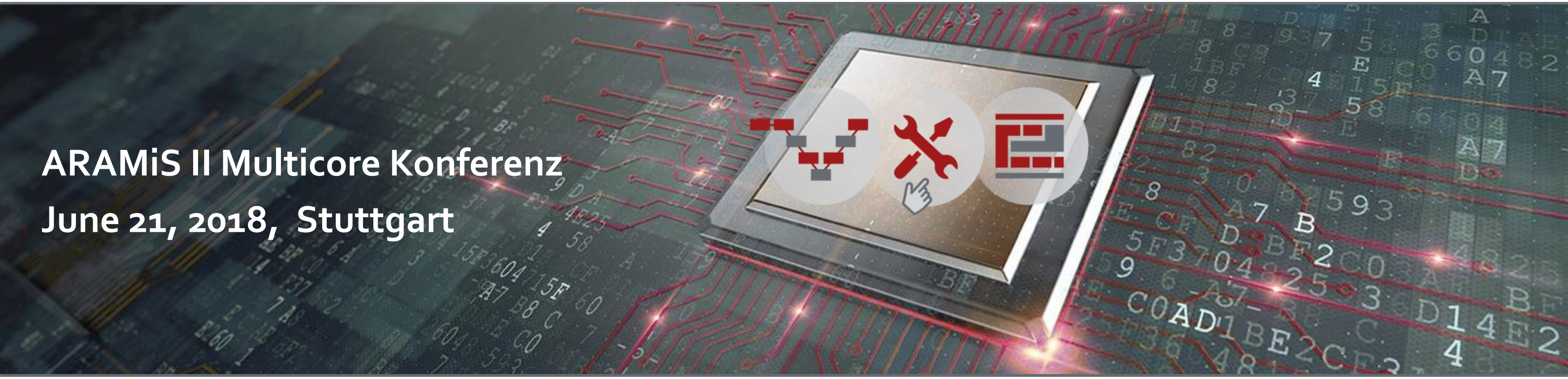
SCHAEFFLER

aramis II

DEVELOPMENT PROCESSES I TOOLS I PLATFORMS
FOR SAFETY-CRITICAL MULTICORE SYSTEMS



E-Mobility goes Multicore

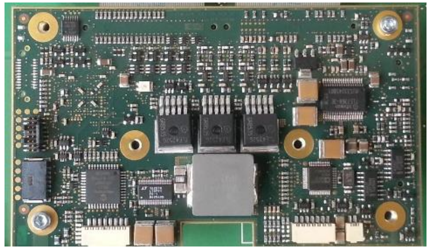
Building Generic Motor-Control Systems by Architectural Design Methods and Static Analysis

Arnd Leitner, LuK GmbH & Co. KG (Schaeffler Gruppe), Bühl

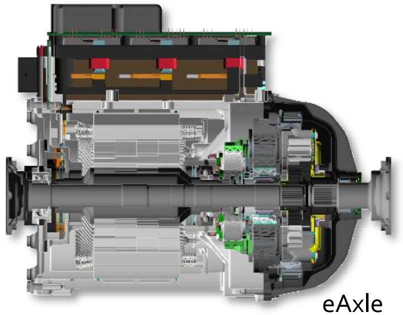
SPONSORED BY THE
Federal Ministry
of Education
and Research

Background and Use Case

- Schaeffler develops different E-mobility products based on a common HW platform and motor control library
- AUTOSAR architecture
- Model-based development (Matlab/Simulink, Targetlink)
- High requirements on functional safety (ASIL-C/D)



