Index	Paper	EC1: Which functional domain(s) does the study analyze and/or modify in relation to SW architecture changes? E.g. ADAS, IVI (In-Vehicle Infotainment), Powertrain, Chassis...	EC2: Which system limitation(s) does the study identify as drivers for SW architecture changes? E.g. Busload, computing power, development costs, development time...	EC3: Which specific technologies does the study identify as enablers or catalysts for changes in the SW architecture? E.g. High-Performance Computing in autmotive, AI & machine learning, electrification, Over-The-Air updates and continuous deployment, connectivity - V2X and 5G...	EC4: How does the study technically address the integration of diverse software requirements (real-time, non-real-time, safety-critical, etc.) within a centralized automotive software architecture? E.g. virtualization via hypervisors, containerization...	EC5: Which architectural patterns or design practices are proposed to systematically support mixed-criticality in centralized automotive software architectures? E.g. Mixed OS environments, Service-Oriented Architectures (SOA), mixed-criticality scheduling, safety island / redundant compute...	Comment
1	Development of vehicle domain controller based on ethernet	ADAS	Cross-domain communication and vehicular communication that led to backbone ethernet	Electrification, intelligence, network connection and sharing	-	-	General: Low textual quality but interesting PoC. Missing mixed-criticality focus. Technical: - Decentralized SW to unified SWA - From CAN to Ethernet - Gateway controller based on Ethernet - SOA --> Service provider and consumer Risk: - Time delay until services are available
2	Contradiction of separation through virtualization and inter virtual machine communication in automotive scenarios	IC & IVI	High number of ECUs --> Increases weight, energy, space, costs for expensive hardware components, development and engineerings costs and time	Fully Digital Cluster Instruments (FPKs)	Virtualization via type-one/baremetall hypervisors, Virtual Machines (VMs), multi-core, isolated communication channels for shared resources; access control mechanisms for VMs to write/access shared resources; communication data signing; data consistency check;	Multi-OS environment; Design patterns for VMs: Safety relevant parts are separated on VMs to not get compromised; Different OSs to benefit of applications specific to particular OSs 'Limiting hardware resources for a VM' Dedicated hardware resources to isolate VM; Decoupling of development cycles - simplified updates; Architectural approaches - see figures: Clear separation approach - two or more OSs/VMs run without interconnections and dedicated I/Os; Layers of interconnections - various types of interconnections: unidirectional, bi-, App to OS, OS to OS, App to App etc.; Minimalistic Approach - Shared memory partition for each VM, only owner is allowed to write, only trusted OS is allowed to access --> no manipulation by other VMs, one way communication (fire and forget)	Consolidation of cluster instruments and IVI; Attention: inter-VM-communication weakens separation; Isolation interferes with the system's many interfaces; Third design as balanced mixture for real world automotive scenarios;
3	Autonomy-driven Emerging Directions in Software-defined Vehicles	SDV (Software-Defined Vehicle) - All of the domains	Embedded HW and SW architectures as bottlenecks - e.g. computing power; Growing demand of software - risk for real-time and FuSa; Zone-based: Delays between sensing and actuation; Service-oriented communication leads to uncertainties; --> 'Pessimistic timing estimates and inefficient implementations' --> Timing analysis challenges	Service-oriented communication - e.g. SOME/IP and DDS;	Containerization	SOAFEE SW architecture for SDV;	Focus of Paper: Digital Twin. Centralization introduces many challenges; Paper already starts from the zone-oriented E/E architecture as baseline; SOC/SDV shall improve update/upgrade capabilities; Scalability due to easier ECU integration; SOC --> Communication entities are loosely coupled; DDS instead of SOME/IP to ensure real-time; Shift Left to enable shorter SW development cycles --> Digital Twin; WCET analysis for processors architectures 'will likely never be fully resolved';
