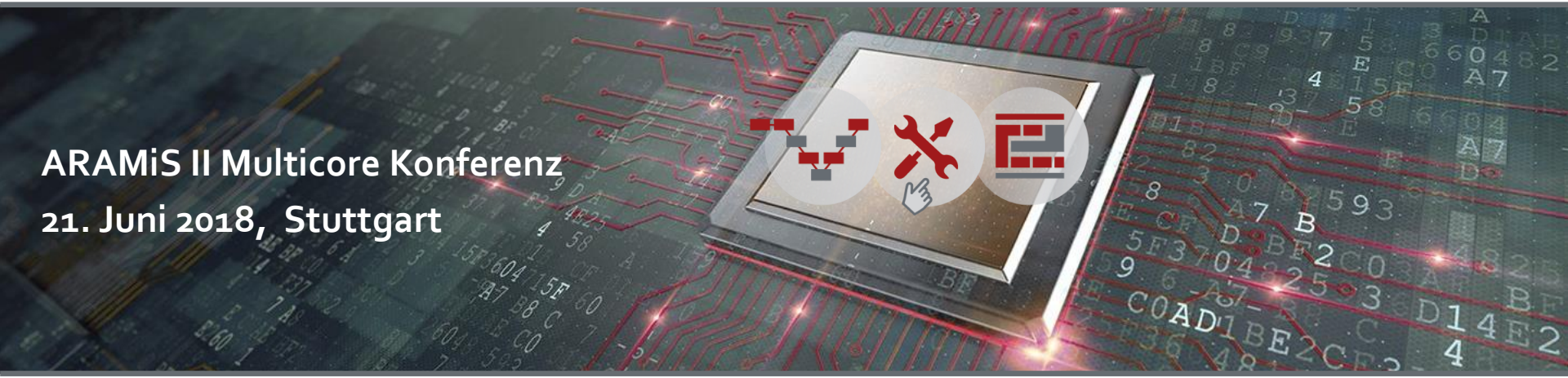
ENTWICKLUNGSPROZESSE I WERKZEUGE I PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME



ARAMiS II Multicore Konferenz
21. Juni 2018, Stuttgart

Multicore Softwareentwicklung

Die ARAMiS Toolchains

Bernhard Bauer
Universität Augsburg

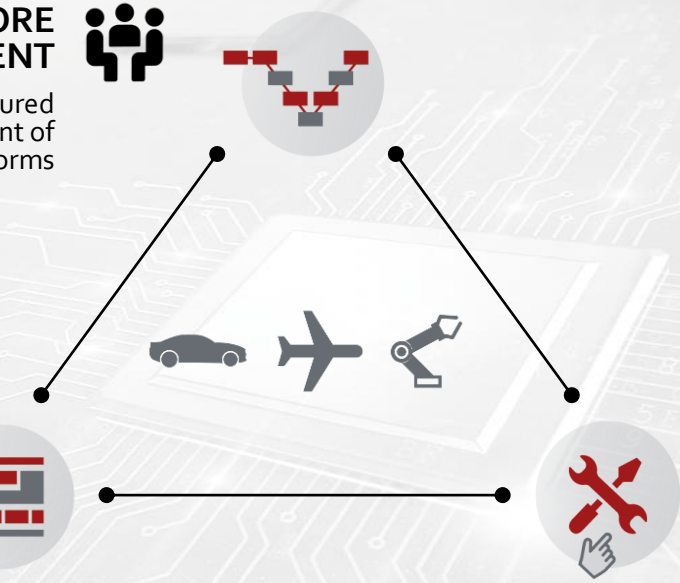
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

ARAMiS as a whole

ENTWICKLUNGSPROZESSE I WERKZEUGE I PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME

STRUCTURED MULTICORE DEVELOPMENT

Provision of systematical and structured approaches for the development of multicore software and platforms



INDUSTRIAL PLATFORM ARCHITECTURES

Development and extension of established industrial platforms with respect to multicore requirements