Hybrid module


eAxle

E-Wheel Drive

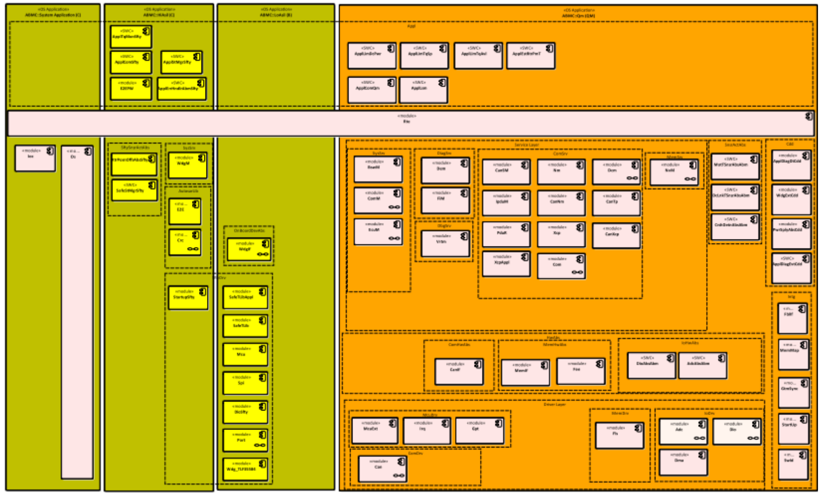
Challenges

- Short development time for new e-drive projects
- Motor control library developed fully generic and HW-agnostic
- Projects choose different HW (µC) variants and have specific requirements on safety, i.e. spatial and temporal isolation
- Add-on SW functions vary from project to project, e.g.
  - >1 motor control
  - OEM SW
  - additional safety supervisions
  - other functions

DEVELOPMENT PROCESSES I TOOLS I PLATFORMS
FOR SAFETY-CRITICAL MULTICORE SYSTEMS

Current Approach

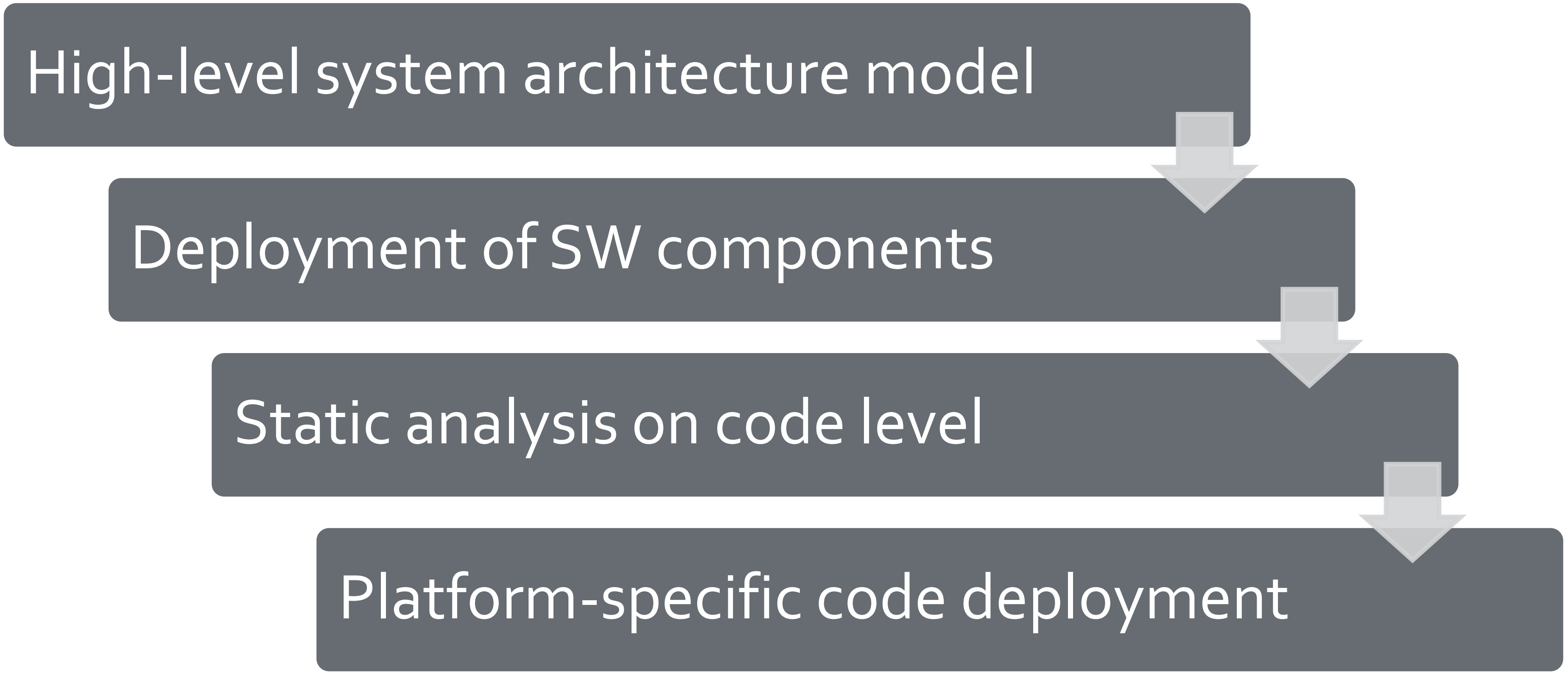
1. Manual distribution and architecture design
2. Implementation
3. Measurements of resource consumptions



