

# Virtualization Technologies for Cars

## Solutions to increase safety and security of vehicular ECUs

*Jan Pelzl, Marko Wolf, Thomas Wollinger  
escrypt GmbH – Embedded Security, [www.escrypt.com](http://www.escrypt.com)*

### Abstract

Virtualization basically means realizing several runtime environments in parallel but strictly isolated on a shared hardware. Nowadays, virtualization is an accepted standard and is used in the desktop and server market as a genuine alternative solution to several individual dedicated hardware systems. Through today's availability of modern and highly efficient virtualization solutions, virtualization becomes extremely interesting also for embedded applications.

The following work first gives a short introduction on virtualization technologies in general. It then presents possible advantages and possible implications as well as feasible fields of application and implementation examples for automotive vehicles. This work will introduce several concrete virtualization solutions and evaluate their feasibility in the vehicular area, before it concludes with a detailed summary and a short outlook for the chances of virtualization technologies in future vehicular ECUs.

## 1 Introduction

Until a few decades ago (virtually until the 1990s) cars were closed, electro-mechanical systems with only a few, isolated, and mainly uncritical IT systems. By contrast, current luxury-class vehicles contain a few tens of interconnected microprocessors with up to several hundred megabyte of software installed. In addition, various external communication ports have been integrated into a lot of these vehicles. Increasing the quantity of electronic control units (ECU) further and thereby also the network and maintenance complexity is neither technically nor economically justifiable [Co-ChAn02]. Consequently, future vehicular IT architectures will try to merge several single control units into a few powerful ones [Frisch04]. The parallel execution of several ECU applications allows for a noticeable more efficient and more flexible utilization of the always scantily hardware resources. Thus, it decreases efforts and costs during production, operation, and maintenance of (redundant) automobile IT hardware and necessary wiring. Furthermore, virtualization enables various novel vehicular IT mechanisms, which can increase the vehicle's safety as well as its security.

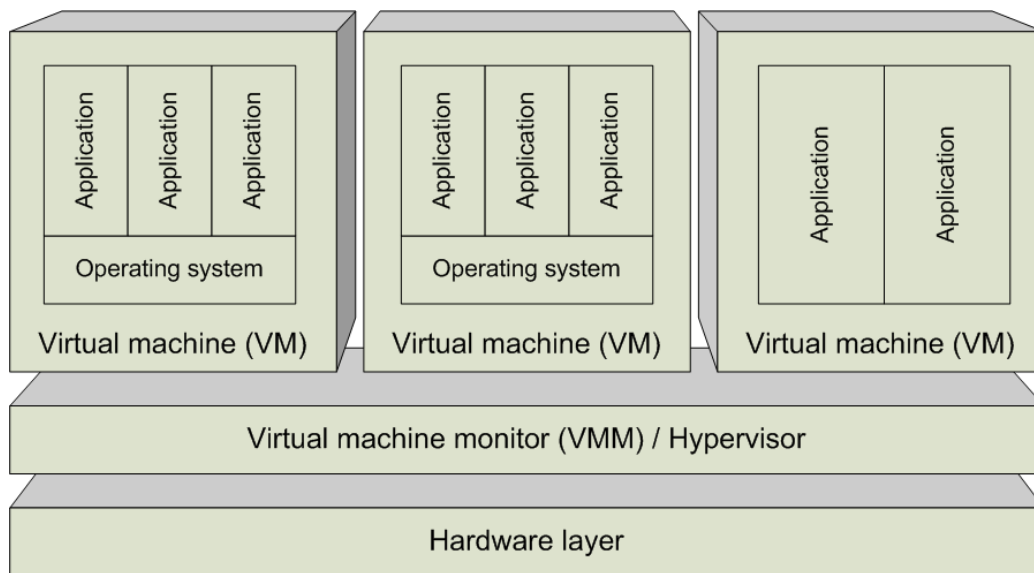
Nevertheless, reliable IT mechanisms are needed to secure ECU applications executed in parallel against each other. That means, neither an accidental malfunction (IT safety) nor a systematic manipulation (IT security) of one application should affect or compromise any other application executed in parallel. The virtualization technologies introduced in this paper, however, allow for an efficient and flexible hardware sharing while reliably enforcing the strict isolation of all parallel executed ECU applications.

