KIA4SM - Cooperative Integration Architecture for Future Smart Mobility Solutions

Sebastian Eckl
Chair of Operating Systems
Technische Universitaet Muenchen
Munich, Germany
Email: sebastian.eckl@tum.de

Daniel Krefft
Chair of Operating Systems
Technische Universitaet Muenchen
Munich, Germany
Email: daniel.krefft@tum.de

Uwe Baumgarten
Chair of Operating Systems
Technische Universitaet Muenchen
Munich, Germany
Email: baumgaru@tum.de

Abstract*—An important topic situated in the emerging research field of Cooperative Intelligent Transport Systems (C-ITS) addresses the mutual interconnection between vehicles, mobile devices and the traffic and transport infrastructure in an ever-changing environment. Originating from completely different application areas, most services and functions offered by these traditionally closed and heterogeneous systems have been developed independently from each other. Hence, considering the vision of realizing cooperative behavior by out-of-the-box seamless integration makes it inevitable to re-design the overall cross-system collaboration. One possible conceptual solution for integration may be found in the overall system's adaptation to Hardware/Software Plug-and-Play properties. In our case, device independent provision and dynamic reconfiguration of functionality at run-time (applying self-* capabilities) will be the concrete usage scenario for demonstrating this concept. Therefore, a flexible integration architecture based on universally applicable Electronic Control Units (ECUs) is needed to assure the feasibility of software components on the above mentioned affected heterogeneous devices. In this paper, a new approach on implementing and testing of such an architecture will be outlined. Plug-and-Play functionality shall be provided by a homogeneous, virtualized software-platform (L4 microkernel-based hypervisor) for heterogeneous hardware devices. Due to virtualization, former fixed hardware/software-bindings will vanish and give way to a flexible management of separated container, each providing secure access to software-based functionality. The hereby enabled context-sensitive reaction behaviour (e.g. en-/disabling, migrating or upgrading/updating of software-based functions) will have to be tested in a virtual field test environment to ensure the retention of the required automotive degree of safety and reliability.

I. INTRODUCTION

Within the foreseeable future, the current trend of steadily advancing intermodal mobility will push existing traffic and transport systems to the limits of capacity. Despite generally declining accident figures, serious accidents causing plenty of casualties and fatalities still remain a constant problem [1]. Accompaniments like traffic jams and traffic disturbances annually generate an enormous economic loss, averaging out at 250 million per day in Germany alone. Moreover, congestion caused fuel consumption of about 33 million liters per day leads to an additional and particularly senseless environmental impact [2]. Therefore, the introduction and establishment of intelligent traffic and transport systems is intended to improve road traffic as a whole by increasing overall safety, efficiency and environmental friendliness.

Hitherto approaches addressing this problem area have been traditionally separated into two independent domains: the car itself on the one and its respective environment on the other hand. Hereby, increasing in-car intelligence over the years has strongly concentrated on continuous technical improvement to exculpate the driver from driving-related tasks, starting from Computer-Assisted, over Highly-Automated to the final state of Autonomous Driving. In contrast, with the help of advanced applications, Intelligent Transport Systems (ITS) target on yielding global intelligent behavior into the overall system environment. Based on a centralized analysis, processing and smart combination of different heterogeneous and mostly isolated information sources from alongside the road infrastructure at run-time, e.g. originating from video surveillance systems, roadside weather sensors or provided by vehicles via floating car data (FCD), it supports the vision of a global intelligent and up-to-date navigation and routing. Irrespective of this, in the meantime a third column, originating from an entire different sector - the area of Information and Communication Technology (ICT) - is about to establish itself in the area of traffic and transportation. The increasing proliferation of smart mobile devices (e.g. smartphones or tablets) slowly introduces a novel, pervasive, low-cost and still solid platform, providing sufficient support for sensing, communication and entertainment.

Nevertheless, the current type of interplay between the above mentioned participants still lacks the possibility of working together ad-hoc and directly in a cooperative way - a prerequisite for effective real-time communication, coercively required in highly dynamic environments like the transportation landscape. Consolidated by the term "Smart Mobility" [3], the current trend of mutually interconnecting vehicles, road infrastructure and mobile devices tries to unify the former independently developed approaches under the enhanced definition of Cooperative Intelligent Transportation Systems (C-ITS). C-ITS, a sub-category of existing ITS services, tries to overcome these drawbacks by fostering local and decentralized vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) interchange via dedicated short-range communication technologies (e.g. IEEE 802.11p). Therefore, the mutual interconnection between different participating road users as well as responsible road authorities will extend the former solely concentrated processing of namely distributed but still isolated data silos. However, in the future this will lead to higher pretense regarding the ability of cooperative collaboration among the participating services and functions. Unfortunately, by the majority, C-ITS clients utilize traditionally closed systems, developed independently from each other and running on heterogeneous hardware devices. Due to these diverging evolutionary histories and application domains a system-inherent incompatibility exists, which naturally prohibits a simple out-of-the-box seamless integration. Therefore, for realizing the vision of an effective cooperative behavior it is inevitable to re-design the overall cross-system collaboration capabilities.

Hence, a flexible integration architecture based on universally applicable ECUs is needed to provide a homogeneous and common run-time environment which assures the feasibility of software components on previously mentioned affected heterogeneous devices (Figure 1). This approach will allow for the overall system's adaptation to Hardware/Software Plug-and-Play properties and thus lay the foundation for device independent provision as well as dynamic reconfiguration of functionality (e.g. migrating SW components between different ECUs) at run-time.