METHODS AND TOOLS

Development of methods and tools supporting the structured multicore development

TP3 Multicore Methods and Tools

Multicore Methods and Tools

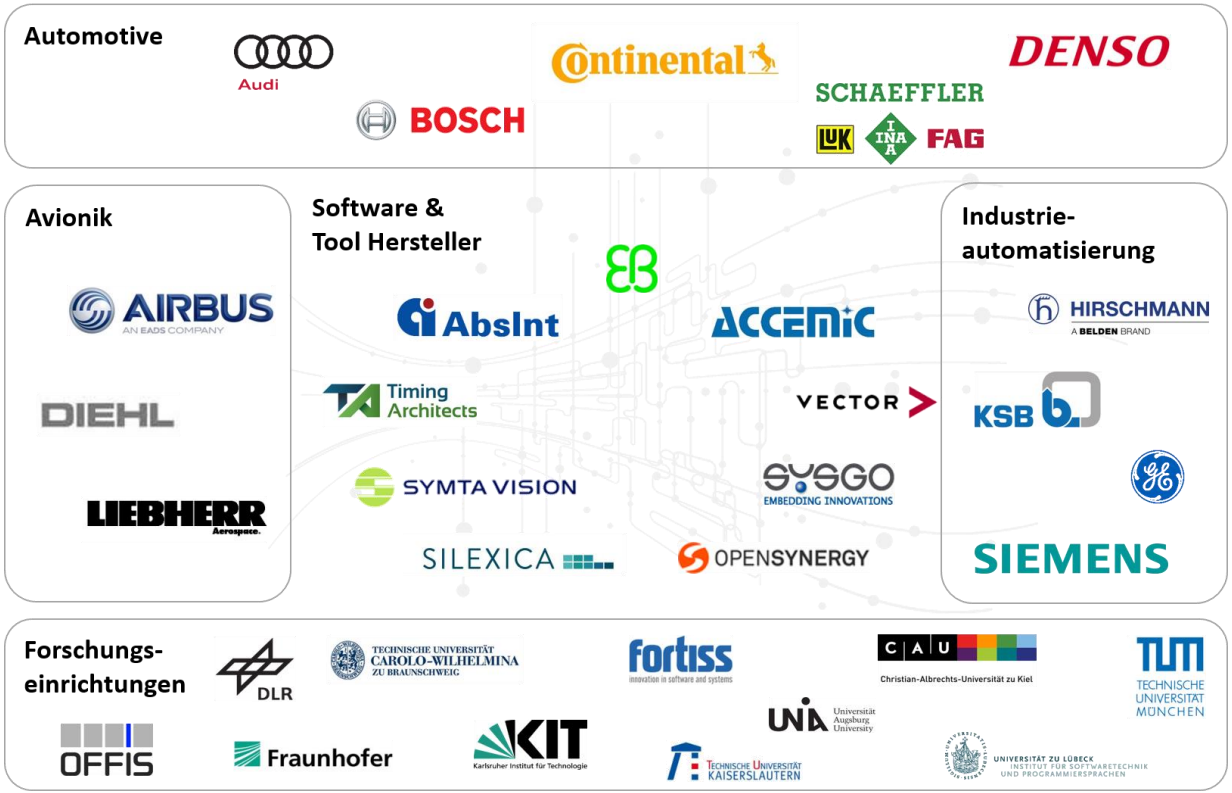
Development of specific methods and tools to support the structured multicore development

Extension of methods for all steps in the development process (e.g. partitioning, deployment, scheduling)

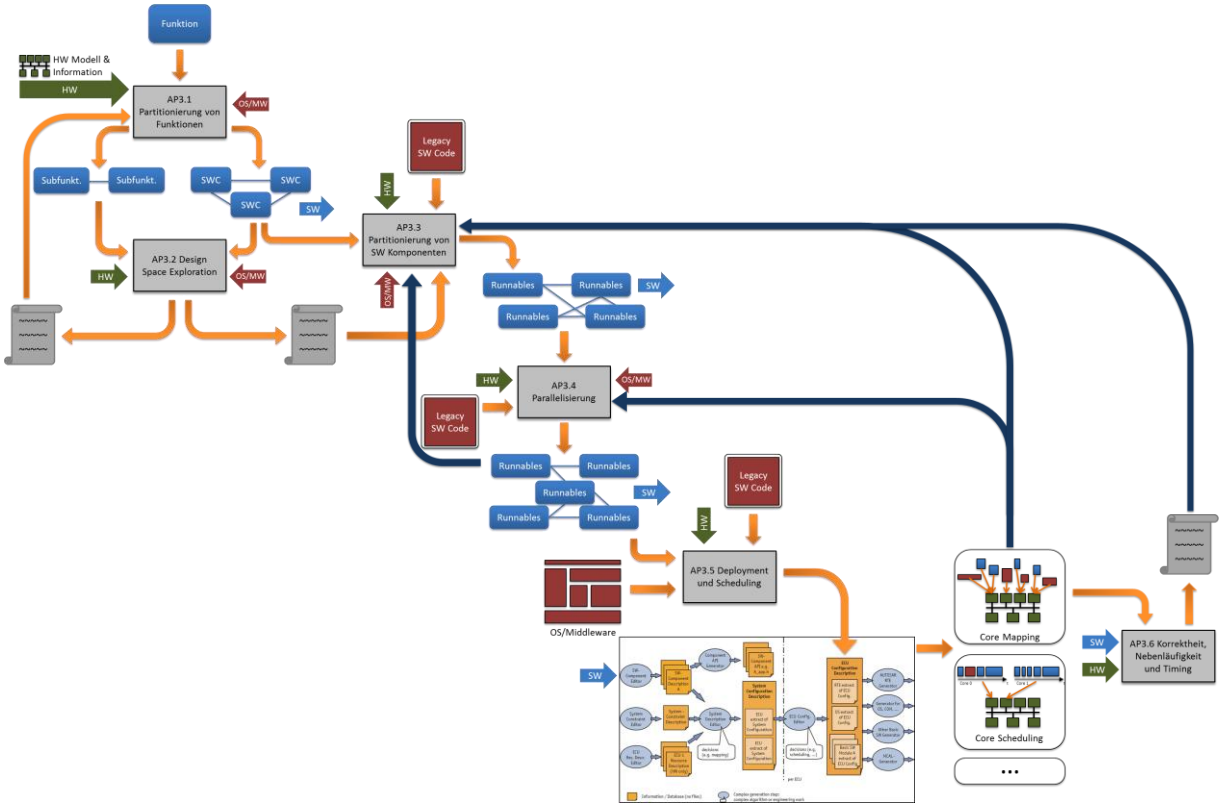
Higher degree of automation in the development due to tool support

More than 30 partners and much more tools

aramis II
ENTWICKLUNGSPROZESSE | WERKZEUGE | PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME



The overall process



What's all that in aid of?