The following work is structured as follows. First, the virtualization technology in general is shortly introduced. We present various advantages and some possible implications together with some feasible fields of application and some implementation examples for automotive vehicles. Then, several virtualization methods are concretely introduced and evaluated for their application and suitability.

in vehicular ECUs. The work concludes with a detailed summary and a short outlook for the chances of virtualization technologies becoming implemented in future vehicular ECUs.

## 2 Virtualization Technologies

As depicted in Figure 1, the concept of virtualization is based on an additional abstraction layer, called the *Virtual Machine Monitor (VMM)* or *Hypervisor* that is situated between the hardware layer and the operating system(s) or application(s). In practice, this abstraction layer can be realized in hardware, in software, or by a hardware/software combination. The main task of the VMM is to enable the sharing of the real physical resources with all existing runtime environments executed in parallel, called *Virtual Machines (VM)*, without causing any resource conflicts or inconsistencies; in one word: to *virtualize*. The utilization of the virtualized hardware resources, in turn, has to be transparent for each VM in a way that it can be executed in almost the same manner as a single individual process on its dedicated hardware. The mutual strict isolation, the access control to all shared hardware resources and the control of the VMs itself is managed by the VMM. That means, the VMM implements all effective access policies for all communications, applications, and data as well as for all shared hardware resources. Therefore, the VMM is the actual crucial component in all virtualization concepts for realizing and enforcing the operational IT safety and the IT security as well.



**Figure 1: Structure of a virtualized IT Architecture**

### 2.1 Benefits and Advantages

The adaption of virtualization technologies enables a series of benefits and advantages for vehicular IT architectures, from which a few will be introduced in the following.

#### 2.1.1 Reduction of Hardware Costs, Increased Hardware Efficiency, and Increased Peak Performance

By migrating and merging several functions and applications into a shared hardware resource, hardware can be entirely saved or existing hardware can at least be utilized considerably more efficient. The shared usage of processor time, memories and peripherals increases hardware efficiency and decreases unnecessary redundancy, wiring, and maintenance efforts as well. It enables even more valuable synergy effects with respect to the production and operating costs, administration efforts, energy consumption, and error-proneness to mention just a few. Moreover, merging mul-

multiple ECUs into a single and hence very powerful multipurpose ECU provides a considerably higher peak performance potential, which is in particular useful for short-time computationally-intensive operations such as routing computations or security computations (e.g., digital signature operations).

### **2.1.2 Increased IT Safety**

The strong isolation of all runtime environments, which is essential to enable reliable virtualization solutions, clearly increases IT safety in comparison to the isolation mechanisms of usual embedded operating systems [EHHPR05]. Up to now, a faulty application or driver was enough to affect the safety of the whole underlying IT platform, whereas strictly isolated runtime environments cannot affect each other in case of a malfunction or crash of a particular runtime environment.

Furthermore, with the help of a virtualization solution, the usage of certain system resources (e.g., processor time, memory allocation) can be strictly controlled by the VMM. This allows guaranteeing availability and extent of certain system resources for a particular runtime environment and, for instance, can be used to guarantee certain real-time capabilities.

ECU manufacturers can deploy their software with an individual preconfigured runtime environment in the form of a preconfigured VM and by that minimize errors caused by a wrong configured runtime environment (e.g., by the OEM) or minimize hard-to-predict implications by other third-party applications.

Furthermore, some more sophisticated virtualization concepts are capable of moving an entire runtime environment (*virtual domains*) across hardware borders in case the currently executing hardware platform is malfunctioning or temporarily providing insufficient resources. Software units kept in “pause mode” without consuming any noteworthy resources, can quickly take over broken functionality (*hot stand-by*).

