – Zusammenfassung

- Methoden und Tools zur Partitionierung auf funktionalen Architekturen
  - Dekomposition und Partitionierung (Safety-Aspekte, Contracts, Resources, …)
  - Design Pattern
  - Software-Allokation in frühen Entwurfsphasen
  - Legacy Code to Model
  - Prototypische Entwicklung in Tools

AP3.2 – Design Space Exploration

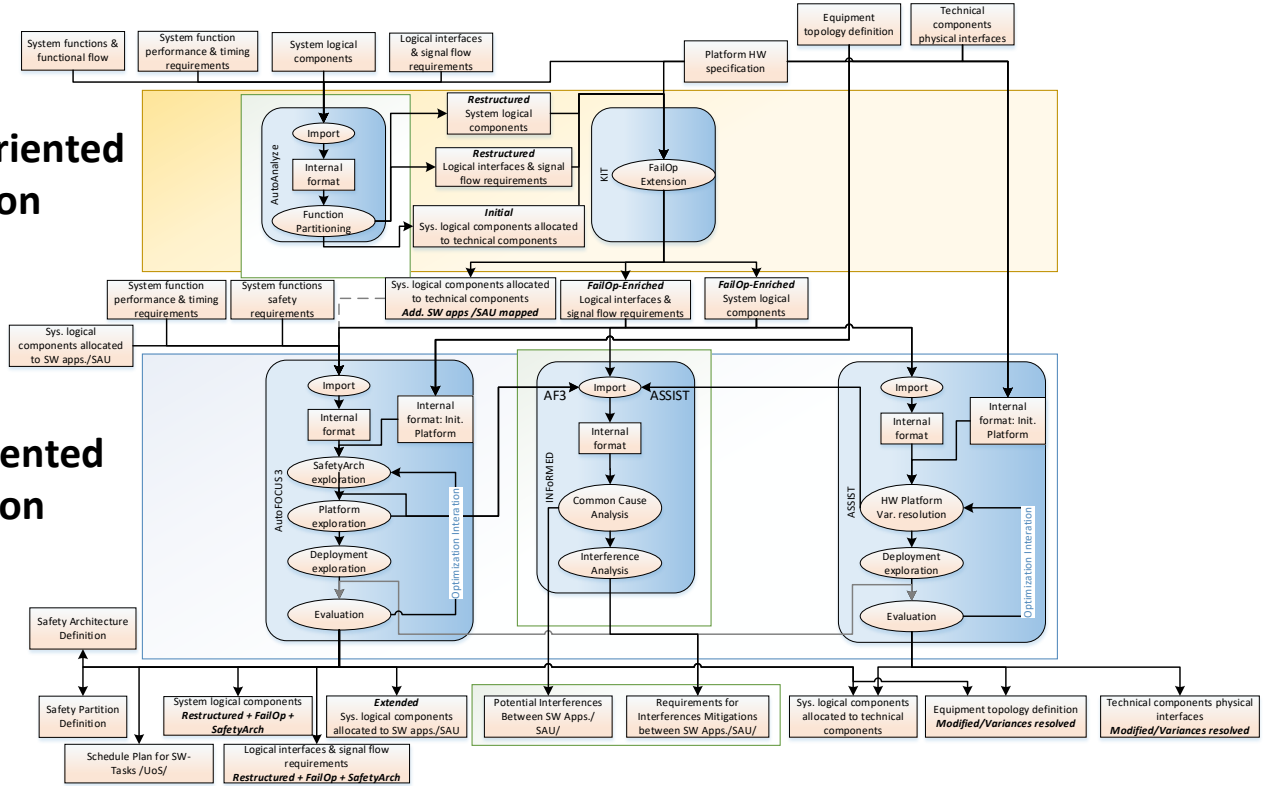


- Synthese
- Integration und Evaluation von DSE-Methoden
- Analyse von Interferenzen

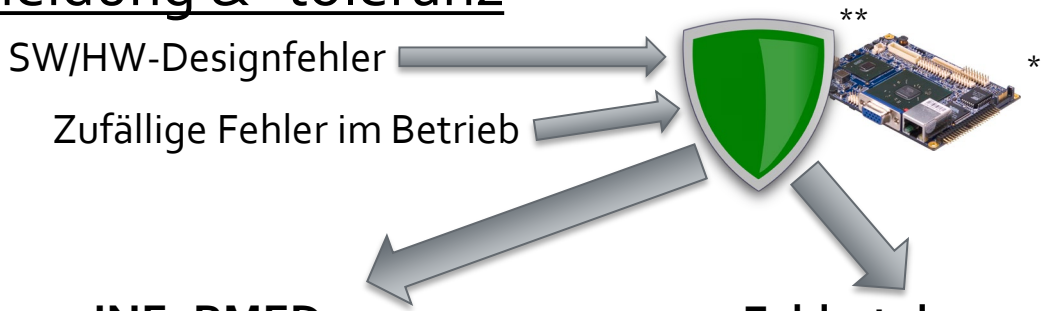
AP3.2 – Workflow und Toolchain

Application-oriented Exploration

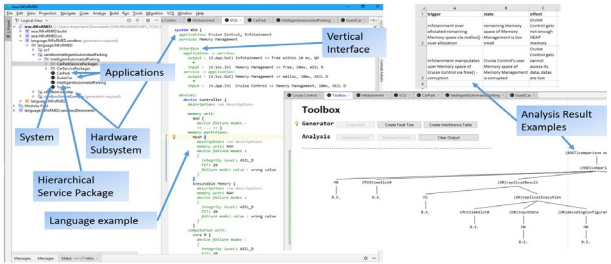
Platform-oriented Exploration



Fehlervermeidung & -toleranz

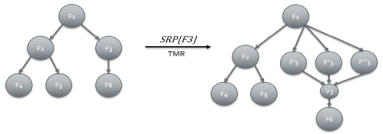


Interferenzen: INFoRMED



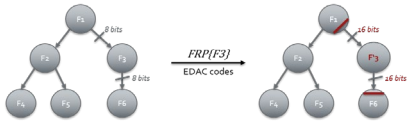
- Basierend auf Service-Modellierung
- Interferenzen: Erkennen & Entfernen im Design

Fehlertoleranz: KIT



Redundanz

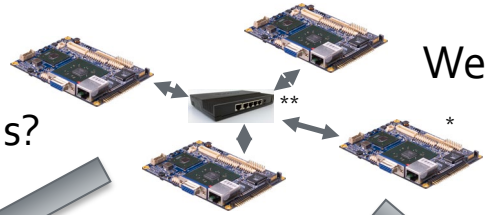
Fehlerkorrektur



- HW/SW-Muster: MTTF erhöhen
- Exploration mit Zuverlässigkeitsanalyse

AP3.2 – Methodische und Technische Highlights

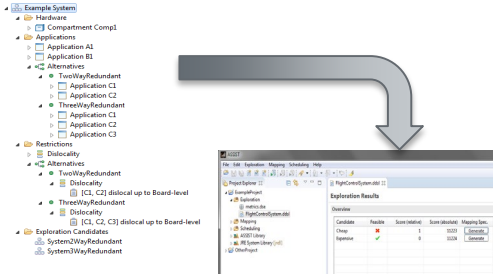
Plattform Exploration



Wie viele Boards oder CPUs?

