« Cooperate on standards – compete on implementation »

Introduction à

AUTOSAR

EISTI

Novembre 2016

Ali Baktash

BMW Group · BOSCH · Continental · DAIMLER · Ford · GM · PSA GROUPE · TOYOTA · VOLKSWAGEN AKTIENGESELLSCHAFT

Déroulement de la formation

- La problématique du Logiciel embarqué
- Les différents Cycle de développement logiciels
- La qualité au rendez-vous de chaque étape
- Rentrons dans les détails, MISRA
- Règles de codages C-ANSI
- La qualité du logiciel, la norme SQUaRE
- Modèles de processus SPICE et CMMI

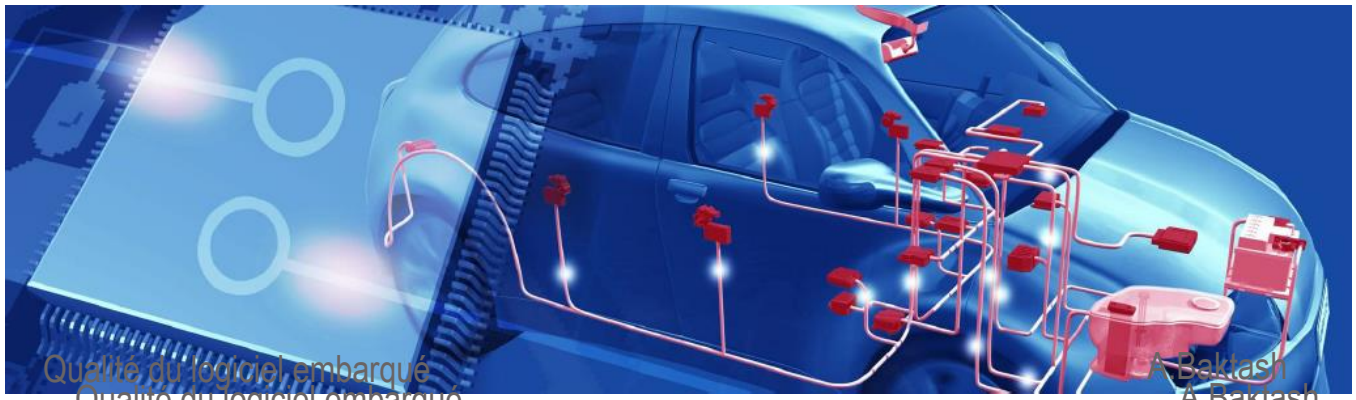
AUTOSAR

Agenda

- Introduction et historique du consortium
- L'architecture logicielle AUTOSAR
- La méthodologie AUTOSAR
- Quelques « solutions » AUTOSAR
- Conclusion