### **2.1.3 Increased IT Security**

The strong isolation of all runtime environments, which is essential to enable reliable virtualization solutions, can also increase the IT security level against different software attacks<sup>1</sup>. For example highly sensitive information (e.g., a cryptographic key) or highly sensitive processes (e.g., a PIN entry) can be isolated from other runtime environments in a way that (intentional or unintentional) malfunctions cannot compromise their critical information or functionality. To give an example for that, one can imagine a standard rich operating system using functions to decrypt and encrypt data without accessing the cryptographic keys itself by invoking the functionality of an extra runtime environment that is executed in parallel but strictly isolated from the invoking rich operating system. Thus, potential malware, misadjustments or malfunctions of the rich operating system cannot result into the exposure of sensitive information (here, the cryptographic keys). In combination with finely granulated and efficient access control mechanisms lots of manifold IT security solutions can be realized [GarWar07].

### **2.1.4 Multilevel-Security and Multilevel-Safety per ECU**

By the use of virtualization technologies processes and applications can (and usually have to) be executed with different individual levels of safety and/or security in parallel on a shared hardware without being able to affect each other. With the help of the VMM implementing appropriate access control mechanisms, highly critical execution environments (e.g., driving relevant applications), critical execution environments (e.g., comfort applications and driver assistance), as well as pro-

---

<sup>1</sup> Virtualization solutions usually can ward off software attacks only, since it usually does not incorporate any hardware tamper-protection measures.

tected execution environments (e.g., multimedia and infotainment), and completely open execution environments (e.g., internet and personal user applications) can be – in comparison to the isolation mechanisms of usual embedded operating systems [EHHP05] – highly protected, controlled, and strictly isolated from each other (*strong isolation*).

Inside of every VM in turn, arbitrary access control mechanisms can be applied. Thus, there is no need for highly complex and extensive access control mechanisms being automatically applied to all applications executed on a single control unit. If some highly critical application is being executed on a shared ECU, the corresponding internal access controls can be individually adapted and executed according to the individual requirements of the individual VM. The (usually less complex) overall access controls between the parallel VMs, however, can be realized highly efficient by the VMM.

### **2.1.5 Increased Flexibility, Interoperability, and Backwards-compatibility**

The possibility to execute several guest-operating systems and thereby several applications for different kinds of operating systems simultaneously and in coexistence on one hardware platform clearly increases flexibility, interoperability, and backwards-compatibility of the respective IT platform. Existing applications could be easily migrated (to some extent even during runtime) to the platform in order to adapt functionality of platform for manifold different situations. Older, already existing legacy applications can be executed in their (old) preconfigured runtime environments parallel to new applications in their up-to-date runtime environments without the need for complex porting procedures (if possible at all). Common desktop applications (office software, email clients, media player, etc.) can be executed without any further modifications parallel to classical automotive applications within a vehicle.

## **2.2 Possible Disadvantages**

The adaption of virtualization technologies for vehicular IT architectures could also induce some disadvantages. The probably most important ones will be briefly described in the following.

### **2.2.1 Performance Overhead**

Depending on the implemented virtualization solution (cf. Section 3), a small performance overhead due to the additional redirection(s) and additional verification (*traps*) of system calls, interrupts, and I/O-accesses can be noticed in comparison to solely native executions without any VMM. The extent of such a performance overhead may vary heavily depending on the actual virtualization solution and the actual hardware basis. For virtualization solutions, which, for instance, are mainly based on specific hardware virtualization extensions (e.g., AMD-V or Intel-VT), the performance overhead can be reduced almost below the detection limit.

Nevertheless, a certain minimal hardware performance is necessary to implement a virtualization platform efficiently at all. Thus, even though it should be possible to implement virtualization solutions also on 8-Bit or 16-Bit hardware architectures, the performance overhead probably would be out of scale.