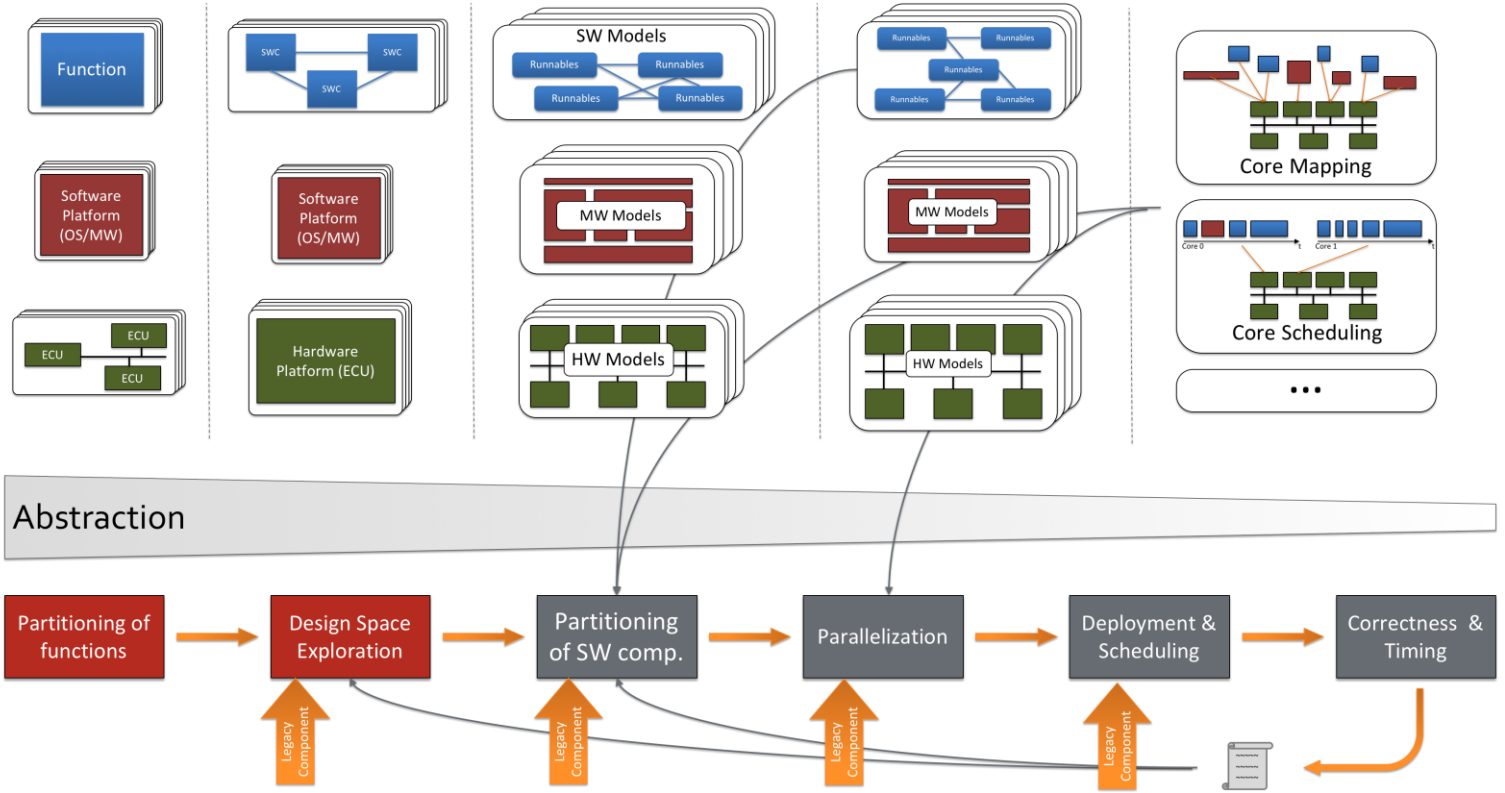
ENTWICKLUNGSPROZESSE | WERKZEUGE | PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME

Automotive

Avionics

Industrie-automatisierung

Scientific and technical approach

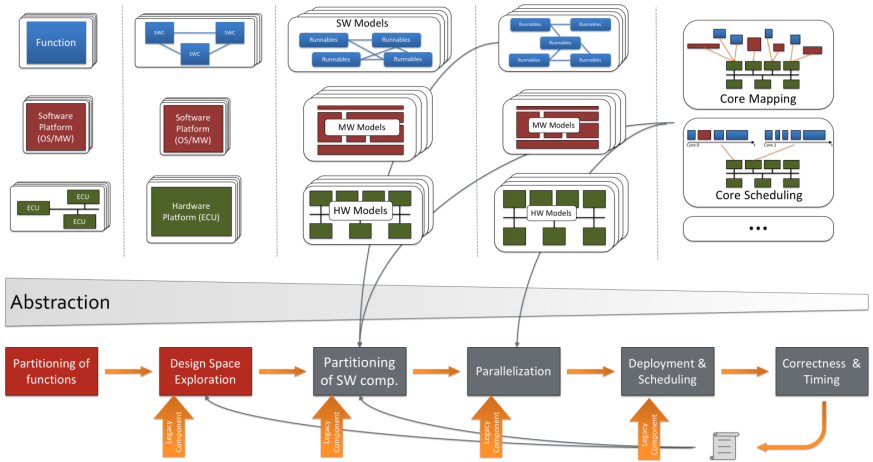


Agenda

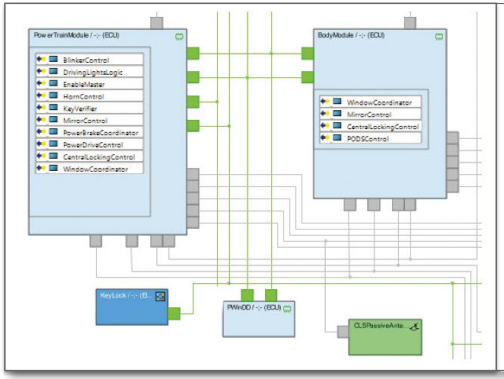
- **Motivation and Approach**
- Tooling
  - Partitioning of functions
  - Design Space Exploration
  - Partitioning of SW-components
  - Parallelisation
  - Deployment and scheduling
  - Correctness and timing
- Tool chains
- Summary and outlook