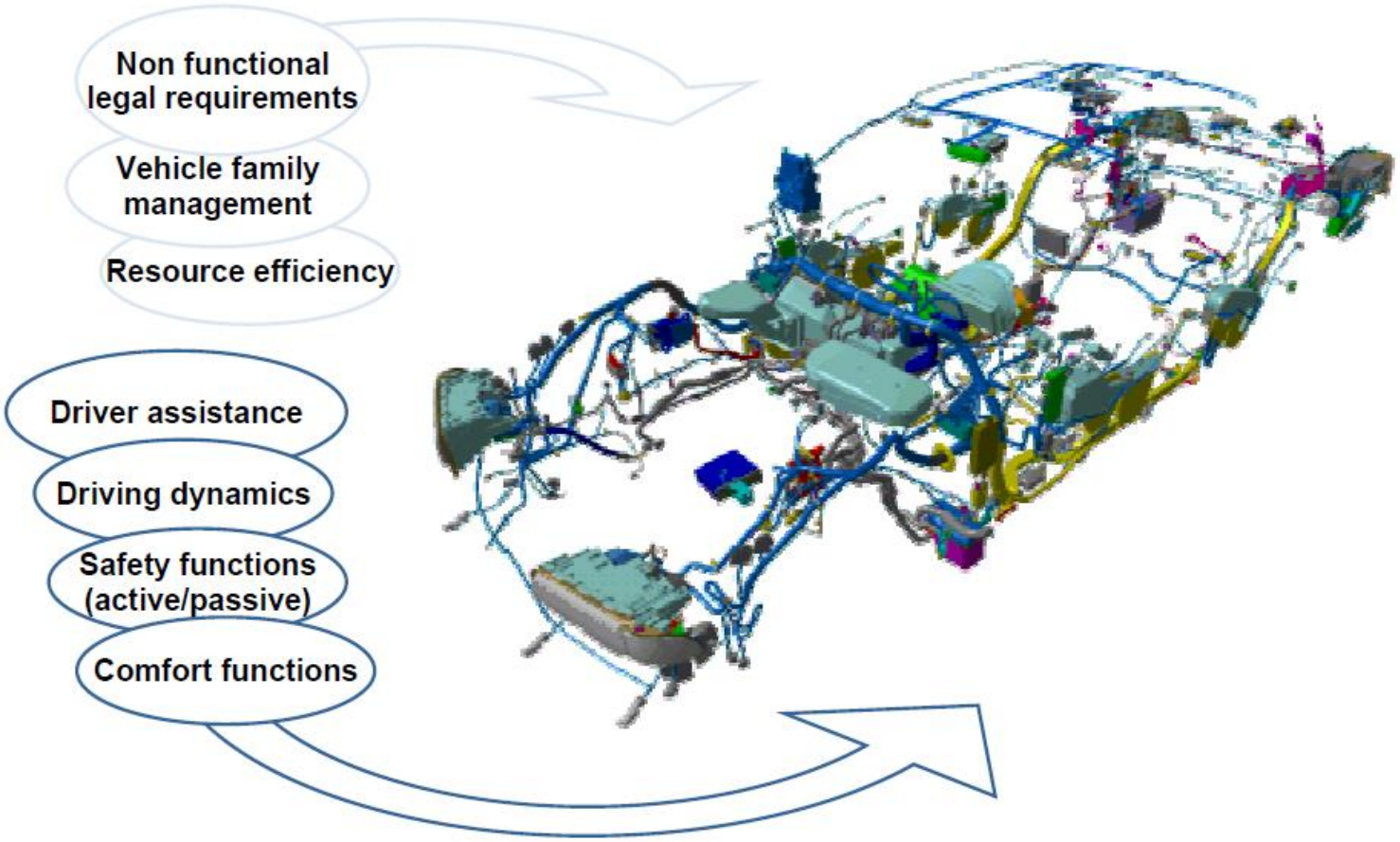
AUTOSAR AUTomotive Open System ARchitecture

- Alliance de constructeurs et d'équipementiers
- Objectif
  - Créer une architecture logicielle standard pour les calculateurs automobiles.
- Depuis juillet 2003 :
  - L'association travaille activement en différents groupes de travail afin d'élaborer l'ensemble des spécifications de l'architecture AUTOSAR.