### **2.2.2 Lesser Physical Redundancy**

In comparison to multiple dedicated ECUs, a virtualization solution merging several ECU applications into a single shared hardware platform – by definition – also reduces the physical redundancy. Thus, a single hardware failure could inherently affect more functionality and more applications, as in case of manifold dedicated ECUs. Nonetheless, there exist already a series of adequate solutions, like *live migration* (cf. Section 2.1.2), to counteract these constraints.

### 2.2.3 Mandatory Access Control

When applying a virtualization solution with several VMs, additional functionality (for hardware and/or software) is necessary, which synchronizes parallel accesses to (physically) limited resources and enforces effective security and safety policies.

### 2.2.4 Little Practical Experience

As with all new emerging technologies, there initially exists only little practical experience in the application of virtualization technologies within vehicular IT environments. However, establishing virtualization solutions in existing vehicular software architectures inherently requires new technical know-how (e.g., partitioning, dependency and collaboration management, timings, etc.) and new (software) engineering measures (e.g., policy engineering, decision processes, or workflows).

## 2.3 Vehicular Application Examples

In the following a short overview about possible areas of application of virtualization solutions will be given in accordance with the former mentioned advantages and areas of application.

### 2.3.1 Possible Fields of Application

ECUs, which today already execute several tasks in parallel, can at most benefit from the additional possibilities and distinguishing characteristics of current virtualization solutions. These are in particular the Head-Unit (HU), the Front-Electronic-Module (FEM), the Rear-Seat-Entertainment-Unit (RSE), and the central gateway (ZGW).

### 2.3.2 Possible Use Cases

The strong isolation of ECU applications executed in parallel and the possibility for multi-level-safety and/ or multi-level-security on a single ECU makes virtualization technologies predestined foremost for the strict isolation of critical driving-relevant applications (e.g., rear view camera, automatic parking system) from less critical and therefore more open and flexible infotainment applications (e.g., media center, internet, mobile office, personal user applications).

Moreover, virtualization technologies can be applied to isolate critical security functionalities (e.g., encryption and verification procedures) and critical security applications (e.g., PIN entry, mobile commerce, driver authentication) from the other non-security-related applications or to dissolve the security functionality from existing applications. Thus, all sensible information involved in such security critical procedures and applications (e.g., cryptographic keys, PIN numbers, or authentication credentials) will be processed isolated from the actual functional procedure or application and hence, cannot be compromised by potential security holes in the application itself or its actual execution environment.

The possibility to reliably enforce even complex usage guarantees on critical hardware resources by “virtualizing” these hardware resources enables to strictly portion for example the communication resources within the ZGW or to portion the memory and process resources inside the HU or the FEM.

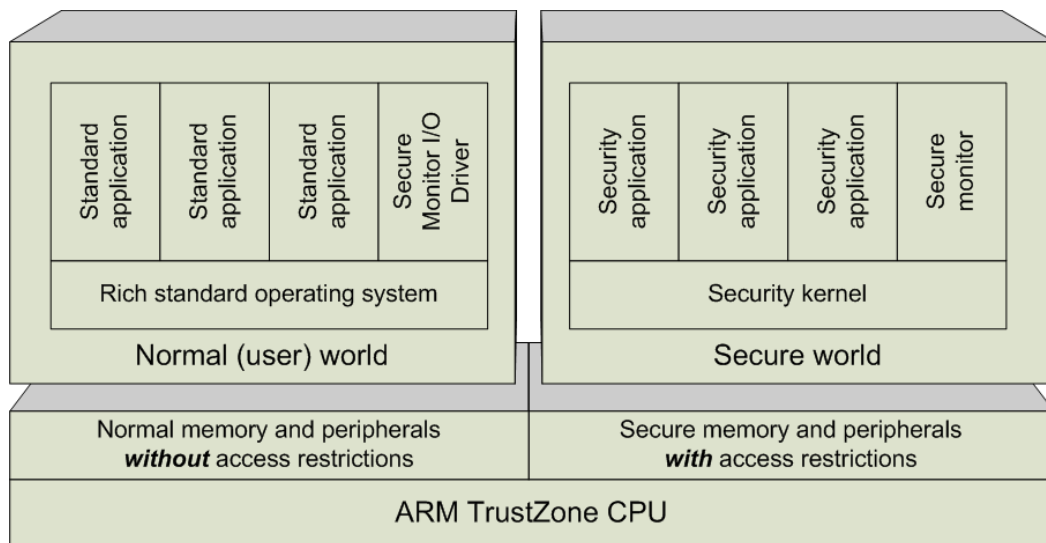
At last, one valuable use case for virtualization are the possibilities offered by the *live migration* of applications and functions from malfunctioning ECUs or ECUs that are temporarily providing insufficient resources to "spare" ECUs with available resources. Thus, every ECU with some idle resources can act as a redundant emergency ECU.