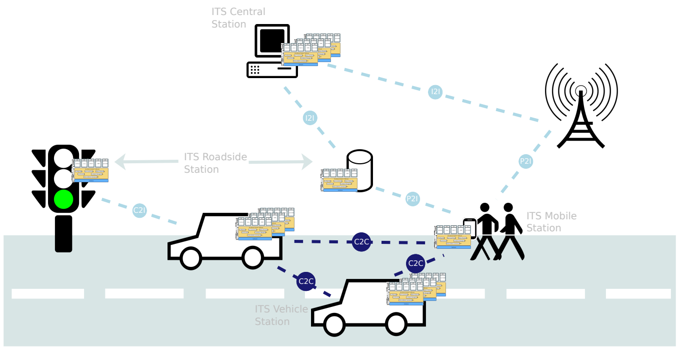

Fig. 1: KIA4SM vision - homogeneous platform for heterogeneous devices

II. FOUNDATION

The proposed restructuring will affect the participating device categories (vehicles, mobile devices, traffic and transport architecture) to a varying degree. In particular, the projection poses a challenge to the existing type of in-car architecture. Compared to the fast change cycles prevailing in the sector of ICT or ITS-related fields, the automotive domain is still subject to a rather long-lasting and inflexible development life-cycle process, prohibiting quick reaction/response times to upcoming trends. The current automotive engineering practices reflect this problem by more or less sticking to the principle of designing a dedicated ECU per new (software-based) functionality. This results in a strong hardware-/software-binding, which complicates subsequent update or change of existing functionality due to the lack of a consistent platform. But with already up to 100 different microprocessors (tightly interconnected with each other) [4] situated inside a modern car, this practice will soon encounter its limits. A car's restricted package space in combination with the perpetual need to further reduce weight and energy consumption leads to high cost pressure in modern-day car manufacturing. In particular, this affects the upcoming trend of electric cars, in which the energy consumption of every single component has direct impact on the car as a whole - and especially on a car's range.

Therefore, hardware consolidation, originating from rather traditional areas like professional server hosting or computing centre operation, can be considered as a promising approach to solving the structural problems mentioned above [5]. Tightly coupled with virtualization, this trend will shift the focus from the existing multitude of small and specialized hardware devices to a manageable number of general-purpose ones. Additionally, this trend will rise the overall hardware compatibility between the in-car platform itself and its surrounding environment (ICT, ITS), which is already used to utilizing less-specific hardware. As a result, a reduced but dependable set of powerful ECUs has to deal with an ever-increasing number of software-based functionality. But in contrast to ICT and ITS applications, in the majority automotive functions are in a way safety critical and have to be clustered by different safety integrity levels (ASIL) [6]. Hardware consolidation hereby implicates new challenges regarding the partitioning of these former strictly separated software components, which are now intended to reside on the same ECUs. Regarding the functional safety aspect, the resulting mixed-criticality system still has to guarantee a strong isolation between the different SW components.

A proven concept for solving these safety-related issues can be found in microkernel-based paravirtualization, a technology that is able to comparably combine software of different criticality (e.g. non real-time and real-time aspects). An interesting approach, known from the area of embedded systems, may be found within the L4 family. The kernel itself acts as a Type-1 Hypervisor, which allows for partitioning of different software components via separated container, making them work on user mode level. The example of PikeOS [7] shows a commercially successful implementation, originating from the field of avionics, an engineering discipline where even stricter safety expectations than in the automotive domain exist.

As the virtualization technology already provides an adequate and solid foundation for addressing safety-related aspects, the main part of our own research will build upon the existing work and will lie its focus on the second field of interest - flexibility. As already depicted in chapter 1, in the context of KIA4SM the term flexiblity addresses the subject of enabling Hardware/Software Plug-and-Play functionality based on universally applicable ECUs. This goal can be decomposed into two objectives:

1) A common platform as foundation for device independent (vehicles, mobile devices, traffic and transport architecture) provision and execution of software-based functionality.
2) Mechanisms that allow for online dynamic reconfiguration, based on
    a) en-/disabling and relocation/migration of software-based functionality
    b) adaptive (data-centric) routing policy
    c) flexible scheduling of tasks per ECU

The foundation for device independent provision and execution of functionality will be provided by the above mentioned L4 microkernel-based paravirtualization, which facilitates the required common run-time environment. For providing the additionally needed reliable context-sensitive reaction behavior, which triggers an actual reconfiguration, in various fields of software-intensive systems, the concept of self-* is already considered as a promising approach. Among others, suitable applications can be found in the areas of computing centres and robotics. Unfortunately, existing automotive system architectures (e.g. AUTOSAR) are usually configured statically at compile-time and therefore lack in opportunities to dynamically change their behavior at run-time. Therefore, a novel in-car platform, which is based on two pillars - a common run-time environment and the principle of self-* - will be required for keeping up the current standards regarding safety and reliability by utilizing a certain degree of flexibility.

By finally mapping and expanding this approach to the remaining C-ITS clients (mobile devices, traffic and transport architecture), also the previously mentioned pretense of adapting the overall system to the aspired Plug-and-Play behavior will become feasible. The common run-time environment acts as a basis for the implementation of universally applicable ECUs. Based on this, self-*-enhanced reconfiguration will then handle the device independent provision and (inter)change of functionality at run-time.

So, in this paper, a new approach on implementing and testing of such a platform will be outlined. The proposed joint basis for this ECU platform will rely on the previously mentioned L4-based microkernel, offering safety and strong modularity by sticking to the principles of separation by virtualization. A dynamic container-management, related to hardware/software Plug and Play (PnP) functionality, can provide (automotive) context-sensitive reaction behavior. This especially affects well known concepts, like en-/disabling, relocating/migrating, upgrading and updating of software-based functions, which originate from existing desktop operating systems. By adapting these concepts to the context of distributed embedded systems, the dynamic interaction and cooperation of affected components will have to be tested in a virtual field test environment to ensure the retention of the required automotive degree of safety and reliability.