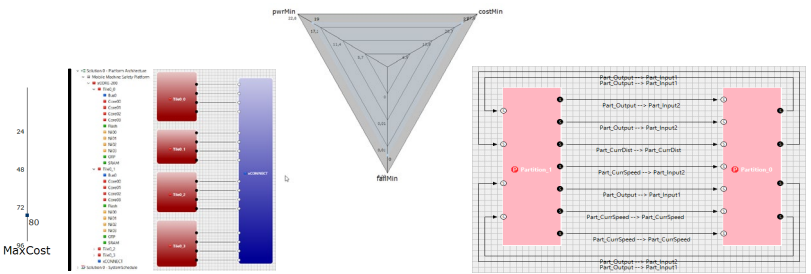
Welche Ausprägung: 2 Cores, 4 Cores?

Produktlinien: ASSIST



- Spezifikation alternativer System-Konfigurationen
- Automatische Bewertung & Task Mappings

HW & Partitionen: AF3 / D³SE



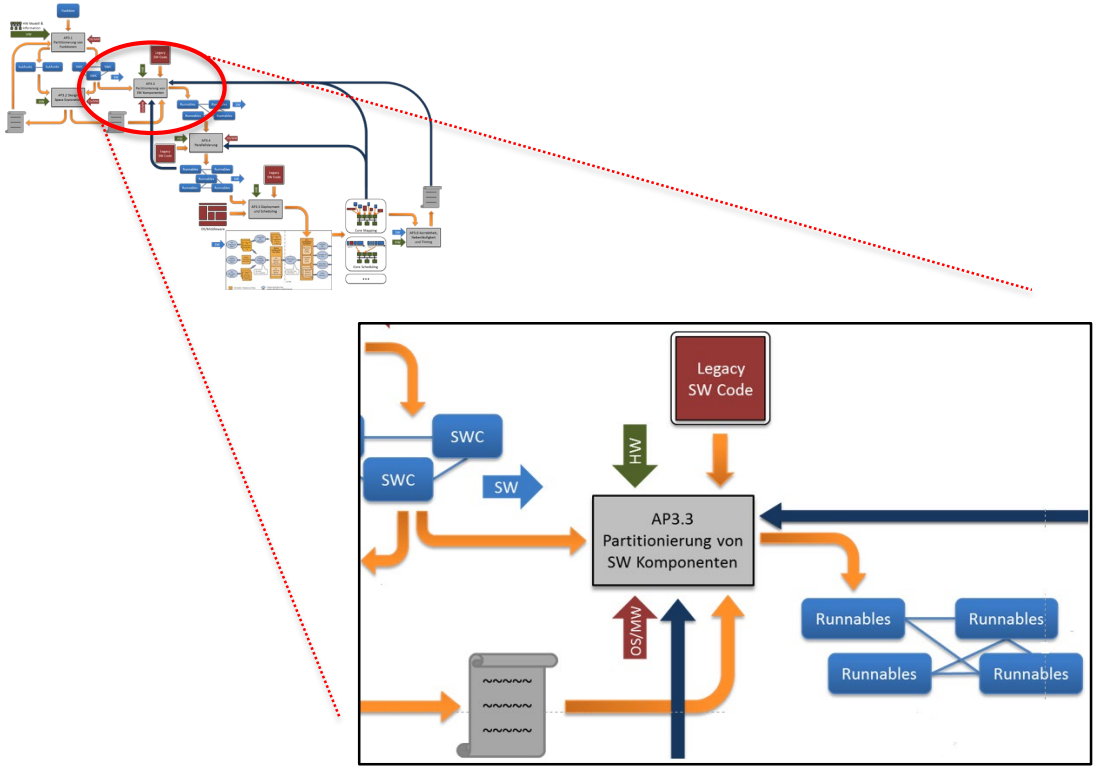
- Erweiterbar & Mehrlagige Mapping-Unterstützung
- Teilweise Integration des AF3-SMT Schedulings

*: ©VIA Technologies Inc
**: © www.heimnetzwerke.net – CC-A

AP3.2 – Zusammenfassung

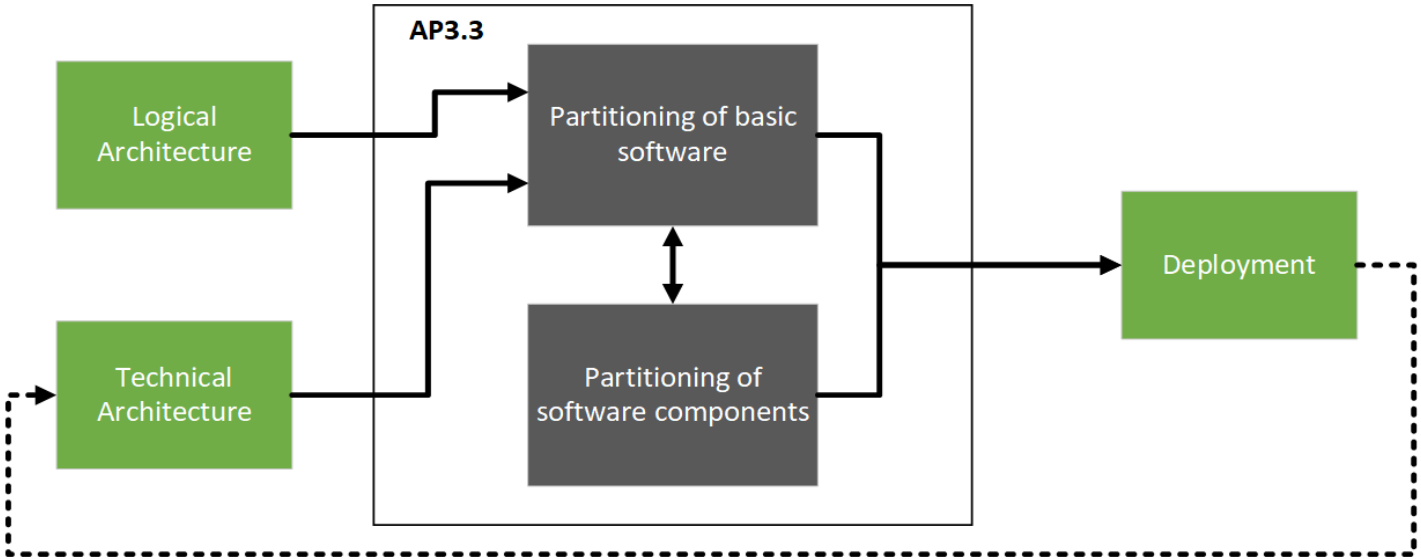
- Vielzahl von Mechanismen und Methoden um die Entwicklung in frühen Entwurfsphasen zu unterstützen
  - Im Besonderen Erfüllung von sicherheitsrelevanten Requirements
  - Verfeinerung des initialen Designs
  - Finden von alternativen logischen oder Plattformarchitekturen
- Fokus auf Besonderheiten von Multicore Architekturen
- Prototypische Entwicklung in Tools sowie Validierung in den Use Cases