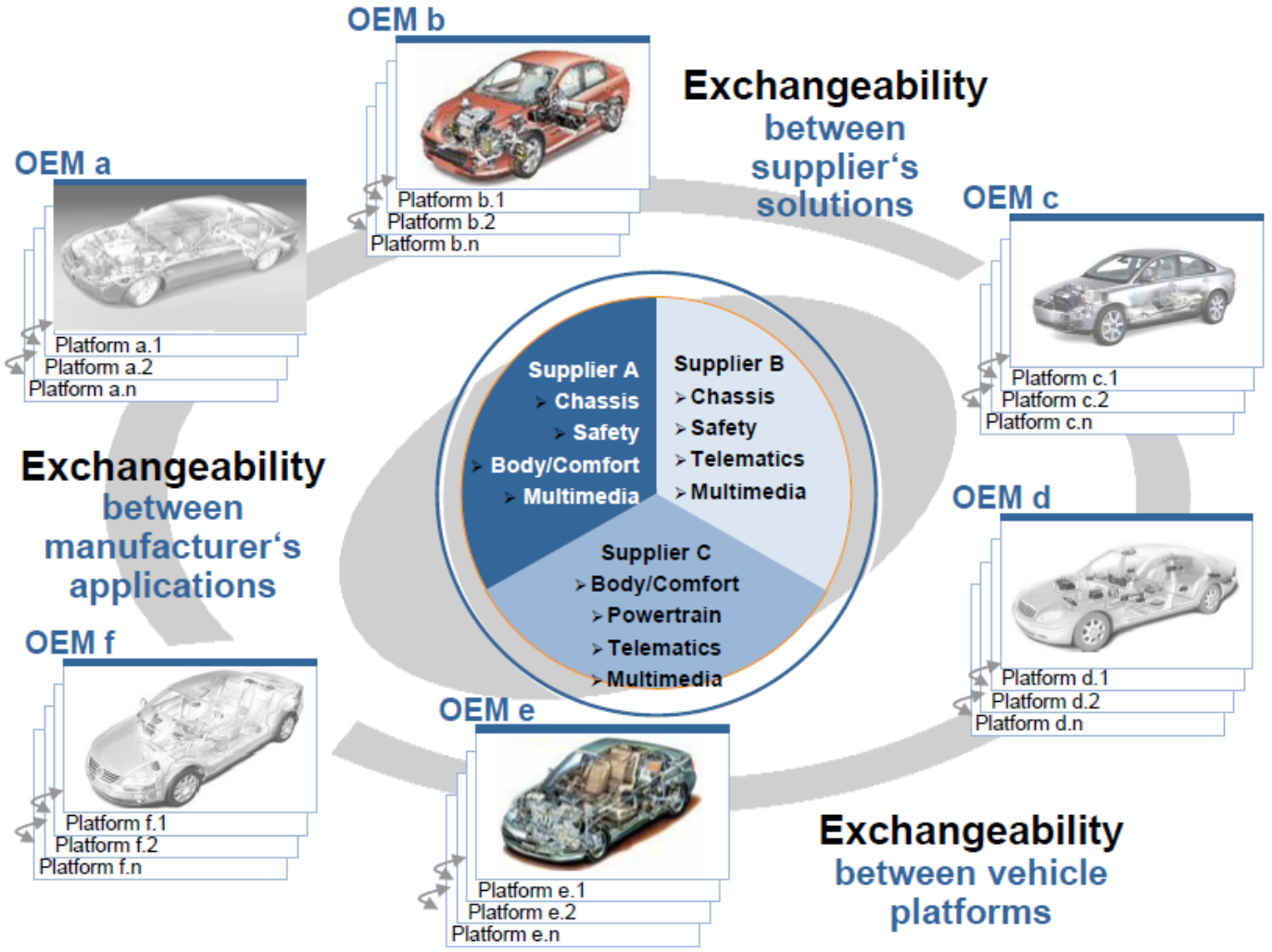


Qualité du logiciel embarqué A.Baktash

L’électronique et le logiciel dans l’industrie automobile

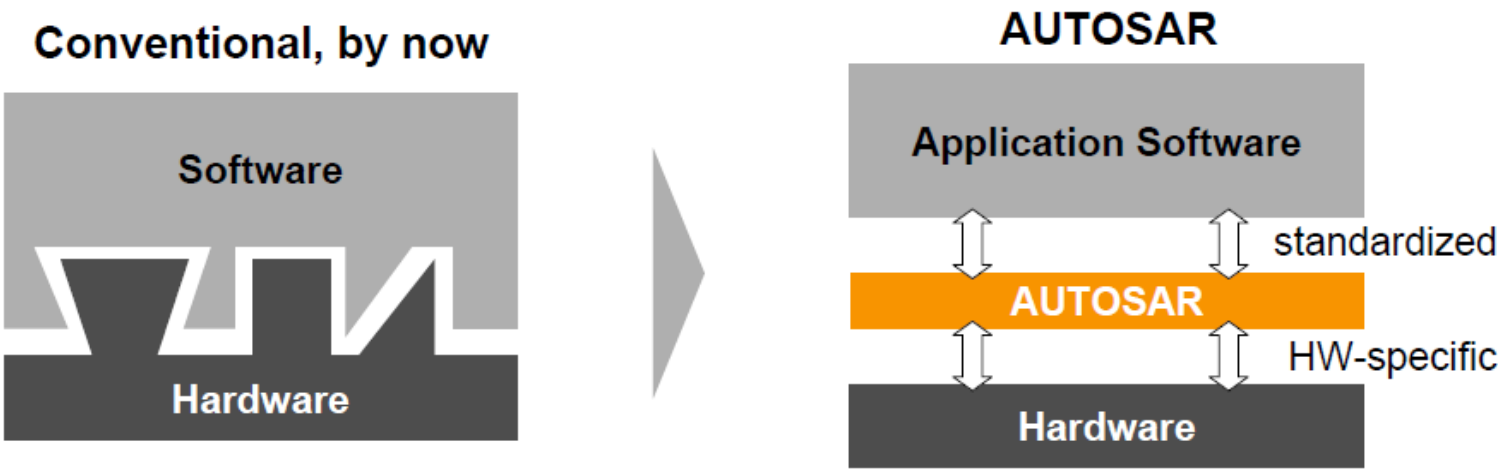


Question de réutilisation et de capitalisation…



Enjeux

Automotive Software Development will change.