4	Time-sensitive autonomous architectures	Autonomous	Complexity, computing power, safety/latency, busload/bandwidth, flexibility, scalability, development effort, growing number of ECUs;	Autonomous driving, V2X, ethernet-based communication, Deep Learning	VTSN - Virtual TSN as virtual switch as one VM/VS; Jailhouse hypervisors to enable virtual machines; Jailhouse includes IVSHMEM as vPCI with permissions to ensure safety and security; Xen hypervisor with null scheduler to pin each virtual CPU to a physical CPU - one core for each virtual machine; Jailhouse outperformed Xen; TSN - Quality of Service; MPSoC Xilinx Zynq UltraScale+ (A- and R-Cores + FPGA accelerator); VLAN to ensure FFI; TSN scheduler Qav (frame prio, idleSlope etc.) and Qbv - Qbv outperforms Qav; Linux (performance, detection) and Erika Enterprise v3 (actuation) as open-source AUTOSAR RTOS; ROS2: DDS, Quality of Service (QoS), highly scalable; Isolation of detection and actuation; Actuation in single-core VM --> enhanced safety and predictability;	TSAA - Time-Sensitive Autonomous Architectures; Service-Oriented Architecture (SOA) --> Enables SDV; FPGA-based switches to enable TSN; Virtualization with advantages related to 'security, cost, reliability, availability, and adaptability'; Paravirtualization (higher performance) vs. full virtualization (all HW emulated); 'time-sharing mechanisms for shared hardware resources'; Freedom from Interferences; Multi OS environment; Temporal and spatial isolation; Domain/Feature/Function isolation; Static partitioning - cells allow 'guaranteed resource access and predictable performance';	Reduction of costs by usage of hypervisors and full hardware ausnutzung; Risk of multithreading - race conditions; DPU - Deep-Learning Processor Unit (DPU) on the FPGA; VS slightly increases latency but reduces standard deviation and thereby stability;
5	An Enhanced Algorithm for Memory Systematic Faults Detection in Multicore Architectures Suitable for Mixed-Critical Automotive Applications	ADAS/Autonomous	Computing power, functional safety	Multi-core, autonomous driving (as root cause for multi-core)	Spatial memory protection; Multi-layered-cache multi-core --> Dedicated caches; SW algorithms to ensure memory data integrity; Multiple copies, double inverse redundant storage, CRC etc.	Stack-monitoring mechanisms;	Only one author - Risk of bias; Focus on FuSa challenges | memory data integrity for multi-core; 'Safety code execution cannot be corrupted by a non-safety code'; Spatial memory protection; 'double inverse redundant storage with majority voting and with error detection codes'; --> Impact on CPU load etc.; CRC etc. also for NVM data; Elaborates 2 algorithms: 1 for FuSa compliant NVM read and 1 for FuSa compliant NVM write;

6	Software Architecture Modeling of AUTOSAR-Based Multi-Core Mixed-Critical Electric Powertrain Controller	Powertrain	Increase in efficiency; increasing complexity of software implementations;	Electrification;	Balance of watchdog checkpoints;	-	Focus on model-driven programming; Non-functional requirements - performance parameters (e.g. CPU load vs. safety constraint); Electric powertrain as mixed ASIL criticality software; AUTOSAR - Emphasizes shift from ECU-centric to functionality-centric approach; Chap. 3.2: Electrification brings more SW complexity; Watchdog to ensure proper program execution by checkpoints and program flow graphs - logical and temporal; Attention: Too many checkpoints can lead to time overhead and thus compromise FuSa; Task comprises many functions called runnables; Bidirectional traceability required between RM and M/S;
7	Problems and their mitigation in system and software architecting	-	SW complexity;	-	-	-	Use of product lines to manage complexity; Importance of architecture often neglected; Many referenced literature that empahsizes importance of architecture; Focus is on problems in software architecture development - No direct focus on technical solutions for mixed-criticality;
8	Design of Criticality-Aware Scheduling for Advanced Driver Assistance Systems	ADAS	Change of criticality level based on dynamic changes in environment	Autonomous driving	Different partitions for different criticality levels	Multi-core partitioned architecture, virtualization using hypervisors, from temporal and spatial isolation to mode changes --> mode based dynamic core allocation	- Partitioning to meet the stringent certification requirements - Sharing/integrating resources in multicore architectures --> reduce power consumption, volume, weight, space, cost and the number of context switches
9	A Security Process for the Automotive Service-Oriented Software Architecture	Autonomous; Powertrain	Cyber-Security	Autonomous driving, electric driving	-	ASOA; task scheduling acc. to criticality;	ASOA uses DDS for a decentralized data transmission; Service-oriented = data-centric; Paper presents security for the ASOA framework; Centrally managed SW architecture based on fully decoupled services; Making maintenance and update of cyber-security simpler and less costly; DDS fully decentralized --> no need for central broker; Beacons are sent out periodically to discover each other in DDS; Quality of service to ensure e.g. hard latency requirements;