### 3 Methods of Virtualization

In the next section the three currently most important technologies for virtualization will be introduced shortly and then evaluated regarding their applicability within a vehicular IT environment.

#### 3.1 Physical Virtualization

Physical virtualization solutions are based on the physical partitioning of the runtime environment and thus are based on an (at least partly) physically implemented VMM. Physical virtualization can be realized through a multitude of different hardware mechanisms. For instance, certain hardware modules can be implemented redundantly (e.g., multi-core processors) or, as shown in Figure 2, common hardware units (e.g., the memory controller) are specially extended to realize several isolated runtime environments. The most popular virtualization implementation within the embedded area that is based on hardware extensions is probably the ARM *TrustZone* technology. The ARM TrustZone (processor) technology divides the processor, memory and all connected peripherals into two physically isolated domains (*secure world*, *normal world*). Both *worlds* can communicate with each other only by applying an additional and specially secured service (*secure monitor*) together with a new microprocessor instruction call (*secure monitor call*).



**Figure 2: Physical partitioning into a so called “normal world” and a thereof isolated “secure world” realized by the ARM TrustZone processor extension.**

As long as the VMM's hardware mechanisms are implemented thorough and correctly, the isolation mechanisms cannot be circumvented by software measures and therefore can provide a highly reliable level of isolation. In addition, VMM mechanisms implemented in hardware can be very efficient, since the loss of performance due to VMM mechanisms implemented in hardware is expected to be rather small in contrast to pure software implementations. Otherwise, hardware based VMMs are quite expensive and, by definition, rather inflexible than software implementations. Furthermore, hardware VMMs can only control and hence only virtualize resources, which provide the necessary hardware isolation mechanisms. The level of granularity of a solely hardware based virtualization solution is therefore generally limited.

#### 3.2 Hypervisor Virtualization

Virtualization based on a so called *hypervisor*, as shown in Figure 1, can be seen as the "classical form" of virtualization. Here, the VMM is being realized by such a hypervisor software component that usually refers to a minimized host operating system, which is exclusively executed in the pro-

cessor's kernel mode (i.e., ring 0 or *supervisor mode*). The hypervisor software handles and controls all accesses to the virtualized resources. Hence, the hypervisor enforces the strong isolation between individual VMs and implements all necessary mutual access control mechanisms. Popular examples for kernel mode based virtualization solutions within the desktop and server market are XEN [Barham03], KVM [KKLLL07], or microkernel based implementations [Liedtke95]. Within the embedded area, particularly virtualization solution based on microkernels such as OKL4 [OKL] and Integrity [GHS] are becoming (also commercially) very successful.

By trying to minimize the amount of code of the hypervisor executed in kernel mode<sup>2</sup>, potential implementation errors that may lead to breaches of the protection mechanisms for process isolation or access control can be minimized as well. The code size of especially small hypervisors (e.g., microkernels) can be reduced down to a level, where even a formal verification (i.e., a formal proof) of the correctness of the implementation is possible [SDNSM04].

Nevertheless, the level of isolation may be somewhat lower in comparison to pure hardware based isolation mechanisms. Otherwise, hypervisor based virtualization solutions are inherently highly flexible and quite inexpensive in comparison to static hardware realizations (if available at all). The typical performance overhead is negligibly small and varies somewhere between 3% and 10% [YWGK06]. For historic reasons, there basically exist two different solutions for hypervisor based virtualizations. The older approach called *paravirtualization* as well as the more current approaches based modern processor virtualization extensions will be introduced shortly in the following two subsections.