The remainder of the paper is organized as follows. Section III presents related work concerning the topics platform, self-* behaviour and existing virtual field testing approaches with regard to the automotive environment. In section IV the applied methods used to design the novel platform will be discussed. In section V, the logical view on the proposed novel in-car ECU platform will be outlined by showing the overall architecture. In section VI, the proposed self-* approach will be outlined in detail, providing information about the overall reconfiguration process and required platform-related adaptations. Section VII provides further details regarding the overall hybrid simulator test setup. The concluding section VIII finally will give a short summary.

III. Related Work

A comparable holistic approach of providing a flexible integration architecture based on the principles of Plug-and-Play by utilizing universally applicable ECUs has not yet been set up in this form. Nevertheless, several research projects originating from related subject areas have already addressed relevant aspects and therefore can provide useful preliminary work.

A. Platform

Based on well-established concepts applied in domains like avionics and automation, within the RACE (Robust and Reliable System Architecture for Future eCars) project [8] a centralised automotive ECU-platform has been developed. The platform offers a generic safety-critical runtime-environment which allows for the execution of mixed-critical functionality. Comparable to KIA4SM, the architecture has to support the combination of safety (fail-operational behaviour) and flexibility (flexible assignment of software-based functions to ECUs). Sensors and actuators are encapsulated within so-called smart aggregates and therefore decoupled from the actual computing nodes. Communication is realised via middleware-based RTE, applying a data-centric approach and guaranteeing deterministic behaviour. However, RACE solely concentrates on the vehicular ICT architecture, excluding the C-ITS environment. By utilizing middleware-based Plug-and-Play mechanisms [9], new components and functions may be added at a later point in time. Unfortunately, no information concerning the point in time of an actual reconfiguration (parking vs. driving state of the vehicle) is provided. As the underlying operating system (based on PikeOS separation kernel) utilizes a static/fixed scheduling approach, it is therefore arguable to which extent the overall system can be reconfigured dynamically. Because of the cyclic execution fashion of applications it seems conceivable that relocation/migration of software components is restricted to within a single RTE partition [8].

Within the project Adaptive City Mobility (ACM), the sub-project Software Defined Car develops an open and flexible ICT-architecture allowing for a future multimodal usage of small electrical vehicles. Based as well on the concepts of hardware consolidation and virtualisation, in contrast to KIA4SM a central electronic control unit (CECU) is intended to facilitate the parallel execution of mixed-critical functionality. Therefore, the platform itself is sub-divided into three strictly isolated partitions, reflecting the different types of stakeholders: a protected manufacturer partition executing real-time driving-related functions, an operator partition which is open for app installations as well as a user partition concerning interactions with a mobile device. In contrast to KIA4SM, execution of software-based functionality is not provided by the hypervisor itself but is taken over by a separate operating system (FreeRTOS for real-time applications, Android for Infotainment) [10] which is acting as intermediary inside an isolated partition. Reconfiguration therefore only affects the application level. To all appearances, relocation/migration of software-based functionality at run-time is excluded due to the fact that a closed environment with all possible interferences known at design time is expected [10]. Comparable to RACE, the project solely concentrates on the vehicular ICT architecture, excluding the C-ITS environment.

The project SafeAdapt (Safe Adaptive Software for Fully Electric Vehicles) [11] deals with the development of a novel and reconfigurable electric/electronic (E/E) architecture in order to manage the complexity when addressing the need for safety, reliability and cost efficiency in future Fully Electrical Vehicles (FEVs). In contrast to KIA4SM, the approach therefore will extend the existing AUTOSAR concept, by introducing a so-called SafeAdapt Platform Core which encapsulates the basic adaptation mechanisms needed for relocating and updating software-based functionalities. By applying hardware consolidation, the number of ECUs shall be reduced and multiple functions have to be combined onto generic platforms. Safety and availability shall be increased by relocating applications between ECUs in case of failure. Furthermore, SafeAdapt seeks to abolish mechanical fall-back solutions and therefore fosters the achievement of fault tolerance (fail-operational as well as fail-safe states) by providing software-based redundancy via automatic context-switching to a redundantly operated function on a different ECU [11]. Nevertheless, in contrast to KIA4SM, reconfiguration based on the migration of software-based functionality during run-time won't be addressed. By addressing the overall design methodology (methods, tools, building blocks) including the certification of individual components (regarding the automotive ISO 26262 standard), SafeAdapt additionally focuses on a rather holistic development approach which exceeds the scope of KIA4SM. But comparable to RACE, the project solely concentrates on the vehicular ICT architecture, excluding the C-ITS environment.

B. *Behaviour*

The project DynaSoft (Dynamic, self-organizing software systems) [12] wants to develop new design processes for future dynamic and self-organizing automotive software systems, in order to increase redundancy and reliability inside a vehicle by likewise simplifying the replacement of and improving the ability to retrofit components through dynamic allocation of software to ECUs according to the driving situation. In contrast to KIA4SM, DynaSoft therefore sticks to a model-based design approach, utilizing the EAST-ADL2 architecture description language and virtual prototyping based on SystemC description and modeling language. Like DynaSoft, the DySCAS (Dynamically Self-Configuring Automotive Systems) project as well focused on dynamic reconfiguration of automotive software systems [13]. Besides the intra-vehicular architecture, the project additionally addressed the management of sporadic system resources, e.g. a user's mobile device. In contrast to KIA4SM, the main focus on reconfiguration/self-configuration of software tasks lies on the middleware level [14].