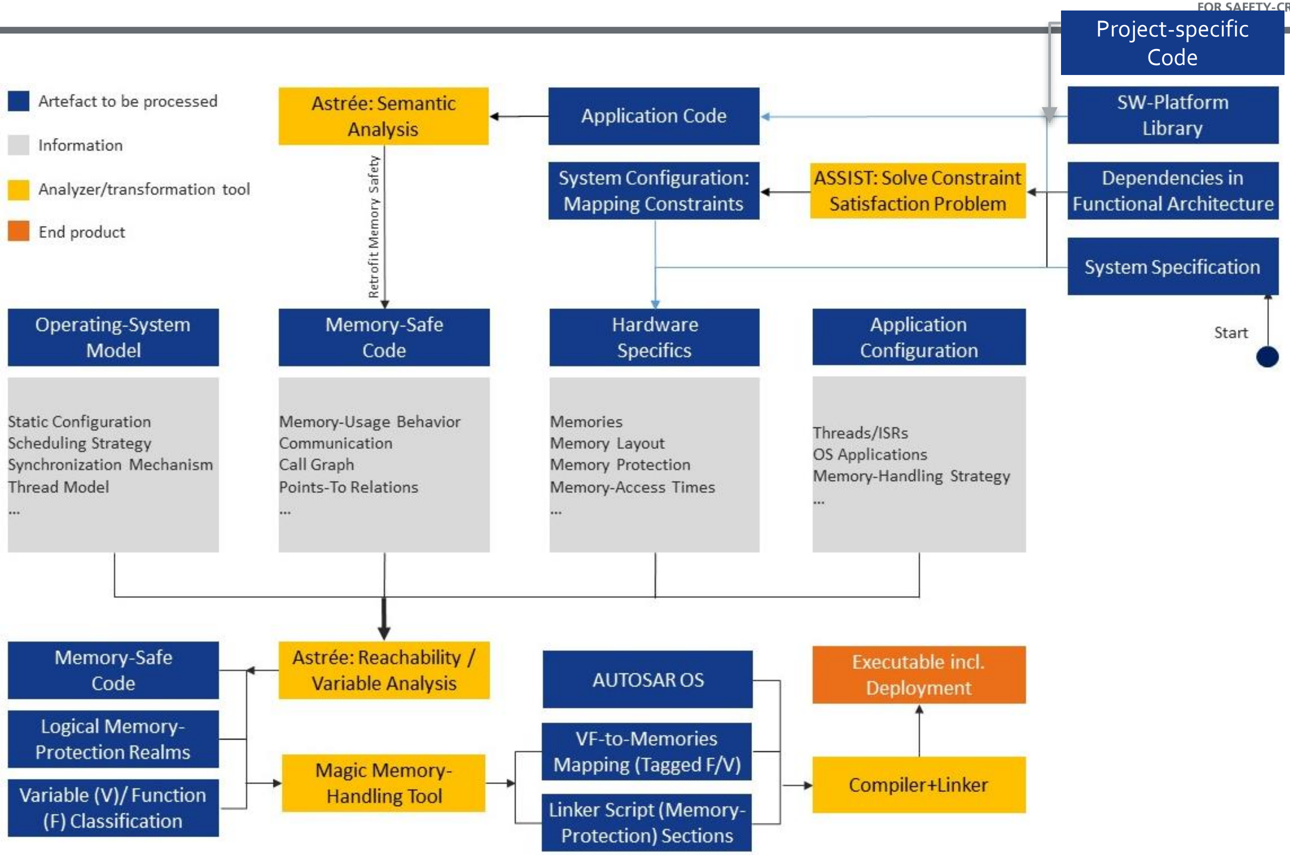
We have a Dream…

- Good estimate of the performance of the system already in the early design phase
- Have a tool set to optimize the design without the need to do implementation cycles

General Approach


High-level system architecture model
Deployment of SW components
Static analysis on code level
Platform-specific code deployment

Concept for Tool Flow



High-level System Architecture Model

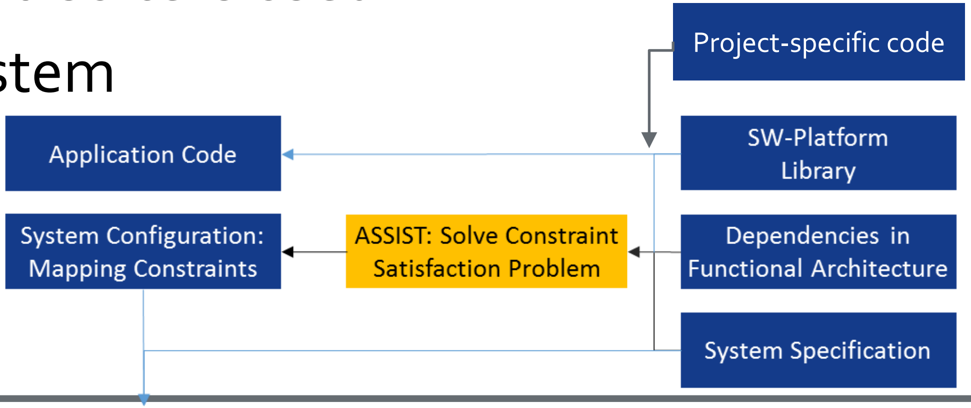
- Available HW-resources