AP3.3 – Partitionierung von SW Komponenten



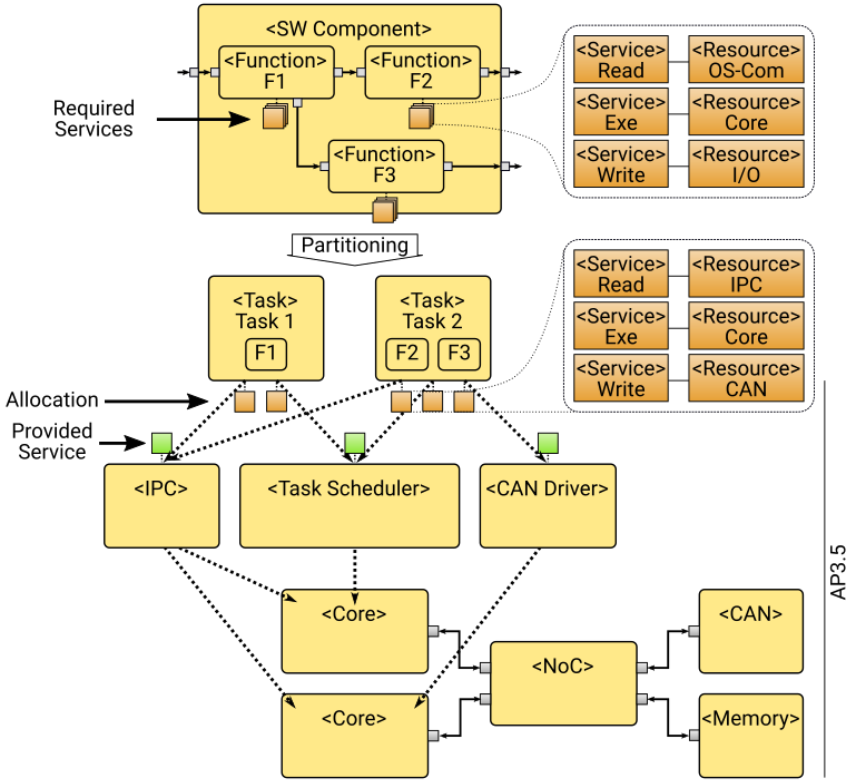
- System-Partitionierung
- Partitionierung auf Multicore-Prozessoren
- Partitionierung der Basissoftware

AP3.3 – Workflow und Toolchain



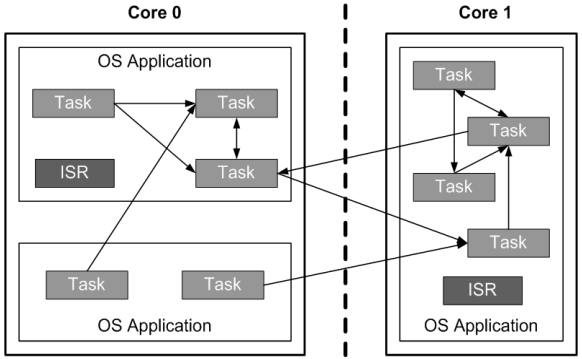
AP3.3 – Methodische und Technische Highlights

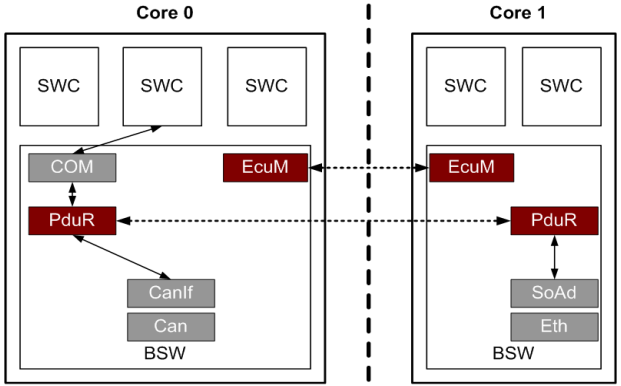
- Modellierungskonzepten der Partitionierung von SW-C
  - Funktionelle Aspekte
  - Aspekte der funktionalen Sicherheit
  - Isolationsmechanismen
  - Automatisierte Optimierungen mit Hilfe der virtuellen Integration



AP3.3 – Methodische und Technische Highlights

- Tool-gestützte Partitionierung von SW Komponenten in der Automotive Domain
  - Visualisierung
  - Analyse des Mappings und der resultierenden Kommunikationskosten
  - Validierung des Mappings sowie bereitgestellter Multicore Mechanismen

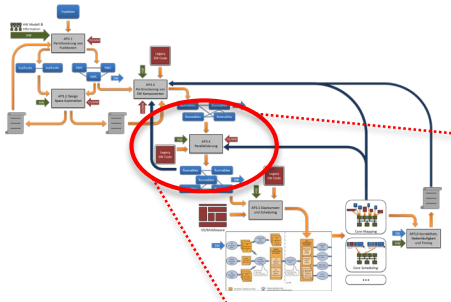


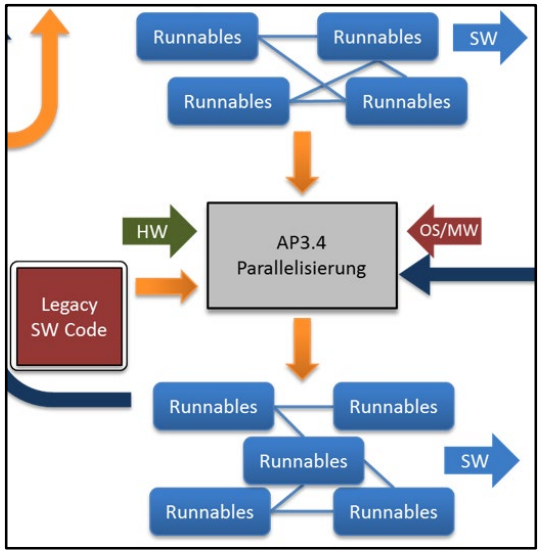


AP3.3 – Zusammenfassung

- Methoden und Tools zur Partitionierung von SW Komponenten auf mehreren Ebenen
  - System Partitionierung: Automatisierte Partitionierung und Parallelisierung von Applikationssoftware auf System Level
  - Multicore Partitionierung: Entwicklung von Modellierungskonzepten; Visualisierung, Analyse und Validierung des AUTOSAR Modells
  - Basissoftware Partitionierung: Entwicklung und Beschreibung von Pattern der Partitionierung von Basissoftware Modulen