C. *Test*

X-in-the-Loop (XiL) simulation describes a technique that is used throughout several engineering disciplines (e.g. automotive, avionic) during development and test of complex real-time embedded systems. Affecting the fields of automotive engineering, in particular Hardware-in-the-Loop (HiL) and Vehicle-in-the-Loop (ViL) are promising realizations. Regarding HiL, new functionality, implemented on a real hardware device (e.g. ECU) will be tested in detail, against a simulated, mostly software-based version of an overall (plant) system (e.g. car), in which it will operate from later on. ViL instead combines the advantages of a real (test) vehicle with safety and reproducibility aspects of a driving simulation.

VIRES Virtual Test Drive (VTD) [15], [16] provides a complete toolchain for real-time driving simulation applications by utilizing a four-step-based iterative XiL-approach, considering the aspects software/model, driver, vehicle and hardware. Key aspect is the assessment of advanced driver assistance systems (ADAS) and active safety systems. Besides traffic and scenario simulation, VIRES VTD provides a solid base for simulating different types of required sensors inside a virtual 3D environment (e.g. ray-tracing based laserscanner). A similar physics-based simulation platform, concentrating on ADAS and active safety, can be found in TASS PreScan [17]. In addition to VIRES VTD it also addresses the design and evaluation of V2V and V2I communication as well as autonomous driving applications. Unfortunately, for both products there is no information stating to what extent a complete vehicle finally can be included into the simulation itself. The OpenDS driving simulator [18] lays its focus on realistic vehicle, engine and environment physics and allows for an integration of real car steering controls (via CAN-bus). An exemplary test-bed implementation, taking special care on safely diagnosing and updating an ECU remotely and at run-time, is offered by Boczar et.al. [19]. Vehicle control/steering signals are sent from a computer-based open-source racing-car simulator (TORCS) to the underlying ECU network. After interpretation by corresponding ECUs, relevant information for actuating elements is sent back to the simulator and the resulting behaviour can be visualized in real-time.

IV. Applied Methods

In order to achieve the required context-sensitive reaction behaviour, different methods have to be applied for introducing and supporting self-* properties on software as well as on hardware level.

A. *Organic Computing*

The Organic Computing (OC) paradigm addresses the challenges of complex distributed systems by enriching them with life-like (organic) behaviour, resulting in the application of self-* capabilities (e.g. self-organization and self-configuration). Instead of having to predefine a system's behaviour to all possible occuring situations at design-time, the system will be endowed with a certain degree of freedom allowing for an intelligent way of reaction to unknown situations during run-time [20]. For adaptation purposes, the OC approach proposes a generic observer-controller architecture, designed in dependence on the MAPE (monitor, analyze, plan, execute) cycle, which originates from the concept of Autonomic Computing developed by IBM [21]. The observer has to collect data from the system and out of this computes indicators, which characterize the global state and the dynamics of the system. The controller has to compare these indicators with goals defined previously by the user and decides if reaction is required at all and what action has to be taken. Combined with the underyling system under observation and control (SuOC) an organic system can be formed (Figure 2, left side).

In the case of KIA4SM, the OC principle may easily be mapped onto the proposed L4 microkernel-based architecture, which will make up the common run-time environment (Figure 2, right side). Both observer and controller are intended to be implemented into independent and therefore isolated software partitions (L4 tasks). Together with the remaining software components running on the user level, microkernel and hardware base will form the SuOC. By separating the OC-related part from the rest of the system, the premise that an organic system has to stay responsive and stable can be guaranteed even if observer and controller stop working [20]. Moreover, KIA4SM applies a decentral OC approach, which means an observer/controller running on each universally applicable ECU. A comparable concept to KIA4SM, also following the microkernel principles, can be found in the CAROS (Connective Autonomic Realtime Operating System) approach [22]. But unlike in KIA4SM, the proposed Organic Manager itself is not part of the Operating System.

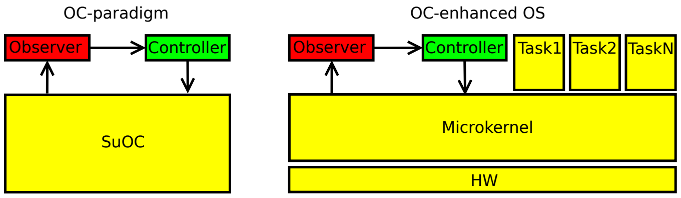

Fig. 2: Organic Computing: Applying the Observer/Controller pattern to existing microkernel architecture

B. *Mesh Networking*

Besides software-based reconfiguration, to a certain extent KIA4SM shall also support self-adaptation on hardware level. To allow for flexible re-routing in case of hardware failures (ECU drop out, cable outage) or purposive physical removal of components (e.g. switch off/disconnect a smartphone), the physical communication infrastructure will rely on a mesh-based networking topology - applied both inside and outside a vehicle. Outside the vehicle, wireless mesh networks have already sufficiently proven their capability in the context of mobile ad hoc networks (MANET) and vehicular ad hoc networks (VANET) [23]. Inside the vehicle, KIA4SM proposes a mapping of existing proven routing concepts originating from the wireless mesh background on their wired mesh counterpart, by relying on Ethernet. But concerning existing automotive (hard) real-time requirements regarding timing-related aspects, an industrial Real-Time (RT) Ethernet standard has to be applied [24]. Although relatively new, Ethernet-based mesh networking is already addressed in enterprise networks and data centres [25] via IEEE 802.1aq Shortest Path Bridging (SPB) [26] and IETF Transparent Interconnection of Lots of Links (TRILL) [27].

C. *Data Centric Communication*

In order to support reconfiguration on the logical communication level as well, KIA4SM will follow a data-centric approach. Data suppliers/consumers (sensors and actuators) are thereby decoupled from the actual processing components (applications) residing on respective ECUs. Therefore, data is not directly delivered from sender to receiver. Instead, following the Publish/Subscribe paradigm, a sender provides data to which a receiver can easily register [28]. By applying this concept to KIA4SM, components will be enabled to uphold data exchange and communication after relocation/migration of software-based functionality between different ECUs or a change of the underlying network topology [29]. Existing real world working examples successfully relying on this type of communication can for example be found in the area of wireless sensor networks [30], [31].