- Description of SW-components with their resource requirements
- Safety requirements (isolation requirements)

- Model based on AMALTHEA metamodel

Deployment of SW Components

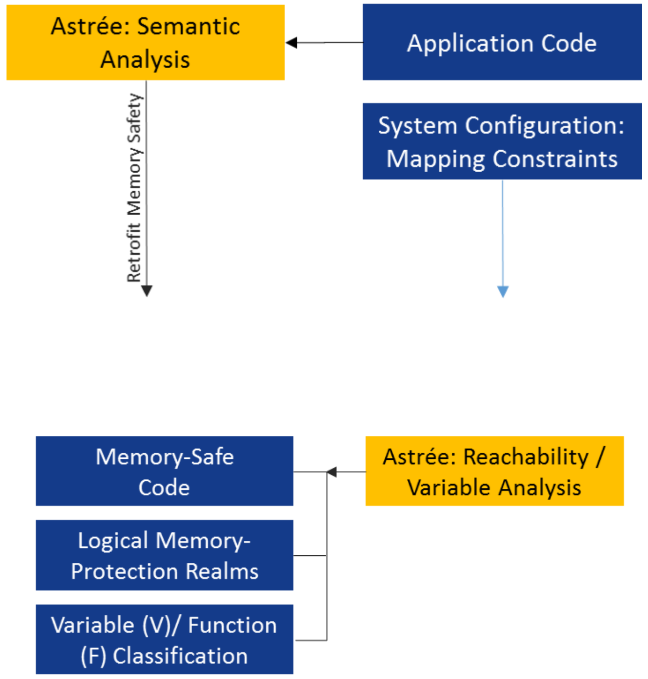
- Determine if the available HW resources are able to satisfy the resource and safety requirements of the SW components
- Mapping and scheduling for all SW components is constructed

- Constraint solver of ASSIST toolsuite is used
- Results can be exported as system configuration files