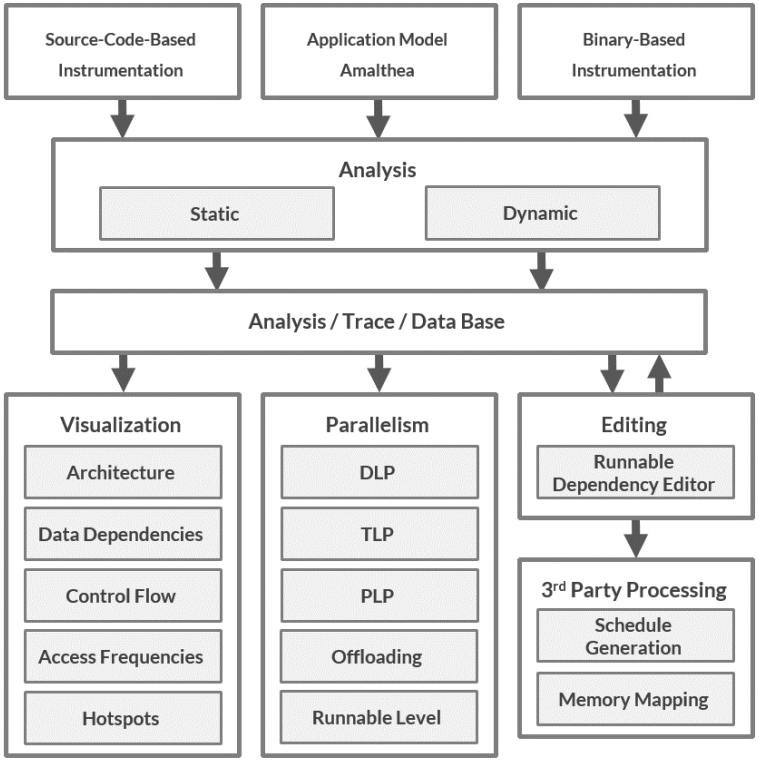
AP3.4 – Parallelisierung





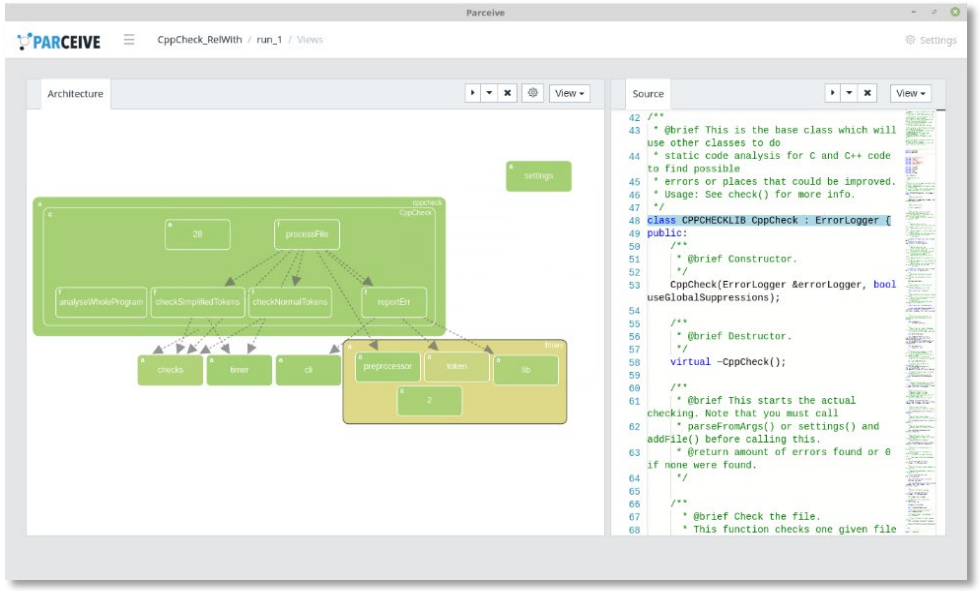
- Shared Data Analyse
- Visualisierung von Shared Data
- Datenaggregation zur Abstraktion
- Bedingte Instrumentierung
- Modellanalyse und Parallelisierung zur Codegenerierung
- Parallelisierung von Legacy C-Code

AP3.4 – Workflow und Toolchain



AP3.4 – Methodische und Technische Highlights

- Visualisierung von Architekturinformationen für die Parallelisierung
  - Flexible Visualisierung
  - Analyse hinsichtlich Parallelisierung durch benutzerdefinierte Regeln