V. ARCHITECTURE

The proposed KIA4SM architecture concept is depicted in Figure 3. It is driven by the idea of likewise combining virtuality and reality by allowing the execution of the proposed OC-enhanced operating system on physical as well as virtual components. The architecture itself therefore can be decomposed into three layers.

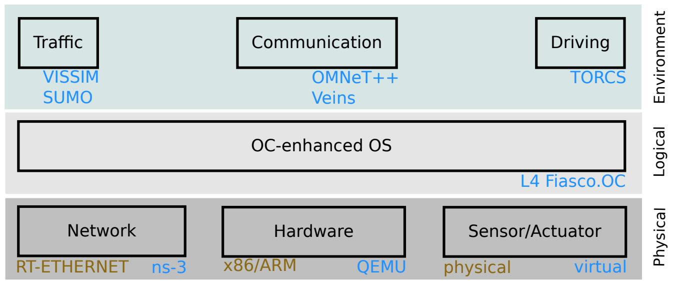

Fig. 3: KIA4SM overall architecture

A. *Physical layer*

The lowest layer will be made up of the physical components, required for the execution of the homogeneous software platform. It can be decomposed in the categories networking, processing hardware and sensing/actuating components. As described in Section IV-B, physical communication will rely on an Ethernet-based mesh network. On the virtual side, this task will be taken over by ns-3, a discrete-event network simulator [32]. As an underlying processing hardware, both x86- and ARM-based processor architecture will be utilized. Their virtual counterparts will be provided via emulation technology, by applying QEMU, a generic and open source machine emulator [33]. Relevant data can be provided by physical sensors and actuators (e.g. radar or lidar sensors) as well as on a virtual base via sensor simulation (see Section III-C).

B. *Logical layer*

The medium layer will consist of the OC-enhanced operating system presented in Section IV-A, which will act as the common run-time environment. As already explained before, it will be executable both on physical and virtual hardware. Further details on the platform composition and relevant OC-related enhancements will be provided in Section VI.

C. *Environmental layer*

The environmental layer will provide a simulated driving environment, thereby trying to reflect most relevant aspects of the physical world. In KIA4SM, driving-related environment will consist of three different frameworks, which will be combined via a simulator coupling approach [34]. First of all, road traffic simulation tools (e.g. SUMO [35]) will provide a realistic model of intermodal traffic participants, including road vehicles, public transport and pedestrians. For realizing decentralized V2V and V2I communication between the different road users, an existing VANET simulator will be utilized (Veins/OMNeT++) [36]. Finally, a simulated physical representation as well as a graphical 3D visualization of the

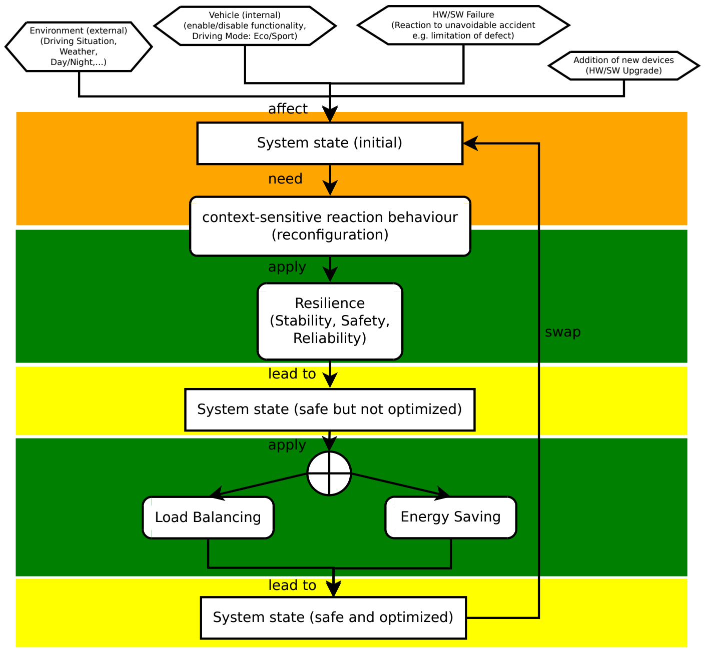
Environment (external) (Driving Situation, Weather, Day/Night,...)
Vehicle (internal) (enable/disable functionality, Driving Mode: Eco/Sport)
HW/SW Failure (Reaction to unavoidable accident e.g. limitation of defect)
Addition of new devices (HW/SW Upgrade)
affect
System state (initial)
need
context-sensitive reaction behaviour (reconfiguration)
apply
Resilience (Stability, Safety, Reliability)
swap
lead to
System state (safe but not optimized)
apply
Load Balancing
Energy Saving
lead to
System state (safe and optimized)

Fig. 4: Two-stage dynamic reconfiguration: combining safe adaptation and optimization

participants will be provided by an open-source racing car simulator [37]. This simulated physical environment will at the same time serve as a container for proposed sensor simulation.

VI. APPROACH

As proposed in Section IV, for introducing flexibility in a previously static system environment, several enhancements/adaptations have to be conducted. Therefore, at first a general abstract system reconfiguration process will be defined (Section VI-A). Afterwards, the necessary specific platform-related adaptations that have to be undertaken will be depicted. As a starting point, the vehicular ECU network architecture will be chosen. It comprises all relevant physical components (hardware, network, sensors/actuators) in a predefined (at least at the beginning) closed system environment. Later on, the concept will be expanded and therefore the remaining ICT and ITS devices will be included by breaking down existing vehicular system boundaries.