### 3.2.1 Paravirtualization

Paravirtualization is a hypervisor based virtualization solution without the need for special processor virtualization extensions such as Intel-VT or AMD-V. In order to handle non privileged and therefore not automatically caught system calls of VMs executed in parallel, all VMs have to be specially prepared in a way such that all *critical instructions* [RobIrv00] are redirected to the hypervisor first, instead of being executed immediately and directly on hardware. This requires adapting the source code for all software components calling these critical instructions. However, modifying the affected software components (i.e., mainly some base components of the guest operating system), requires access to the source code of the respective software component and requires also the continuous adaption of each newly released version. In addition, catching and handling all critical instruction induces a small performance overhead.

### 3.2.2 Hardware-based Virtualization

Considering the drawbacks of paravirtualization, modern processors already include some additional extensions for virtualization (e.g., Intel-VT or AMD-V), which implement a few parts of the VMM in hardware. Thus, conflicts in connection with the known critical processor instructions [RobIrv00] can be handled very efficiently. Having a virtualization extension available, most critical processor instructions can be called without being redirected or handled by the (software) hypervisor first. This means, existing software can be executed unmodified within a VM without incurring any performance penalties due to the mandatory redirection of all critical processor instruction calls.

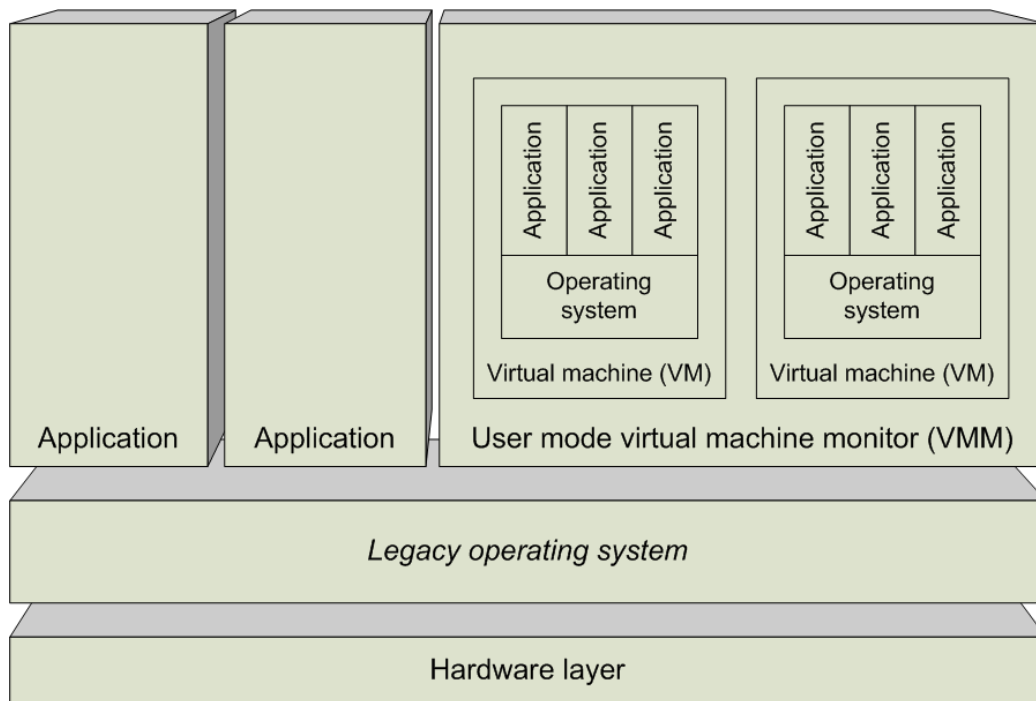
---

<sup>2</sup> Normally, the code size of a hypervisor is less than 10% of the code size a usual Linux or Windows based operating system kernel.