Agenda

- Motivation and Approach
- **Tooling**
  - **Partitioning of functions**
  - **Design Space Exploration**
  - **Partitioning of SW-components**
  - **Parallelisation**
  - **Deployment and scheduling**
  - **Correctness and timing**
- Tool chains
- Summary and outlook

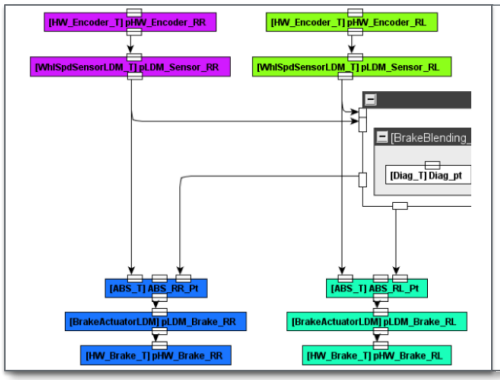


Tooling
Partitioning of functions



PREEvision (Vector)

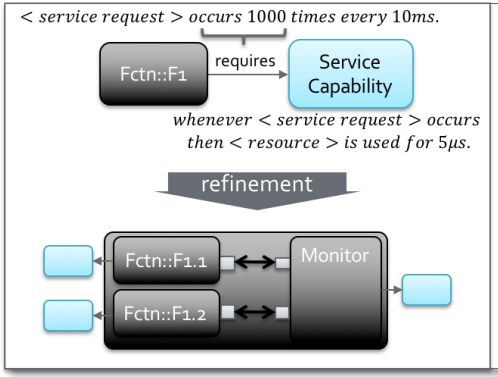
- Design, manage, test and document complete E/E systems
- Modelling multicore parameters in an early development step



AutoAnalyze (UNA)

- high-level function partitioning on e.g. EAST-ADL Feature Models
- transformation between abstract levels, abstraction of needed data

Tooling
Partitioning of functions

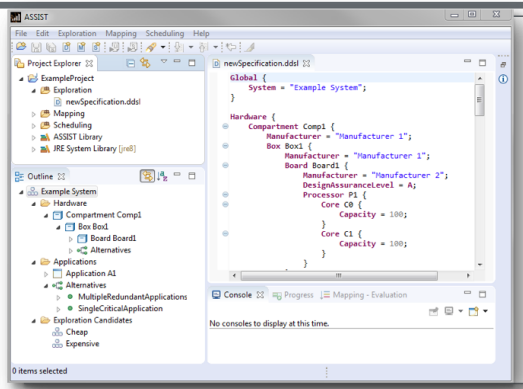


RTana$_2$ (OFFIS)

Supporting function partitioning:

- Formalization of required service capabilities in terms of contracts.
- Analysis of (contract) refinement.

Tooling
Design space exploration



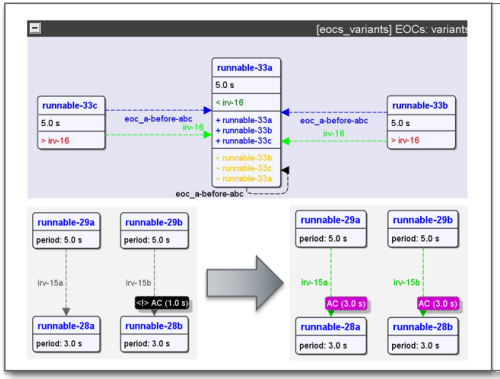
ASSIST

- Describe design space with variance points
- Feasibility check and optimization
- Available: OpenSource at GitHub

Semi-Automated Safety Analysis

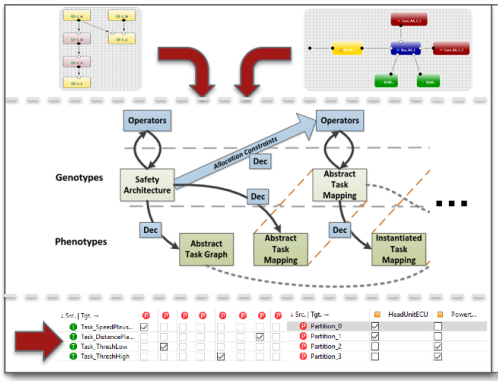
- Early safety assessment of the decomposition concept
- Knowing gaps in the interference safety concept
- Efficiency through model-based analysis method

Tooling
Design space exploration



AutoAnalyze (UNA)

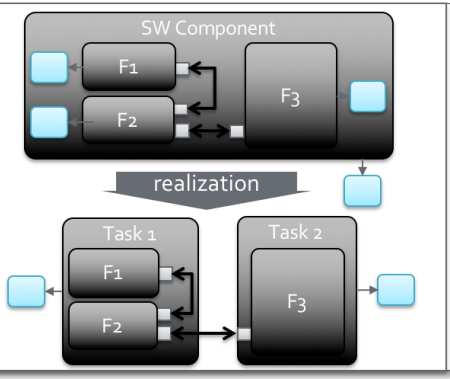
- model analysis (integrity and data consistency) and visualization of data dependencies
- semi-automatic validation