A. *Phase I: Two-stage OC workflow*

The proposed two-stage process can be seen in Figure 4. Two-stage thereby means that the overall reconfiguration process will be decomposed firstly into a safety-related and secondly into an efficiency-related component. The impulse for a potentially required adaptation of the system can be triggered by system external or internal events (Figure 4, upper layer) which will affect a particular (initial) system state. During the observer phase (orange phase), the need for context-sensitive-reaction behaviour will be investigated. If a need for adaptation is identified, the first controller phase (green phase) will check for a potentially required application of adequate reconfiguration mechanisms (e.g. relocation/migration of functionality) in order to guarantee resilience. This method can be seen as a type of software-based redundancy. This will lead to a new safe but not coercively optimized system state which will be processed to and executed by the SuOC (yellow phase). Therefore, afterwards two different optimization scenarios - either load balancing or energy saving - can be applied to improve the overall system's efficiency (green phase). This will lead as well to a new but this time safe and optimized system state (yellow phase) which from there on can be seen as the system's new valid (initial) system state (simultaneously closing and restarting the feedback loop).

B. *Phase II: Transformation to OC-enhanced microkernel*

The application of the proposed two-stage process from section VI-A onto the actual microkernel-based architecture demands modifications on the same. Therefore, the overall Observer/Controller pattern is subdivided into four subcomponents, representing the MAPE cycle (see section IV-A) already known from Autonomic Computing. The interplay between and the composition of the components can be seen in figure 5 (green block, medium and lower image). A central monitoring module (orange) gathers relevant data from the local underlying system. It represents the observer aspect of the organic system. The combination of local and relevant global data (delivered via the intra-vehicular network and monitored by other ECUs inside the distributed system) then is to be forwarded to the analysis (pink) and planning/learning (brown) modules, which together make up the controller part. Afterwards, the decisions made will be communicated back to the SuOC.

The most complicated aspect will be the derivation of an efficient online planning/learning methodology and therefore the main focus will concentrate on this subject. We propose a combination of both offline and online based machine learning techniques (Figure 5, orange and green boxes). The underlying problem to be solved can be described as the mapping of

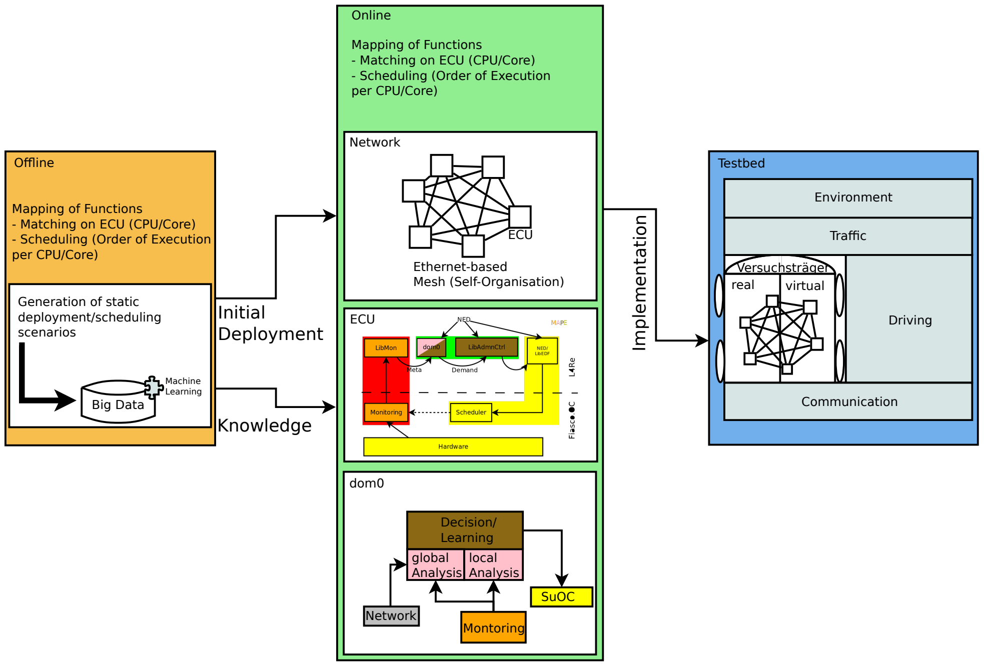

Fig. 5: Making a system organic: combining offline and online machine learning techniques

tasks to machines regarding the attributes space and time, meaning at first a matching of tasks to eligible processors/cores and afterwards calculating a valid schedule. This problem can be described as multi-objective optimization problem, often depicted in literature by multiprocessor scheduling or task allocation in distributed systems [38].

As a starting point, an offline-based design space exploration (DSE) toolchain (orange box) will be utilized to calculate possible and impossible task-to-processor/core combinations, each leading to schedulable or non-schedulable system configurations. A such potentially suitable toolchain may be provided by AutoFocus [39], [40]. At first, an adequate model of an ECU system and network has to be defined. Addtionally, some kind of self-description of the to-be-mapped software components (e.g. required system resouces, deadlines) as well as of the available hardware (e.g. processor, memory) is required in order to compute respective mappings. Resulting valid and invalid computed combinations will have to be saved in a database. Main research will now focus on the detection of suitable patterns, which can help in identifying correlations between valid and invalid combinations by analyzing their corresponding transitions. Therefore, offline machine learning (pattern recognition) algorithms will have to be applied.

As next step, a valid deployment configuration can be applied as an initial system state (Figure 5, upper box). Generated information (resulting out of the offline analysis phase) will additionally serve as knowledge base for the focused online learning capability, which has to be fulfilled by the OC-enhanced operating system (medium box). This will lay the foundation for deciding where to relocate/migrate software components in case of a required reconfiguration.