10	Towards the deployment of a centralized ICT architecture in the automotive domain	Driving performance - Powertrain; Comfort - ?; passive and active safety - ?; drive-by-wire - ?;	Increasing number of heterogeneous, distributed ECUs; Increasing system complexity; Demand for computing power and data bandwidth; Heterogeneous networks; lack of fail-operational behavior; increasing demand for interconnection, complex system verification; limited flexibility;	Drive-by-wire; autonomous driving assistants; electrification;	-	Redundancy of centralized HPCs; Data-centric paradigm - Consideration of functional and non-functional requirements ensures that runtime system fulfills QoS demands; Centralized ICT architecture will pay off on the long run (lowered barrier for market entrance when standardized SW architecture etc., resource saving, faster development cycles, personalization);	Increasing amount of sensors with high demand on bandwidth and quality-of-service; Data-driven - Focus on data flows and processing instead of components; Electrification leads to disruptive change as a chance to rework ICT (info. and com. tech.) architectures;
11	A Modular Five-Layered V-Shaped Architecture for Autonomous Vehicles	Autonomous	Safety, Comfort	Autonomous	-	Safety-critical partitions to be designed redundantly with self-diagnostic capability to hot-switch;	Discusses functional system architecture and not a software architecture as stated in the abstract - To be excluded?; Autonomous system must be self-aware and redundantly operable; Autonomous system must be able to self-diagnose;
12	Autonomous driving systems hardware and software architecture exploration: optimizing latency and cost under safety constraints	Autonomous	Latency, safety, cost	Processor technology; computer vision; object recognition; deep learning; cross-domain dependencies;	Usage of multiple processors; ASIL related bundling of functionalities to dedicated uPs to not interfere; Mapping of similar SWCs onto same SoC partition;	Degeneration of tasks (ensure redundancy and fault-tolerance);	processing capacity of the processors influence how centralized the architecture can be' 'method to design autonomous driving systems for latency, cost, and safety'; Issue - Transfer of data between processors increases latency; High computing --> high costs; One single uP --> Requires high computing power, high costs, low latency; Paper indicates the relatency of a centralized SW architecture and a centralized SoC architecture with different uP partitions;
13	MPSoC-Based Platform for FailOperational Control of an Automated Research Vehicle	Autonomous, Powertrain	Safety	Autonomous, electrification	RPU (real-time processing unit): RTOS for most safety and time critical application; API (application processing unit): Embedded Linux with PREEMPT_RT kernel; Redundant memory partitions;	ASOA; Mixed OS environment; memory redundancy;	Target: fail-operational, real-time ECU; 'safety-critical real-time control of a fully automated vehicle via a service-oriented architecture;
14	Modelling centralised automotive E/E software architectures	Autonomous;	Software complexity; scalability, robustness, maintainability;	Electrification, connectivity, autonomous, over-the-air updates, cloud computing, AI, Neural Networks, wireless protocols / 4G&5G, V2X	Multi-core; memory management; resource partitioning; private memory; private cache; manycore processors;	Redundancy, Spatial and temporal isolation to enable FFI; AUTOSAR Adaptive; SOA;	Study investigates automotive architectural languages; Rubus Component Model supports modeling of centralized SW architectures best; Focus more on E/E architecture in general instead of SW architecture in detail; Study emphasizes lack of standard/well-accepted SW architecture for various vehicle manufacturers; Distributed (traditional) --> domain-centralized (in-development) --> Vehicle-centralized (future)

15	A Method for designing and analyzing automotive software architecture: A case study for an autonomous electric vehicle	ADAS, Powertrain, Battery Management System, Connectivity	Increasing complexity, increasing required CPU and memory resources, decreasing evolvability, safety and real-time, time-to-market, development costs, end product cost reduction. Hard real-time vs. soft real-time vs. no real-time (quality of service characteristics - SOME/IP or DDS) - Decreasing HW dependability.	Autonomous driving, electrification, connectivity	CBSPA: 7 layers / hierarchies of automotive software - Sensor/actuator types vs. process types - Hardware dependency on lower layers - Update frequency, computational complexity and agility on higher layers - Serveral combined OSs and platforms with hypervisor - Decouple software functions from hardware	SOA, microservices, software-defined vehicle, centralization/consolidation For FuSa focus: Redundancy, FuSa decomp --> Reduce development costs, timing and memory partitioning for FFI, real-time scheduling with timing partition for mixed-ASIL tasks Good safety-related SW architecture: Higher number of lower or equal rated SWCs than C, low number of connections between FuSa related SWCs, low number of low-ASIL SWCs assessing high ASIL-SWCs Protected dual channel pattern vs. decomposed channel with centralized voter with monitor pattern	- From traditional V-Model only to V-Model/Agile - Zonal controllers should not include processing functions - Page 2, first two sentences: Is it meant that developers should focus on SWCs and SW architecture, not in features?