### 3.3 User Mode Virtualization

User mode virtualization (also called software virtualization or application virtualization) refers to all VMMs being executed as pure software implementations in *user mode* (i.e., ring > 0) normally on top of a common standard operating system as shown in Figure 3. In contrast to a hardware simulator or a hardware emulator, where all hardware instructions are handled in software (and thereby also instructions not being implemented on the hosts hardware can be simulated or emulated), some user mode VMMs are able to execute most instructions directly on hardware. The guest software being executed within a user mode VM normally does not have to be modified. Nevertheless, since almost all hardware instructions and system calls have to be redirected through the (non-minimal) host operating system, considerable performance losses (e.g., in comparison to a physical or and hypervisor based virtualization) are usually inevitable.

The crucial disadvantage of a user mode virtualization approach; however, is the comparatively low gain of IT security and IT safety. Both, security and safety of all software executed inside of a user mode VMM directly depends on the security and safety of the underlying host operating system, which is normally some traditional monolithic one (e.g., Windows or Linux). The user mode VMM and all executed VMs are therefore in the same manner prone to security holes and reliability problems as the underlying operating system itself. Thus, all potential weaknesses and all concurrently executed (malicious or faulty) processes of the host operating system may also affect the security and reliability of the executed VMs. Hence, a user mode VMM normally is used for the isolating their VMs against the host operating system and thus protecting foremost the host operating system. This is also known as *sandboxing*.



**Figure 3: Virtual Machine Monitor (VMM) in user mode.**

Popular examples for user mode virtualization approaches are VMware Workstation [VMW], Microsoft Virtual PC [Honeycutt03], or emulators like QEMU [Bellard05], and even the Java Virtual Machine [LinYe199], which is widely used also in the embedded area.

## 4 Summary and Outlook

Table 1 summarizes the preliminaries, the advantages, and disadvantages as well as some potential application areas in a vehicle for the virtualization solutions presented in this paper.

	Preliminaries	Advantages	Disadvantages	Application areas
<b>Physical virtualization</b>	Redundant hardware or additional hardware isolation functionality	Maximal performance and high level of isolation, normally no software modification necessary	Comparatively expensive, poor granularity and flexibility, only redundant or specially prepared hardware can be virtualized	High level security and high performance ECUs, (e.g., central gateway)
<b>Hypervisor with paravirtualization</b>	Software modifications of all virtualized runtime environments necessary	Efficient and flexible virtualization without need for a particular virtualization hardware	Software modifications may be costly or even impossible (e.g., <i>closed source</i> ), only existing hardware can be virtualized	Middle class ECUs without any particular hardware extensions for virtualization (e.g., FEM or RSE)
<b>Hypervisor with hardware supported virtualization</b>	Processor with compatible virtualization extension	Efficient and flexible virtualization without need for any software modifications	Processor extensions required that may cause extra costs, only existing hardware can be virtualized	High performance ECUs with hardware virtualization extensions (e.g., central multimedia/head unit)
<b>User mode virtualization</b>	Adequate rich host operating system necessary	Any hardware configuration including different processors can be simulated or emulated	Comparatively inefficient, security and safety of VM depends on the security and safety of the host OS	Isolation of untrustworthy applications within a trustworthy runtime environment (e.g., Internet access)

**Table 1: Summary of preliminaries, advantages, disadvantages, and some potential application areas for virtualization solutions in a vehicular IT environment.**

Even though, to the authors current knowledge, and except for a few sandboxing implementations for isolating applications inside an ECU (e.g., a GSM portal access application), none of the presented virtualization solution are actually deployed in current vehicles. Nevertheless, virtualization solutions are already a prevalent IT safety and IT security technology for most upcoming vehicular IT architectures (i.e., 2012ff). The important hardware independent and hence solely module- and function-oriented AUTOSTAR approach (“Automotive Open Software Architecture”) for an open, flexible, and standardized vehicular software architecture [ASAR], for instance, hardly can be realized without a reliable and efficient isolation mechanism. The ongoing integration of IT applications and (consumer) electronic devices into the vehicle, which yet are no longer under the full control of the corresponding OEM or supplier (e.g., user devices, user applications, or free Internet access) virtually cannot be realized with even more isolated and dedicated ECUs. The continuously ongoing migration of individual and dedicated ECUs into central, high performance, multipurpose ECUs also can be hardly realized efficiently, reliably, and securely without one of the virtualization solutions mentioned in this paper.