In order to expose the system to relevant events, potentially triggering a need for reconfiguration (as described in section VI-A), it will be connected to a simulator coupling set-up (see section V-C) that provides a useful environment for ongoing learning through generating sufficient data out of different driving-related scenarios (Figure 5, blue block).

VII. TESTBED

The operation of field tests is usually an expensive, time and resource consuming effort. However, an entire car can be tested under realistic conditions and particularly in cooperation with additional vehicles. But, thereby only a limited set of potential scenarios can be realized, mostly ignoring possibly hazardous situations. Existing XiL approaches and virtual test drive setups instead are executed in a rather static (virtual) environment, providing the ability for safely testing of even dangerous scenarios (see Section III-C). Unfortunately, they are usually limited to a specific subset of a vehicle's entire functionality, and therefore lack in opportunities for testing the big picture. Pure software-based simulation would offer the highest flexibility concerning potential testing scenarios, but is limited to the borders a simulation inevitably implicates. Therefore, the idea of likewise combining the advantages of both virtuality and reality, by placing an entire physical car together with its virtual counterparts into the same simulation, sounds very promising.

The underlying testing platform will rely on the concept of an ECU network, representing a (future) vehicle's electronic architecture. A physical implementation thereof will be placed on a 1/5 scale model RC car. Its virtual counterparts will consist of a simulated network of emulated hardware instances, running distributed on a high-performance server (see Figure 6). Each of the interlinked ECUs (both physical and virtual) will thereby represent an instance of the OC-enhanced operating system, presented in section IV-A. Hence, the same type of

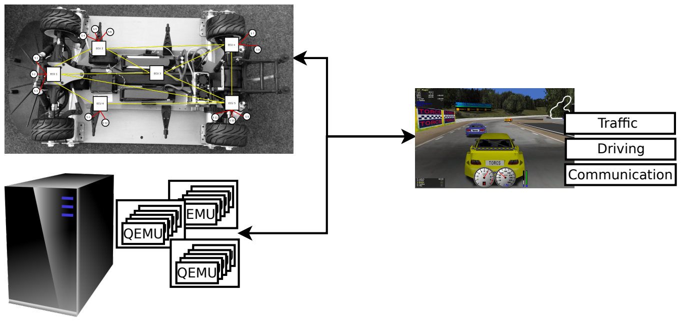

Fig. 6: KIA4SM testbed: hybrid simulator combination of physical and virtual components

software can be executed both on real and emulated hardware.

Both platform types will be connected to the proposed simulator coupling environment (see Section V-C), in order to imitate a realistic testing environment, which will be capable of executing different kinds of dynamic driving scenarios. Relevant information, concerning vehicle-typical actuating elements, will be sent to the simulator and the calculated driving behavior can be visualized in real-time (both on simulator and on model car). Therefore, data originating from real sensors (plugged to a physical ECU) as well as from virtual sensors (integrated in the racing-car simulator) shall be assimilated similarly. As the same in-car platform is running on both physical and virtual ECUs, also the testing of a hybrid vehicle, combining instances of both worlds, seems to be conceivable. Software will be able to run solely based on real ECUs, solely based on virtual ECUs or even on a mixed set of both components.

VIII. Conclusion

The anticipated result should exhibit that the currently existing system boundaries of each particpiant can be broken down by applying the novel integration architecture and therefore will be replaced by a cooperative, holistic and scalable system concept. The aspired cooperative collaboration will be based on the general ability of exchanging functionality between participating devices as needed. Particularly concerning the vehicle, this flexible fallback on existing residual capacities of the overall system could lay the foundations for context-sensitive reaction behaviour, allowing for a dynamic reconfiguration at run-time by at the same time keeping up the required level of automotive safety and reliability.

Acknowledgment

This work is partially funded by the German Federal Ministry of Education and Research under grant no. 01IS12057 through the project Software Campus (sub-project KIA4SM).

References

[1] ADAC, "Verkehrs- und Unfallstatistiken," https://www.adac.de/_mmm/pdf/statistik_2_1_unfaelle_insgesamt_42792.pdf, 2014.

[2] Bundesregierung, "Mobilität durch Fahrzeug- und Verkehrstechnologien," http://www.bundesregierung.de/Content/DE/Artikel/WissenschafftWohlstand/2008-01-01-hightech-verkehr-innovationsstrategie-januar-2008.html, 2010.

[3] smobility.net, "Smart MobiliTy in Thüringen (sMobiliTy)," http://www.smobility.net/, 2015.

[4] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, "Software Engineering for Automotive Systems: A Roadmap," in *2007 Future of Software Engineering*, ser. FOSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 55–71.

[5] G. Klasche, "Zum Stand von AUTOSAR," http://www.carmeq.de/media/auto%20805s.34PDFZum%20Stand%20von%20AUTOSAR.pdf, 2005.

[6] National Instruments, "What is the ISO 26262 Functional Safety Standard?" http://www.ni.com/white-paper/13647/en/#toc2, 2014.

[7] Sysgo, "PikeOS Hypervisor - Safety Architecture," https://www.sysgo.com/products/pikeos-rtos-and-virtualization-concept/safety-architecture/, 2015.

[8] S. Sommer, A. Camek, K. Becker, C. Buckl, A. Zirkler, L. Fiege, M. Armbruster, G. Spiegelberg, and A. Knoll, "RACE: A Centralized Platform Computer Based Architecture for Automotive Applications," in *Electric Vehicle Conference (IEVC), 2013 IEEE International*, Oct. 2013, pp. 1–6.

[9] C. Buckl, M. Geisinger, D. Gulati, F. J. Ruiz-Bertol, and A. Knoll, "CHROMOSOME: A Run-time Environment for Plug & Play-capable Embedded Real-time Systems," *SIGBED Rev.*, vol. 11, no. 3, pp. 36–39, Nov. 2014.