AF3

- Explores architectural design spaces: Evaluate impact on system configs
- Automate dependent design activities
- Safety: From design to system config

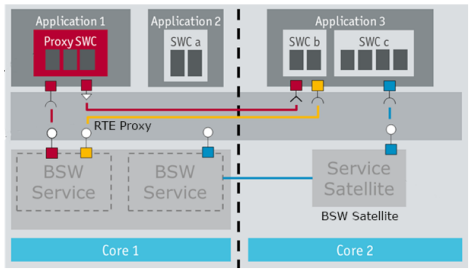
Tooling
Partitioning of SW-components



RTana$_2$ (OFFIS)

Supporting SW-comp. partitioning:

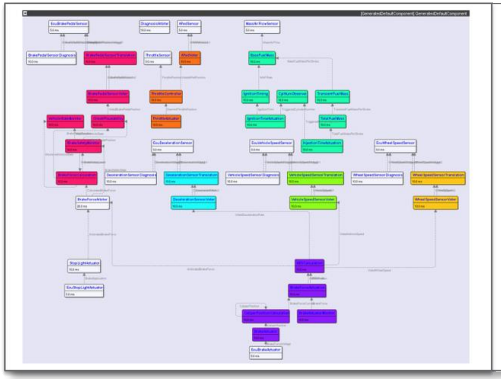
- Analysis of realization.
- From functions to SW-components to deployable tasks.



DaVinci Tools (Vector)

- Partitioning of AUTOSAR BSW
- Master-Satellite architecture
- Visualization and analysis of the configured AUTOSAR model

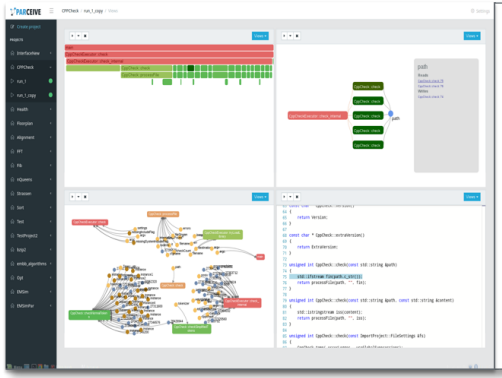
Tooling
Partitioning of SW-components



AutoAnalyze (UNA)

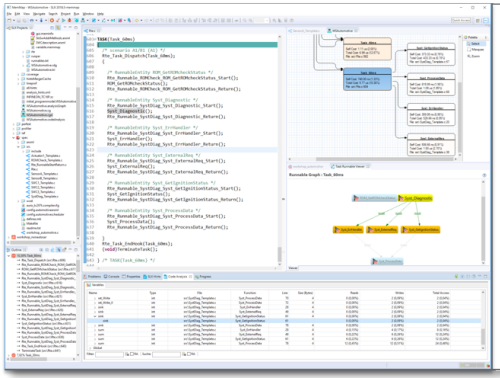
- partitioning of highly complex application software
- fast, efficient, widely applicable and scalable

Tooling Parallelisation



Parceive

- Dynamic analysis of legacy code
- Visualization of data accesses, function calls, timing information
- Different architectural views

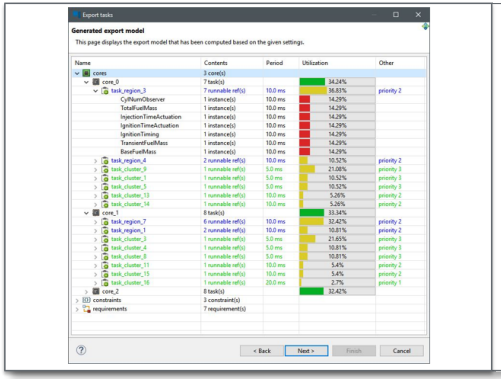


SLX code analysis

SILEXICA

- To get application insights
- Show application hotspots
- Investigate data dependencies
- Find parallelism in source code

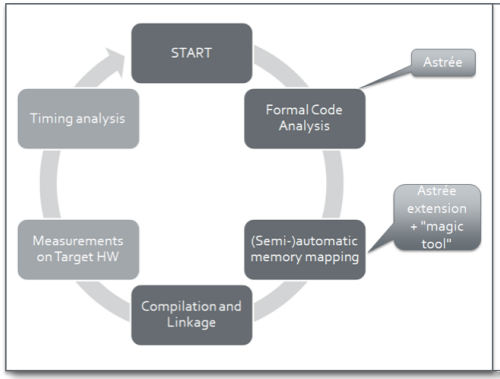
Tooling Parallelization



AutoAnalyze (UNA)

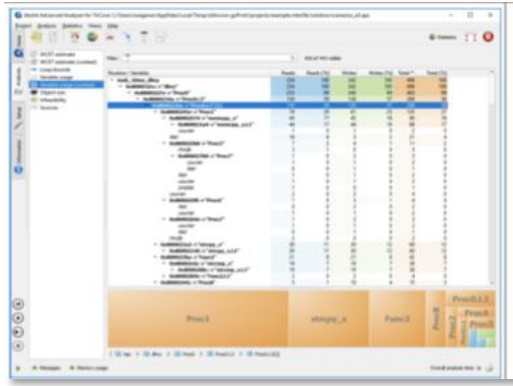
- mapping for heterogeneous multi-core target platforms
- optimized for safety, reliability or load-balancing concerns