16	Key Technology and Standardization Route for New Electronic and Electrical Architecture of Intelligent and Connected Vehicles	Connectivity, Autonomous; Powertrain; Chassis; IC & IVI;	Busload / data transmission; complexity of network communication; intelligence; computing performance;	Cloud/Edge computing; connectivity; autonomous driving; V2X;	Virtual Machines; Containers; multi-core;	SOA; BCEA (Brain Centralized Electronic and Electrical Architecture); AUTOSAR Adaptive; mixed OS environments;	In-vehicle Ethernet technologies need to be standardized;
17	RACE: A centralized platform computer based architecture for automotive applications	ADAS, autonomous, powertrain	Increasing lines of code / SW complexity; Cross-domain communication;	Autonomous driving; electrification; After-market extendibility;	Time and space partitioning of applications;	SOC - Publisher/Subscriber to decouple communication between applications;; IMA; ARINC --> Partitioning to segregate mixed-criticality applications; AUTOSAR also supports partitioning? SIMATIC?	Researchers predicted in 2013 shift from distributed to centralized; RACE platform with centralized platform computer same as avionics; Central HPC resulted in redundant power supply, com. infrastructure and controllers; Ring topology for ethernet so that no parallel redundant network is necessary; Cyclic safety relevant and acyclic non-safety relevant frames - IEEE 802.1 TSN;
18	E/E Architecture Synthesis: Challenges and Technologies	ADAS, Autonomous, IVI	Computational power; safety (low latency, safe persistency, safe data transmission); software/application complexity; growing number of ECUs, wiring harness, communication bandwidth, cost, software variants; security;	ADAS and autonomous driving features; AI-based applications; deep and machine learning; gaming in IVI; edge and cloud computing; OTA updates; V2V; perception, mapping, planning;	Multi-core; SoCs; middleware; hypervisor; TSN; Type 1 hypervisor - 'direct control and access to hardware resources'; One core per partition to isolate resp. avoid FFI; Type 1 better than Type 2 with host OS; static versus dynamic task mapping;	Software-defined vehicle; Virtualization to integrate safety and non-safety;	Good summary on evolution of E/E architectures; Highlights challenge of integration of an increasing complexity consisting out of non-critical and safety-related partitions; Mapping SWC to hardware elements; Software integration and configuration;
19	HyFAR: A hypervisor-based fault tolerance approach for heterogeneous automotive real-time systems	ADAS/Autonomous, Powertrain	Safety / Fault tolerance; updateability; busload, computing power;	Autonomous; battery cooling; lane departure warning; cloud connectivity, user connectivity;	Virtual Machines via hypervisors; virtual address spaces; Type 1 (bare metal) versus Type 2 hypervisors - Type 2: If host OS fails, hypervisor fails and all VMs; Full (HW emulation leads to overhead) versus Para Virtualization (highest level of FFI); Cache coloring, memory throttling or Arm's Memory Partitioning and Monitoring (MPAM); Type 1: Exclusive/static resource assignment / partitioning to virtual machines - Highest level of FFI; Type 1: Better performance and efficiency due to direct hardware resources access;	HyFAR; SOA; mixed OS environments; virtualization; freedom from interference;	Fault tolerance - Instead of redundancy recover concept via existing other ECU - reduces freedom from interference in case of same HW/SW redundancy failures; AUTOSAR Classic monolithic structure --> Low updateability; SOA - less strict requirements for real-time and safety; SIG architecturecfor uC while SOA architecture for uP with HW support for virtual address spaces via MMU; Hypervisor is only booted when needed; SIG in SOA-ECU feasible while SOA in SIG-ECU challenging;
20	Architectural patterns for cross-domain personalised automotive functions	ADAS, Comfort, Chassis, Powertrain	Computing power; Design complexity;	Integration of machine learning for e.g. computer vision, automated driving, customer personalization	-	-	Does not answer EC4 and EC5; Focus is on architectural patterns and their pros/cons, how ML can support to realize personalized functions; Context Aware Systems - Adaptive and intelligent functions that consider customer behavior; Conway's Law - Technical SW/HW architecture of product folllows the organisational structure;
21	Time-of-Flight 3D imaging for mixed-critical systems	ADAS, IC & IVI	Computing performance, RAM/Memory	Time-of-Flight 3D image processing	Multi-core; lock-step operation mode;	Redundant compute; mixed-criticality scheduling;	Major challenge for Time-of-Flight 3D image processing is limited amount of SRAM memory; SEooC - Development based on assumptions about the intended use later on; --> Safety Manual required; HW and SW architecture are not clearly separated resp. study discusses more the HW related side;