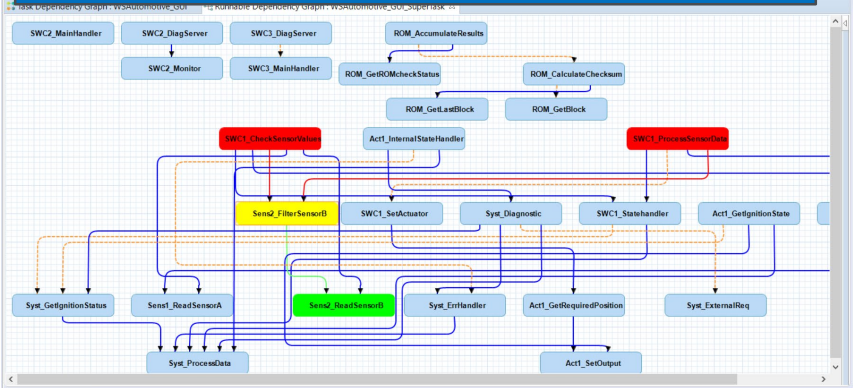


AP3.4 – Methodische und Technische Highlights

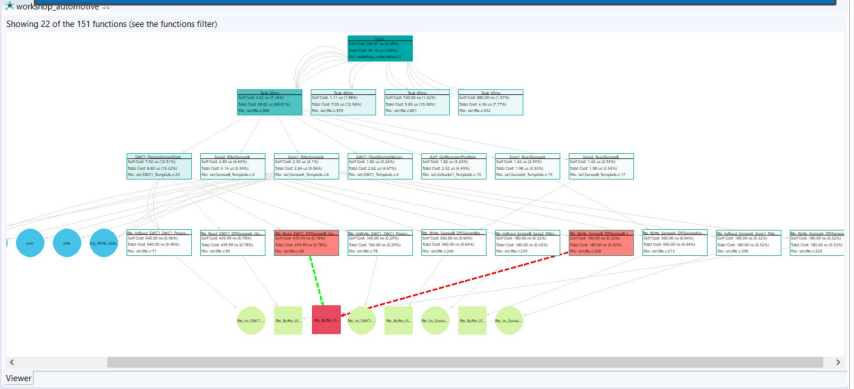
- Parallelisierung auf Runnable Ebene
  - Exploration von Daten Abhängigkeiten durch statische und dynamische Analysen
  - Modellierung von Constraints
  - Generierung eines automatisierten parallelen Schedules
- Einsatz in Use Case von Denso und GE Avionic

AP3.4 – Methodische und Technische Highlights

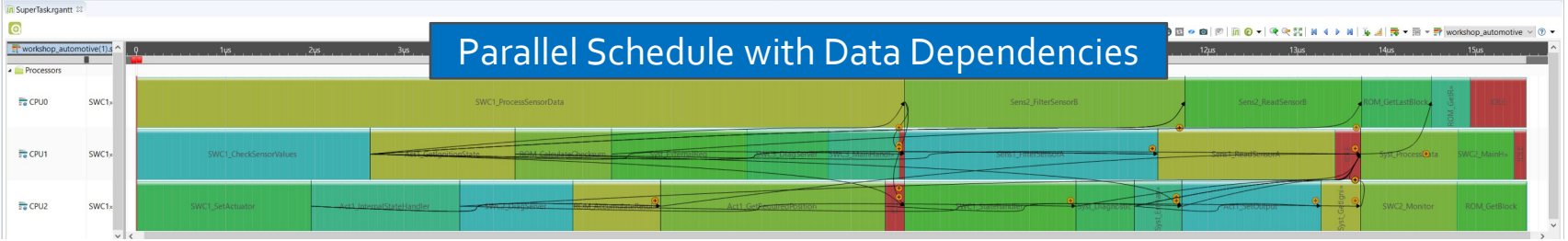
Runnable Dependency Graph



Call Graph with Shared Data



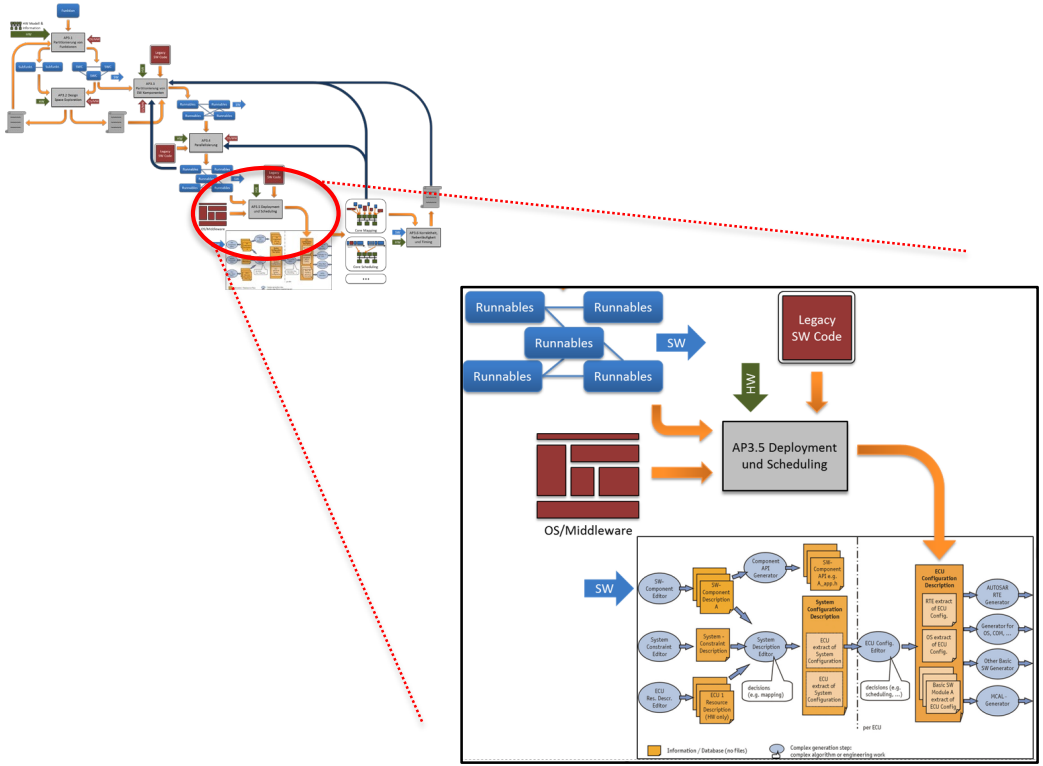
Parallel Schedule with Data Dependencies



AP3.4 – Zusammenfassung

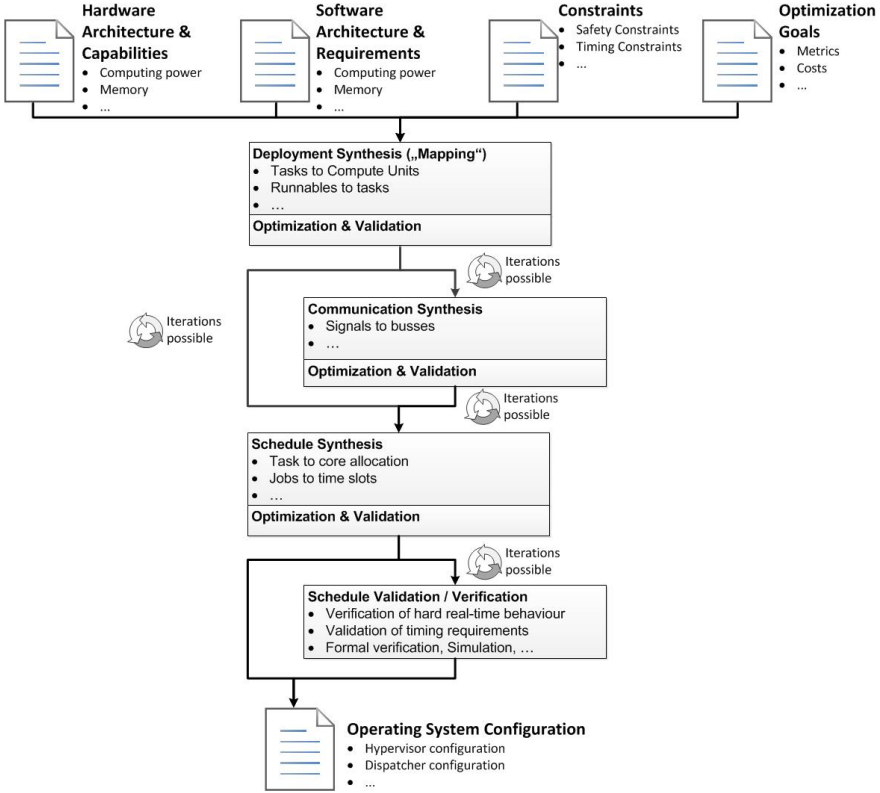
- Tools wurden um Mechanismen der Datenanalyse und Datenvisualisierung erweitert
- Bedingte Instrumentierung und Verbesserung der statischen Analyse wurden entwickelt, um Softwareanalyse zu beschleunigen
- Verbesserung der Interoperabilität der Tools
- Erweiterung der Parallelisierung zur Erstellung von Multicore Schedules auf Basis von C-Funktionen
- Erfolgreiche Implementierung der Methoden in Use Cases