Tooling
Deployment and scheduling



(Semi-)automatic memory mapping for safety-critical embedded systems

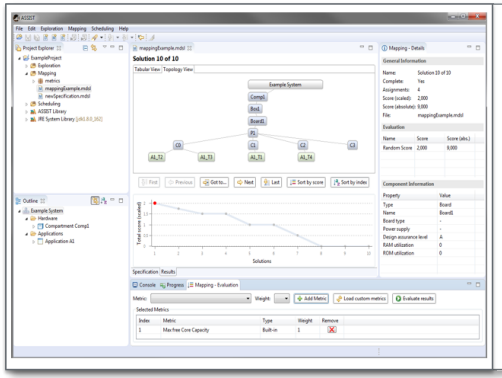
- Memory safety guaranteed through sound static C code analysis (Astrée)
- Mapping of variables and data structures to isolation realms (MMMT)



Understand and minimize interferences by static program analysis

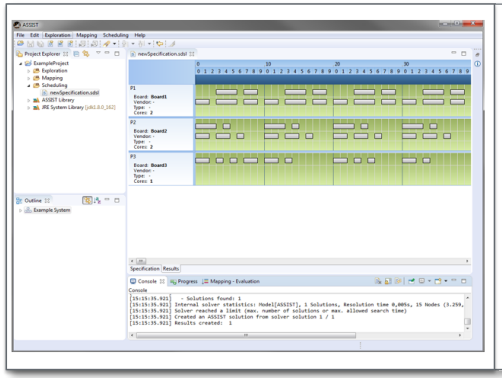
- Astrée data usage reports
- aiT/TimeWeaver variable usage statistics

Tooling
Deployment and scheduling



ASSIST (Mapping)

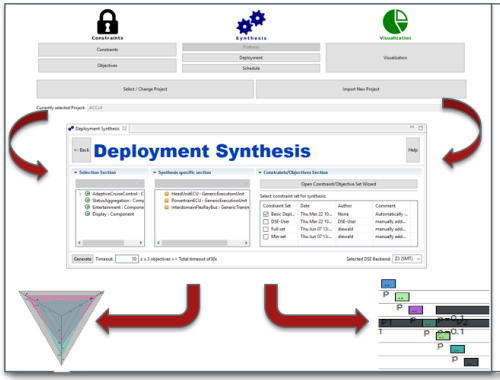
- Automated deployment generation and optimization
- Considers resources and safety
- Available: OpenSource at GitHub



ASSIST (Scheduling)

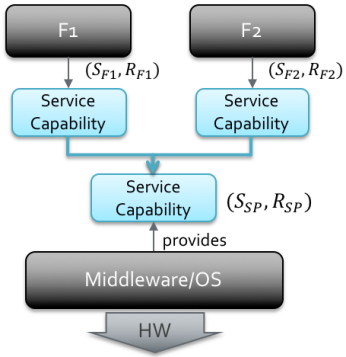
- Automated schedule generation
- Considers data dependencies and shared resources
- Available: OpenSource at GitHub

Tooling
Deployment and scheduling



AF3 – Fast Multiobjective Synthesis

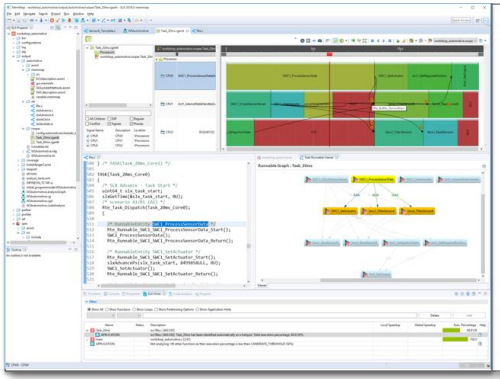
- Efficient deployments/schedules acc. to system metrics (e.g. energy) and platform properties
- Reusable constraint/objective spec.
- Predictable system behavior



RTana$_2$ (OFFIS)

- Allocation of service capabilities.
- SW functions → OS → Hardware
- Analysis of consistent realization for deployment and scheduling.

Tooling
Deployment and scheduling



Use SLX to migrate from sequential code to heterogeneous multicore

- Create parallel schedules
- Perform Memory Mapping
- Update AUTOSAR configuration

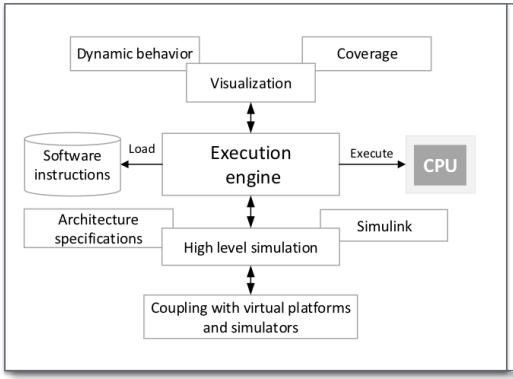
SILEXICA

TOOL SUITE

TA Tool Suite (Vector)

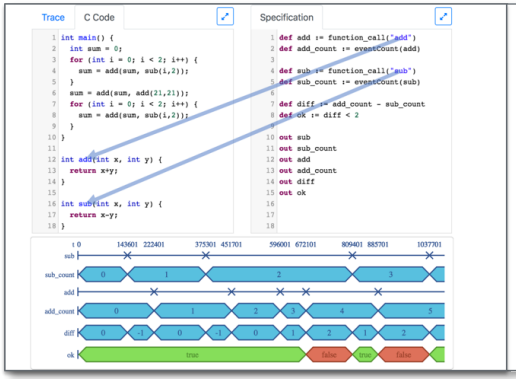
- Automatic deployment and time-triggered schedule generation
- Simulation (time-triggered and non-time-triggered environments)
- Considers Logical Execution Time