Regarding this, particularly the combination of hardware and software virtualization mechanisms (cf. Section 3.2.2) is able to provide a maximum on flexibility and efficiency. Currently the processor manufacturer Intel and the automotive Linux developer WindRiver are already cooperating on the development of an automotive Linux distribution that employs the hardware virtualization extensions of Intel’s Atom processor.

Thus, the application of virtualization solutions in vehicular IT environments will be virtually inevitable on the one hand, but on the other hand, offer also a great chance to considerably increase safety and security for all vehicular IT applications.

## References

- [ASAR] The Automotive Open System Architecture (AUTOSAR). [www.autosar.org](http://www.autosar.org).
- [Barham03] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*. New York, NY, USA, 2003.
- [Bellard05] F. Bellard. QEMU, a Fast and Portable Dynamic Translator. In *Proceedings of the USENIX Annual Technical Conference, FREENIX Track*, 2005.
- [CoChAn02] E. Coelingh, P. Chaumette, M. Andersson. Open-Interface Definitions for Automotive Systems – Application to a Brake-By-Wire System, Technical Paper 2002-01-0267, SAE International, March 2002.
- [EHHPR05] K. Elphinstone, G. Heiser, R. Huuck, S. M. Petters, S. Ruocco. L4Cars. In *3rd Workshop on Embedded Security in Cars (escar)*. November 2005.
- [Frisch04] H.-G. Frischkorn. Automotive Software – The Silent Revolution. In *Workshop on Future Generation Software Architectures in the Automotive Domain*. San Diego, USA, 2004.
- [GarWar07] T. Garfinkel, A. Warfield. What Virtualization can do for Security. In *The USENIX Magazine, Volume 32, Number 6*. December 2007.
- [GHS] Green Hills Software (GHS). The Integrity Embedded Operating System. [www.ghs.com](http://www.ghs.com).
- [Honeycutt03] Jerry Honeycutt. Microsoft Virtual PC 2004 – Technical Overview. Microsoft Corporation, November 2003.
- [KKLLL07] A. Kivity, Y. Kamay, D. Laor, U. Lublin, A. Liguori. KVM: The Linux Virtual Machine Monitor, In *Proceedings of Linux Symposium*. Ottawa, Canada, 2007.
- [Liedtke95] J. Liedtke. On Microkernel construction. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*. Copper Mountain, CO, USA, 1995.
- [LinYel99] T. Lindholm, F. Yellin. Java(tm) Virtual Machine Specification. Addison-Wesley Professional, 1999.
- [OKL] Open Kernel Labs (OKL). The OKL4 Embedded Operating System. <http://wiki.oklabs.com/>.
- [RobIrv00] John Scott Robin, Cynthia E. Irvine . Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor. In *Proceedings of the 9th USENIX Security Symposium*. Denver, Colorado, USA, August 2000.
- [SDNSM04] J. Shapiro, M. S. Doerrie, E. Northup, S. Sridhar, M. Miller. Towards a verified, general-purpose operating system kernel. In *Proceedings of NICTA FM Workshop on OS Verification*. National ICT Australia, 2004.
- [YWGK06] L. Youseff, R. Wolski, B. Gorda, C. Krintz. Evaluating the Performance Impact of Xen on MPI and Process Execution For HPC Systems. In *Proceedings of the International Workshop on Virtualization Technologies in Distributed Computing*, Tampa, FL, USA, 2006.
- [VMW] VMware Inc. VMware Workstation. <http://www.vmware.com/products/ws/>