AP3.5 – Deployment und Scheduling



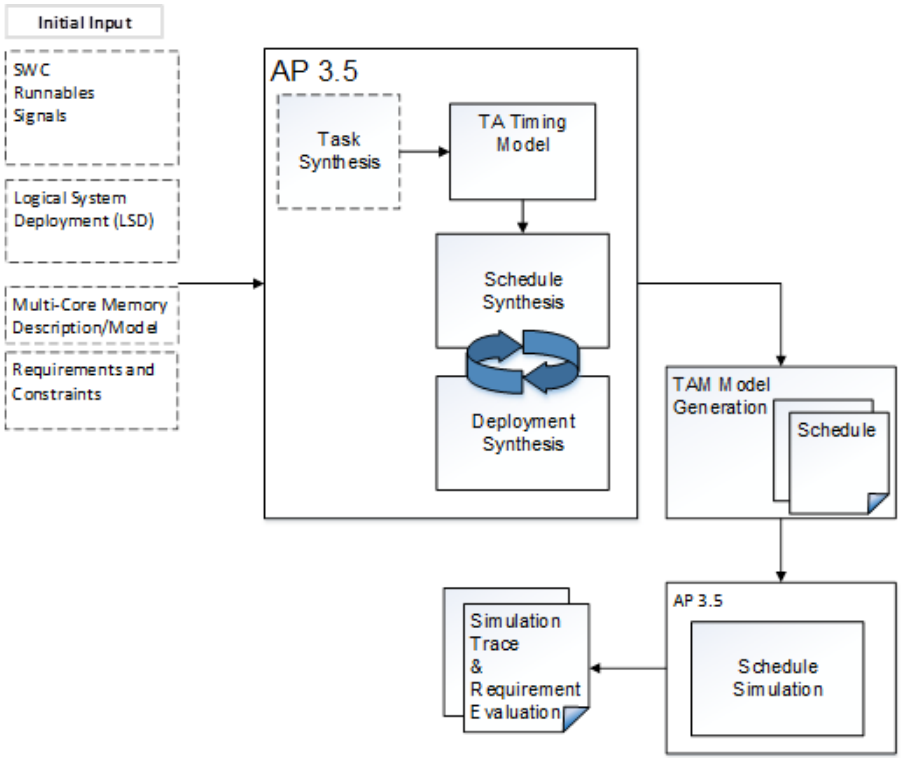
- Deployment Synthesis
- Schedule Synthesis
- Communication Synthesis
- Schedule Analysis
- Optimierungs-techniken

AP3.5 – Workflow und Toolchain



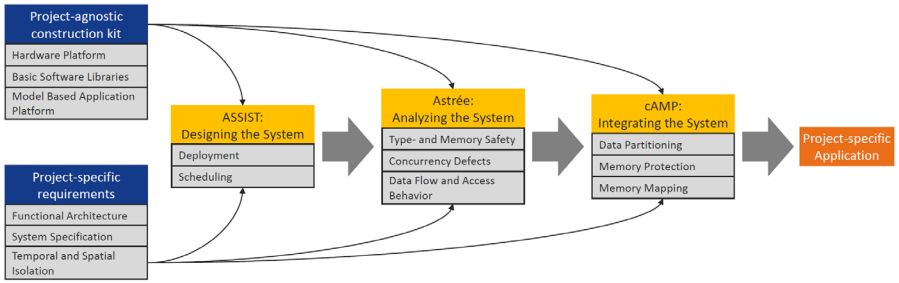
AP3.5 – Methodische und Technische Highlights

- Kombinierte Deployment und Scheduling Synthese
  - Berücksichtigung von Multicore Speicherarchitekturen sowie Requirements und Constraints
  - Einbeziehung von Logical Execution Times
  - Evaluation anhand Simulation Traces
- Einsatz im Avionik Use Case



AP3.5 – Methodische und Technische Highlights

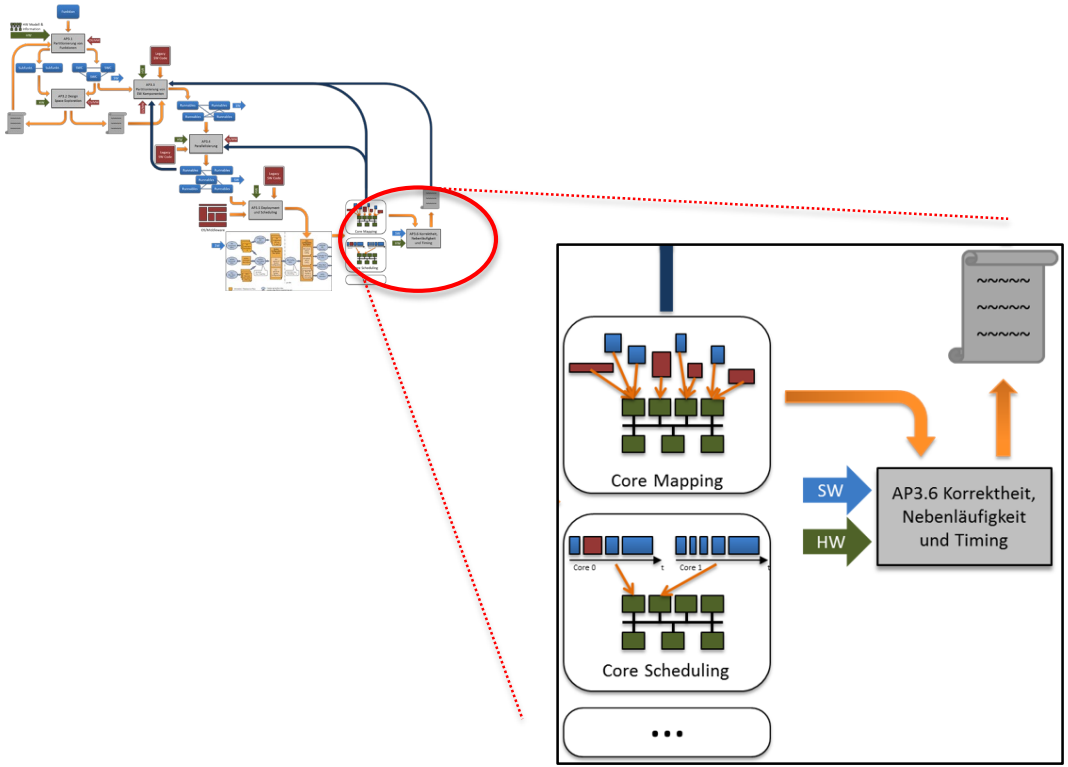
- Korrektes und optimales Memory Mapping
  - Platzierung von Variablen in verschiedenen Speicherbereichen einer Multicore Architektur
  - Toolchain aus ASSIST (DLR), Astrée (Absint) und cAMP (LuK)
- Validierung im Use Case von Luk - Motorsteuerung