Tooling
Correctness, currency, timing



Testing on virtual platforms:

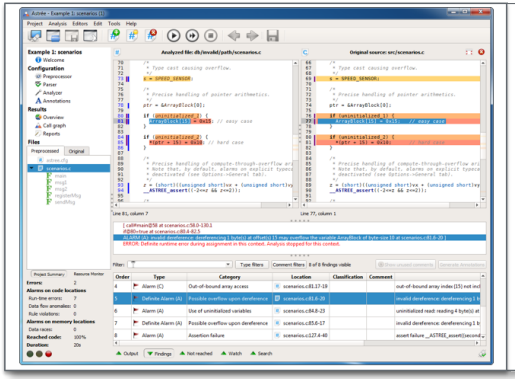
- Non intrusive trace collection.
- Sequential and concurrent coverage criteria.
- Dynamic behavior visualization.



Non-Intrusive Runtime Verification

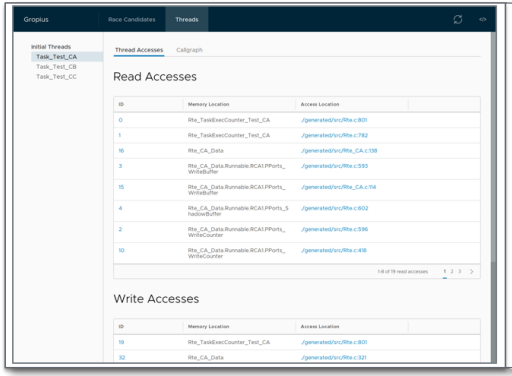
- Temporal Stream-based Specification Language (TeSSLa)
- Specify correctness properties and quantitative analysis of traces
- FPGA-based trace monitoring

Tooling
Correctness, currency, timing



Astrée: Sound C source code analysis

- Detect data races
- Taint analysis to provide evidence for security properties
- Exploit AUTOSAR OS configuration

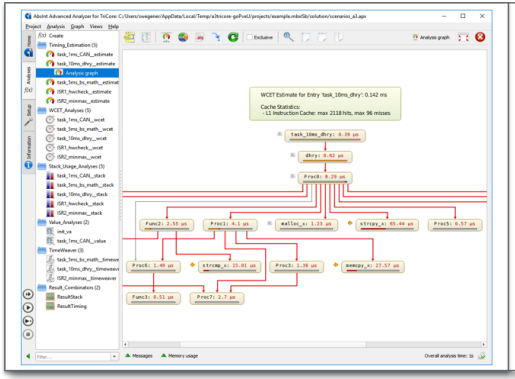


Gropius

- Static analysis of data dependencies between tasks
- Over-approximative search for data race candidates

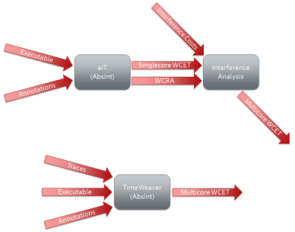
Tooling
Correctness, currency, timing

aramis II
ENTWICKLUNGSPROZESSE I WERKZEUGE I PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME



Multicore WCET Analysis

- aiT: static WCET + interference analysis
- TimeWeaver: hybrid measurement-based timing analysis

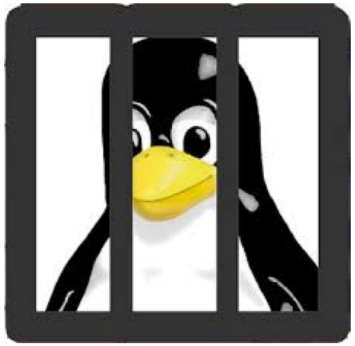



```
caramon@workmahine:~/Development/Lodin/release 76x22
[caramon@workmahine release]$ ./Lodin -p testmodels/threads/lls/petersons.ll

Lodin 0.2 (Jun  1 2018)
Revision: 0.2-329-g4f602da-dirty
LLVM: 6.0.0

Warning: Function 'crit' not defined by platform
Warning: No entry-point specified. Assuming main.
Random seed: 1528113493
System: NaiveGraph-explicit
Platform: PThread
Storage: SharedMem Storage
Successor: Standard
Prob-Successor: Standard

> Pr[<=500] (<> DataRace )
Verifying: Pr[<=500](<>DataRace )
47648 KB Runs: 8214 Satis: 3075 -  4122runs/sec [0.36388, 0.384931]]
```

Lodin

- verification of safety properties on LLVM code
- Under-approximates reachable states using statistical model checking