[10] O. Horst, P. Heinrich, and F. Langer, "ICT-Architecture for Multi-Modal Electric Vehicles," in *3rd Conference on Future Automotive Technology, CoFAT 2014*, Mar. 2014.

[11] P. Schleiss, M. Zeller, G. Weiss, and D. Eilers, "SafeAdapt-Safe Adaptive Software for Fully Electric Vehicles," in *3rd Conference on Future Automotive Technology, CoFAT 2014*, Mar. 2014.

[12] M. Zeller, "DynaSoft: Dynamisch selbstorganisierende Softwaresysteme für Automobile," in *Informatik 2009. Im Focus das Leben*, 2009, pp. 2253–2267.

[13] R. Anthony, A. Rettberg, D. Chen, I. Jahnich, G. de Boer, and C. Ekelin, "Towards a dynamically reconfigurable automotive control system architecture," in *Embedded System Design: Topics, Techniques and Trends*. Springer, 2007, pp. 71–84.

[14] R. Anthony, A. Leonhardi, C. Ekelin, D. Chen, M. Trngren, G. de Boer, I. Drke, S. Burton, O. Redell, A. Weber, and V. Vollmer, "A Future Dynamically Reconfigurable Automotive Software System," in *Tagungsband der 26. Tagung "Elektronik im Kraftfahrzeug. Systeme von Morgen - Technische Innovationen und Entwicklungstrends"*, 2006.

[15] VIRES, "VIRES," http://www.vires.com/products.html, 2015.

[16] VIRES Simulationstechnologie GmbH, "VIRES Virtual Test Drive Details," http://www.vires.com/docs/VIRES_VTD_Details_201403.pdf, 2014.

[17] PreScan, "PreScan," https://www.tassinternational.com/prescan, 2015.

[18] OpenDS, "OpenDS," http://opends.de/, 2015.

[19] U. Drolia, Z. Wang, Y. Pant, and R. Mangharam, "AutoPlug: An automotive test-bed for electronic controller unit testing and verification," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2011, pp. 1187–1192.

[20] J. Branke, M. Mnif, C. Muller-Schloer, and H. Prothmann, "Organic Computing - Addressing Complexity by Controlled Self-Organization," in *Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, 2006. ISoLA 2006*, Nov. 2006, pp. 185–191.

[21] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[22] F. Kluge, J. Mische, S. Uhrig, and T. Ungerer, "An Operating System Architecture for Organic Computing in Embedded Real-Time Systems," in *Autonomic and Trusted Computing*, ser. Lecture Notes in Computer Science, C. Rong, M. G. Jaatun, F. E. Sandnes, L. T. Yang, and J. Ma, Eds. Springer Berlin Heidelberg, 2008, no. 5060, pp. 343–357.

[23] F. Li and Y. Wang, "Routing in vehicular ad hoc networks: A survey," *IEEE Vehicular Technology Magazine*, vol. 2, no. 2, pp. 12–22, Jun. 2007.

[24] EPSG, "EPSG - Homepage," http://www.ethernet-powerlink.org/.

[25] Cisco, "10 Gigabit Ethernet DWDM Interconnections in Enterprise Campus Networks," http://cisco.com/c/en/us/products/

collateral/interfaces-modules/transceiver-modules/prod_white_paper0900aecd8054d53d.html.

[26] IETF, “RFC 6329: IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging,” http://www.rfc-base.org/txt/rfc-6329.txt, Apr. 2012.

[27] J. Touch and R. Perlman, “Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement,” http://tools.ietf.org/html/rfc5556.

[28] G. Pardo-Castellote, B. Farabaugh, and R. Warren, “An introduction to DDS and data-centric communications,” *RTI, Aug*, 2005.

[29] S. Sommer, M. Geisinger, C. Buckl, G. Bauer, and A. Knoll, “Reconfigurable Industrial Process Monitoring Using the CHROMOSOME Middleware,” *SIGBED Rev.*, vol. 10, no. 4, pp. 61–64, Dec. 2013.

[30] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, “Data-centric Storage in Sensornets,” *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 137–142, Jan. 2003.

[31] B. Krishnamachari, D. Estrin, and S. Wicker, “The impact of data aggregation in wireless sensor networks,” in *22nd International Conference on Distributed Computing Systems Workshops, 2002. Proceedings*, 2002, pp. 575–578.

[32] ns 3, “ns-3 homepage,” https://www.nsnam.org/.

[33] F. Bellard, “QEMU, a Fast and Portable Dynamic Translator,” in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC ’05. Berkeley, CA, USA: USENIX Association, 2005, pp. 41–41.

[34] DCAITI, “Simulation of Vehicle-2-X Communication,” https://www.dcaiti.tu-berlin.de/research/simulation/.

[35] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, “SUMO - Simulation of Urban MObility - an Overview,” Oct. 2011, pp. 55–60.

[36] C. Sommer, R. German, and F. Dressler, “Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis,” *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[37] TORCS, “torcs News,” http://torcs.sourceforge.net/.

[38] B. A. Shirazi, K. M. Kavi, and A. R. Hurson, *Scheduling and load balancing in parallel and distributed systems*. IEEE Computer Society Press, 1995.

[39] F. Hlzl and M. Feilkas, “13 autofocus 3 - a scientific tool prototype for model-based development of component-based, reactive, distributed systems,” in *Model-Based Engineering of Embedded Real-Time Systems*, ser. Lecture Notes in Computer Science, H. Giese, G. Karsai, E. Lee, B. Rumpe, and B. Schtz, Eds. Springer Berlin Heidelberg, 2010, vol. 6100, pp. 317–322. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16277-0_13

[40] fortiss af3, “Main Features- fortiss,” http://af3.fortiss.org/main-features/.