AP3.5 – Zusammenfassung

- Methoden und Tools zur Erstellung und Analyse des Deployment und Schedulings
  - Beschreibung des Ziels sowie der benötigten Eingangsparameter
  - Detaillierter Vergleich der Software Tools
- Verfeinerung der zugrunde liegenden Methodiken und Modellierungsansätzen der Tools
- Validierung der Methoden und Tools in den Use Cases der verschiedenen Domänen

AP3.6 – Korrektheit, Nebenläufigkeit und Timing

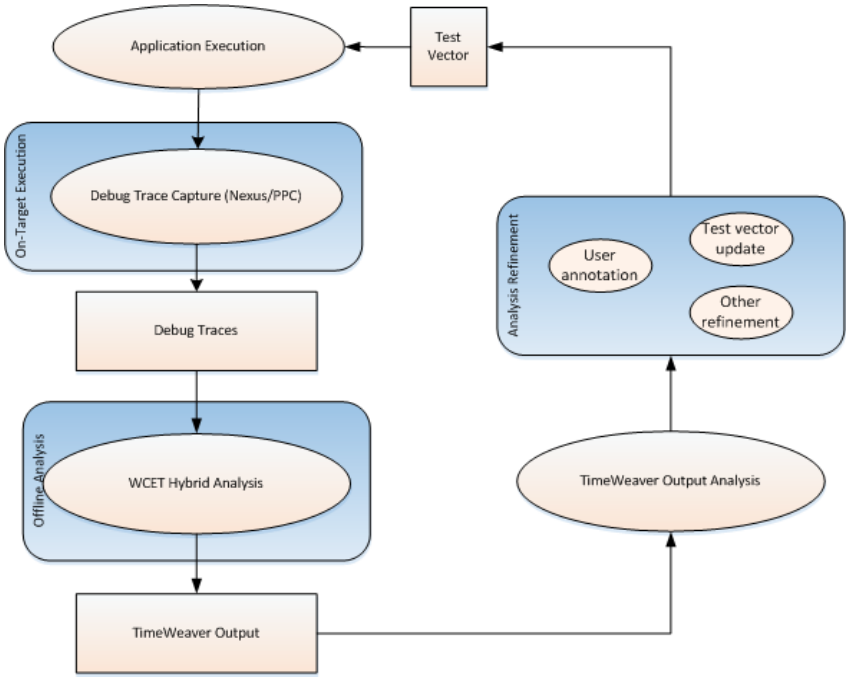


- Nebenläufigkeit, Timing-Analyse und Scheduling
- Performance-Analyse und Optimierung
- Statische Analyse und Testing
- Fault-Erkennung und -Korrektur
- Safety-Herausforderungen
- Tools zur Verifikation von partitionierenden Hypervisor
- Debugging, Integration und Test

AP3.6 – Workflow und Toolchain

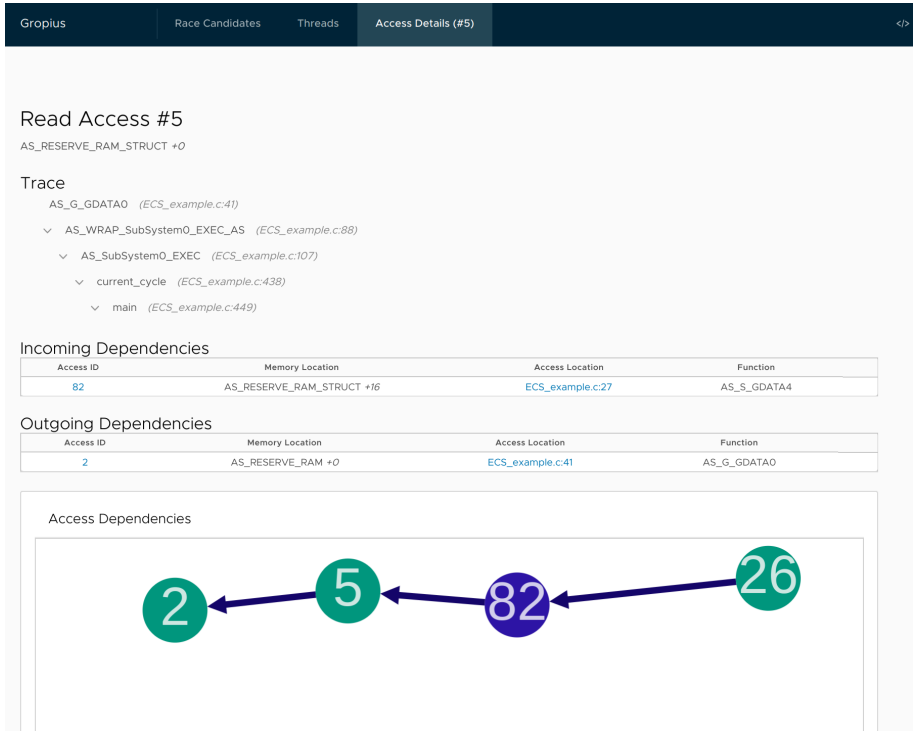
AP3.6 – Methodische und Technische Highlights

- Hybride Timing Analyse
  - Kombiniert statische und dynamische Analysen
  - Bestimmung von engen oberen Grenzen der WCET
- Einsatz unter anderem im Use Case von GE