Project-specific code
Application Code
SW-Platform Library
System Configuration: Mapping Constraints
ASSIST: Solve Constraint Satisfaction Problem
Dependencies in Functional Architecture
System Specification

Static Analysis on Code Level

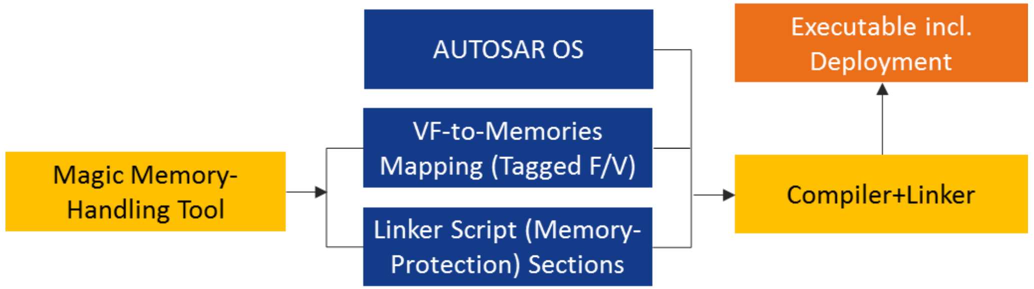
- Avoid memory violations by applying a static code anlysis tool (here: AbsInt Astrée)
- Variable and data structures are classified
  - function-local data
  - thread-local data
  - thread-global data
  - core-local data
  - core-global data.
- With this input and the system architecture and configuration, memory protection realms are created


Astrée: Semantic Analysis
Application Code
System Configuration: Mapping Constraints
Retrofit Memory Safety
Memory-Safe Code
Logical Memory-Protection Realms
Variable (V)/ Function (F) Classification
Astrée: Reachability / Variable Analysis

Platform-specific Code Deployment

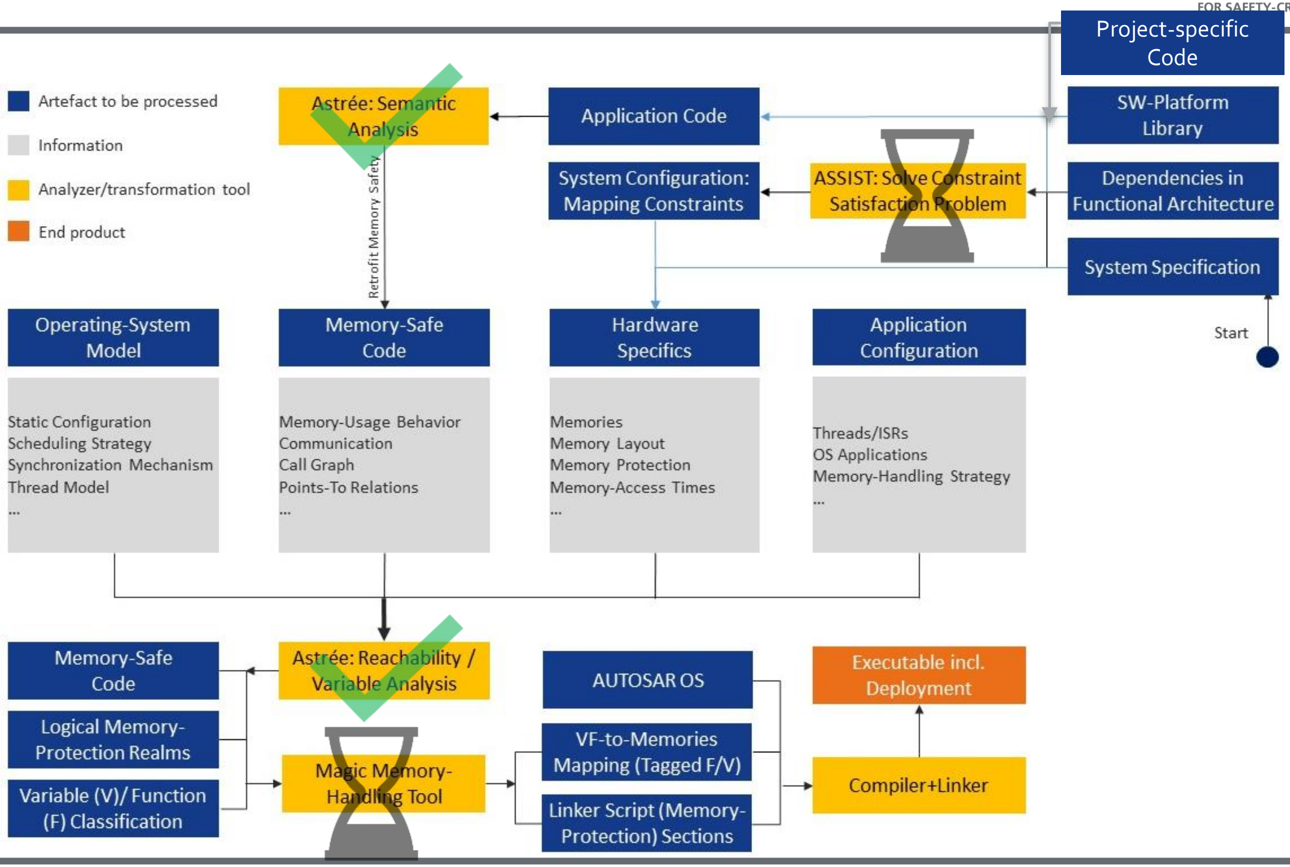
- Isolated "OS applications" which are separated by a Memory Protection Unit (MPU) are defined
- Now all variables can automatically be mapped to physical memories