Tooling
Correctness, currency, timing



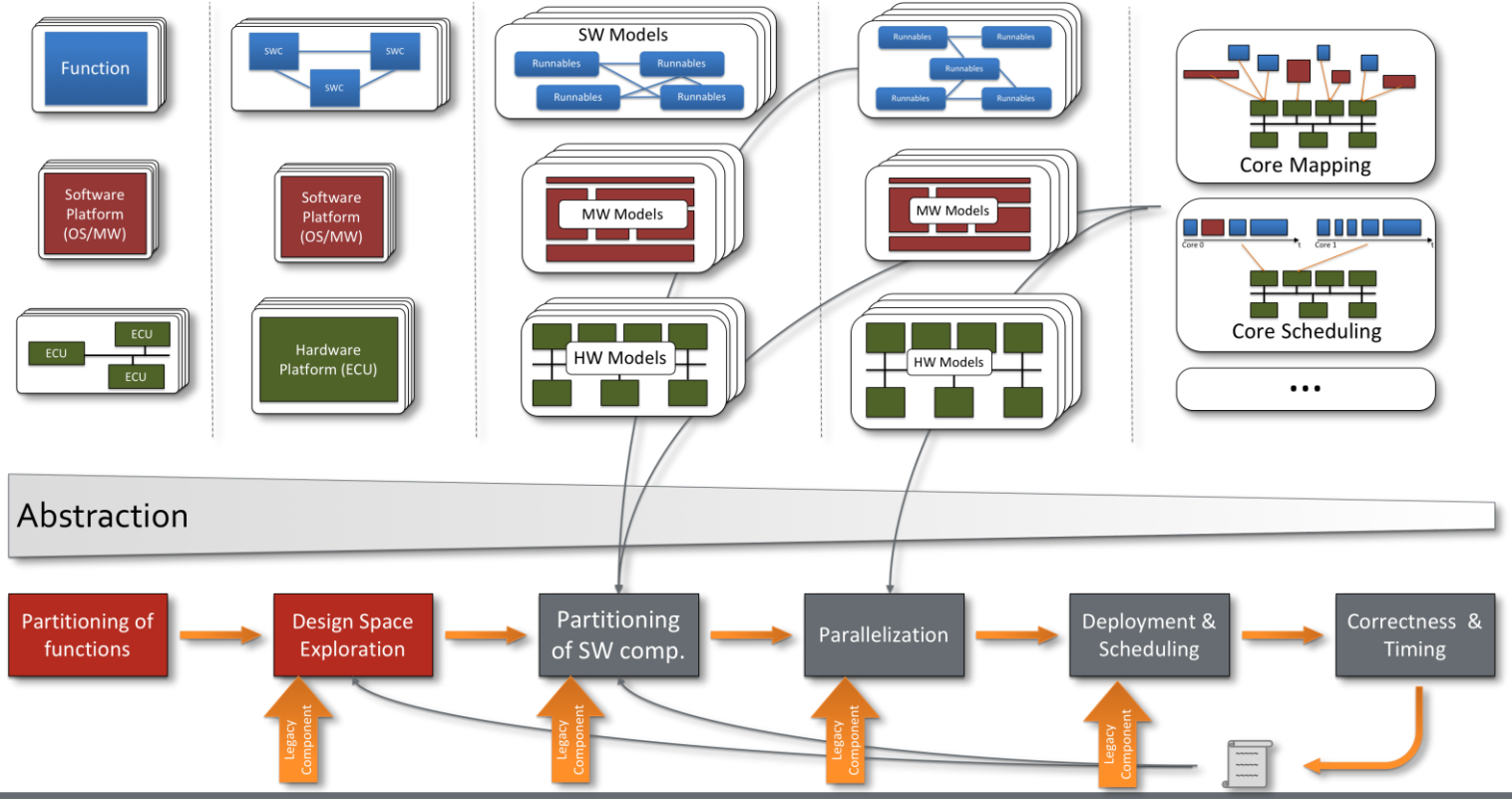
Jailhouse:

- Linux-based partitioning hypervisor for strong isolation
- Focus on hard real-time and safety-critical applications

Agenda

- Motivation and Approach
- Tooling
  - Partitioning of functions
  - Design Space Exploration
  - Partitioning of SW-components
  - Parallelisation
  - Deployment and scheduling
  - Correctness and timing
- **Tool chains**
- Summary and outlook

Tool chains


Function
SWC
SWC
SWC
SW Models
Runnables
Runnables
Runnables
Runnables
Runnables
Runnables
Runnables
Runnables
Core Mapping
Software Platform (OS/MW)
Software Platform (OS/MW)
MW Models
MW Models
Core 0
Core 1
Core Scheduling
ECU
ECU
ECU
Hardware Platform (ECU)
HW Models
HW Models
Abstraction
Partitioning of functions
Design Space Exploration
Partitioning of SW comp.
Parallelization
Deployment & Scheduling
Correctness & Timing
Legacy Component
Legacy Component
Legacy Component
Legacy Component

Agenda

- Motivation and Approach
- Tooling
  - Partitioning of functions
  - Design Space Exploration
  - Partitioning of SW-components
  - Parallelisation
  - Deployment and scheduling
  - Correctness and timing
- Tool chains
- **Summary and outlook**

Summary and Outlook

ARAMiS supports

- a lot of highly sophisticated tools with different funtionalities
- upcoming tool chains and interoperabilities and
- application in the different use cases
- integrated into the ARAMiS methodology

aramis II

ENTWICKLUNGSPROZESSE I WERKZEUGE I PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME

STRUKTURIERTER MULTICORE ENTWICKLUNGSPROZESS

MULTICORE METHODEN UND WERKZEUGE

INDUSTRIELLE PLATTFORMEN FÜR MULTICORE SYSTEME

Thank you for your attention!

Bernhard.Bauer@informatik.uni-augsburg.de

Universität Augsburg

Fakultät für Angewandte Informatik, Universitätsstraße 6a, 86135 Augsburg

aramis II

ENTWICKLUNGSPROZESSE I WERKZEUGE I PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME

Thanks to the whole ARAMiS consortium for their input

Bernhard.Bauer@informatik.uni-augsburg.de
Universität Augsburg
Fakultät für Angewandte Informatik, Universitätsstraße 6a, 86135 Augsburg