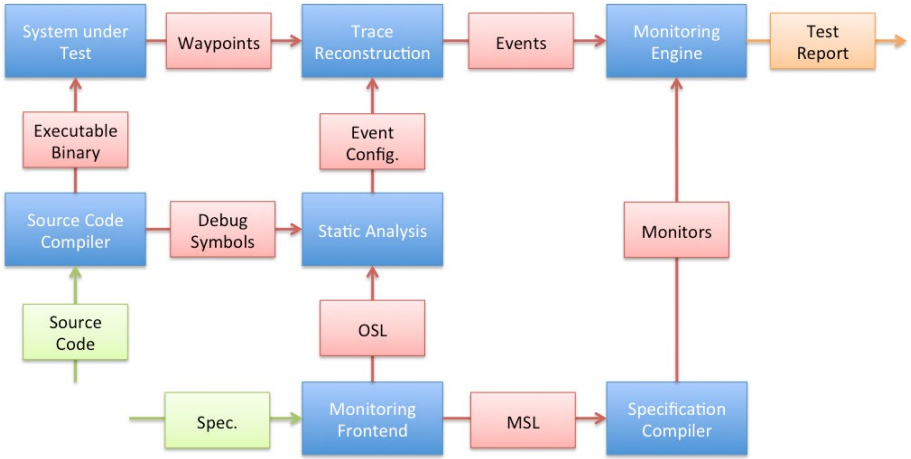
AP3.6 – Methodische und Technische Highlights

- Lokalisieren von Problemen durch Nebenläufigkeit
  - Statische Analyse von Gropius
  - Race Conditions im nebenläufigen Automotive RTE Code
  - Wiederherstellen von Datenabhängigkeiten zwischen Subsystemen
- Validierung der Methoden im Use Case von AUDI und GE



AP3.6 – Methodische und Technische Highlights

- Trace-basiertes Debugging und Runtime Verifikation
  - Verwendung von Embedded Tracing Units
  - Rekonstruierung des Kontrollflusses ohne Interferenzen mit SuT
  - Non-intrusive Laufzeit Verifikation, Fehleranalyse, Fehlerinjektion und Tracing
- Validierung in Avionik Use Cases

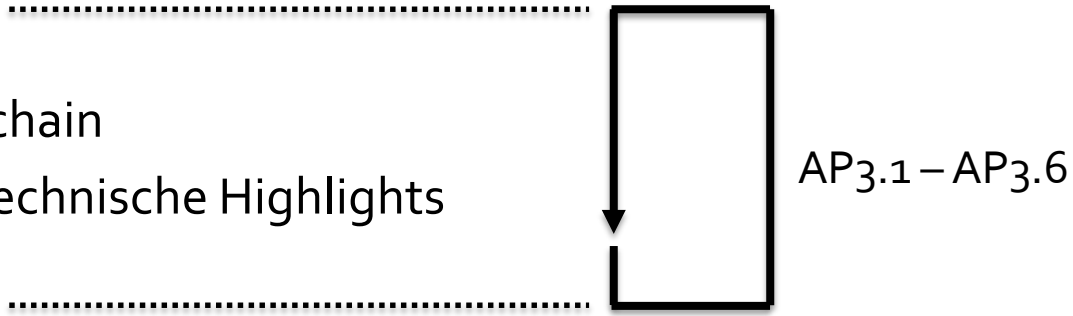


AP3.6 – Zusammenfassung

- Adressierung der Korrektheit, Nebenläufigkeit und Timing Herausforderungen bei Multicore Systemen
  - Einsatz von statischen und dynamischen Analysen
- Verfeinerung der Methodiken der Tools sowie Validierung in Use Cases der verschiedenen Domänen
- Beschreibung der In/Output Artefakte sowie Aktivitäten der Tools
  - Implementierung von geeigneten Toolchains

Agenda

- Motivation und Vorgehen
- Arbeitspakete
  - Ziele
  - Workflow und Toolchain
  - Methodische und Technische Highlights
  - Zusammenfassung



- **Ergebnisse TP3**

Ergebnisse TP3

Multicore Methoden und Tools

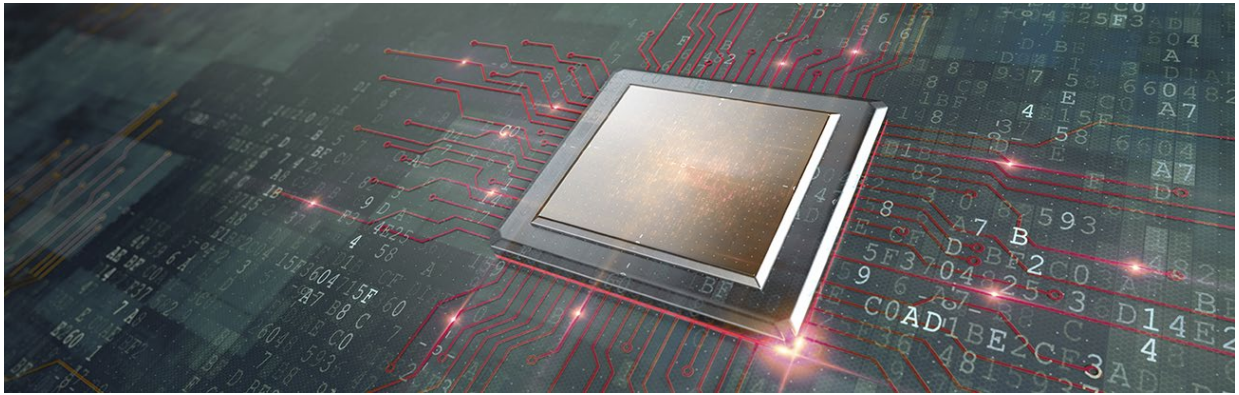
- Entwicklung spezifischer Methoden und Werkzeuge zur Unterstützung der Multicore-Entwicklung
- Erweiterung der Methoden für alle Schritte im Entwicklungsprozess (z.B. Partitionierung, Deployment, Scheduling)
- Höherer Automatisierungsgrad in der Entwicklung durch Werkzeugunterstützung

Ergebnisse TP3

Multicore Methoden und Tools

- Einsatz eines durchgängigen gemeinsamen Metamodells – AMALTHEA
- Entwicklung von Domänen-übergreifenden Methodiken und Tools für Multicore Architekturen
- ARAMiS II Ergebnisdokumente stellen für Entwickler einen „Tool Baukasten" zur Verfügung

Ergebnisse TP3



Die in TP3 entwickelten Methoden und Tools gewährleisten die Entwicklung von Software für Multicore Architekturen im sicherheitskritischen Umfeld.