- Final result is a memory-optimized deployment and binary code


AUTOSAR OS
Executable incl. Deployment
VF-to-Memories Mapping (Tagged F/V)
Magic Memory-Handling Tool
Compiler+Linker
Linker Script (Memory-Protection) Sections

Status

DEVELOPMENT PROCESSES I TOOLS I PLATFORMS
FOR SAFETY-CRITICAL MULTICORE SYSTEMS



Project-specific Code
■ Artefact to be processed
■ Information
■ Analyzer/transformation tool
■ End product
Astrée: Semantic Analysis
Application Code
SW-Platform Library
System Configuration: Mapping Constraints
ASSIST: Solve Constraint Satisfaction Problem
Dependencies in Functional Architecture
Retrofit Memory Safety
System Specification
Start
Operating-System Model
Memory-Safe Code
Hardware Specifics
Application Configuration
Static Configuration
Scheduling Strategy
Synchronization Mechanism
Thread Model
...
Memory-Usage Behavior
Communication
Call Graph
Points-To Relations
...
Memories
Memory Layout
Memory Protection
Memory-Access Times
...
Threads/ISRs
OS Applications
Memory-Handling Strategy
...
Memory-Safe Code
Astrée: Reachability / Variable Analysis
AUTOSAR OS
Executable incl. Deployment
Logical Memory-Protection Realms
Magic Memory-Handling Tool
VF-to-Memories Mapping (Tagged F/V)
Compiler+Linker
Variable (V)/ Function (F) Classification
Linker Script (Memory-Protection) Sections

Benefits of the Concept

- It enables hardware-independent software development via generic application libraries (“Motor Control Platform").
- It forces application designers into stating explicit requirements regarding temporal and spatial resource partitioning (assignment of CPU-time and memory-usage budgets).
- Freedom from interference concepts can be included in the partitioning analyses.
- Semi-automated spatial and temporal isolation by semantic program analyses is possible.

Next Steps

- Extension of the used tools is ongoing
- Tool flow will be stepwise applied to the automotive E-drive use case

- Timing and WCET issues will be tackled next
  - AbsInt's TimeWeaver (amongst others) will be employed

aramis II

ENTWICKLUNGSPROZESSE I WERKZEUGE I PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME

STRUCTURED MULTICORE DEVELOPMENT

MULTICORE METHODS AND TOOLS

INDUSTRIAL PLATFORMS FOR MULTICORE SYSTEMS

Thank you for your attention!

Arnd Leitner
LuK GmbH & Co KG