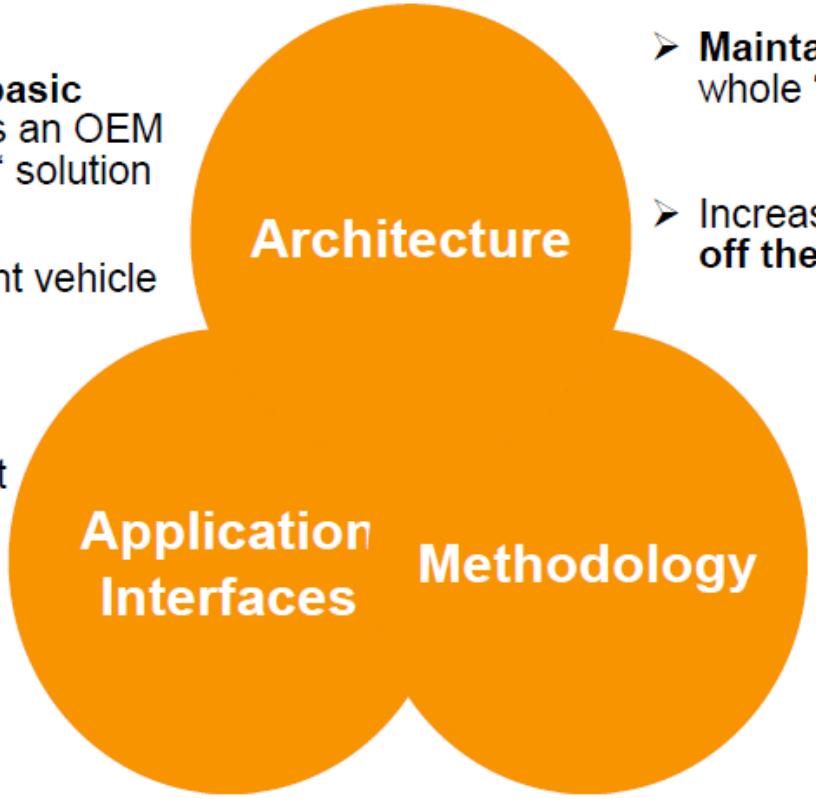


- **Hardware- and software** will be **widely independent** of each other.
- **Development processes** will be **simplified**.
  This **reduces development time** and **costs**.
- **Reuse** of **software increases** at OEM as well as at suppliers.
  This **enhances** also **quality** and **efficiency**.

Automotive Software will become a product.

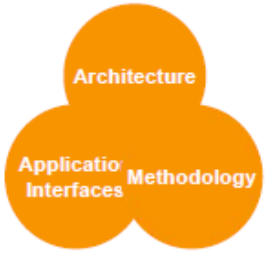
Objectifs du consortium

- Implementation and **standardization of basic system functions** as an OEM wide “Standard Core“ solution
- **Scalability** to different vehicle and platform variants
- **Transferability of functions** throughout network
- **Integration** of functional modules from **multiple suppliers**



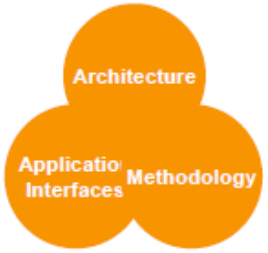
- **Maintainability** throughout the whole “Product Life Cycle“
- Increased use of **“Commercial off the shelf hardware“**
- **Software updates** and upgrades over vehicle lifetime
- Consideration of availability and **safety** requirements
- **Redundancy** activation

Principaux sujets d'étude



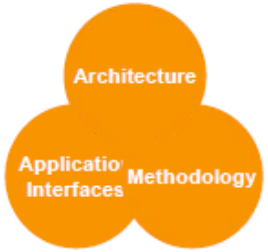
- Architecture:

  Software architecture including a complete basic or environmental software stack for ECUs – the so called AUTOSAR Basic Software – as an integration platform for hardware independent software applications.



- Methodology:

  Exchange formats or description templates to enable a seamless configuration process of the basic software stack and the integration of application software in ECUs and it includes even the methodology how to use this framework.

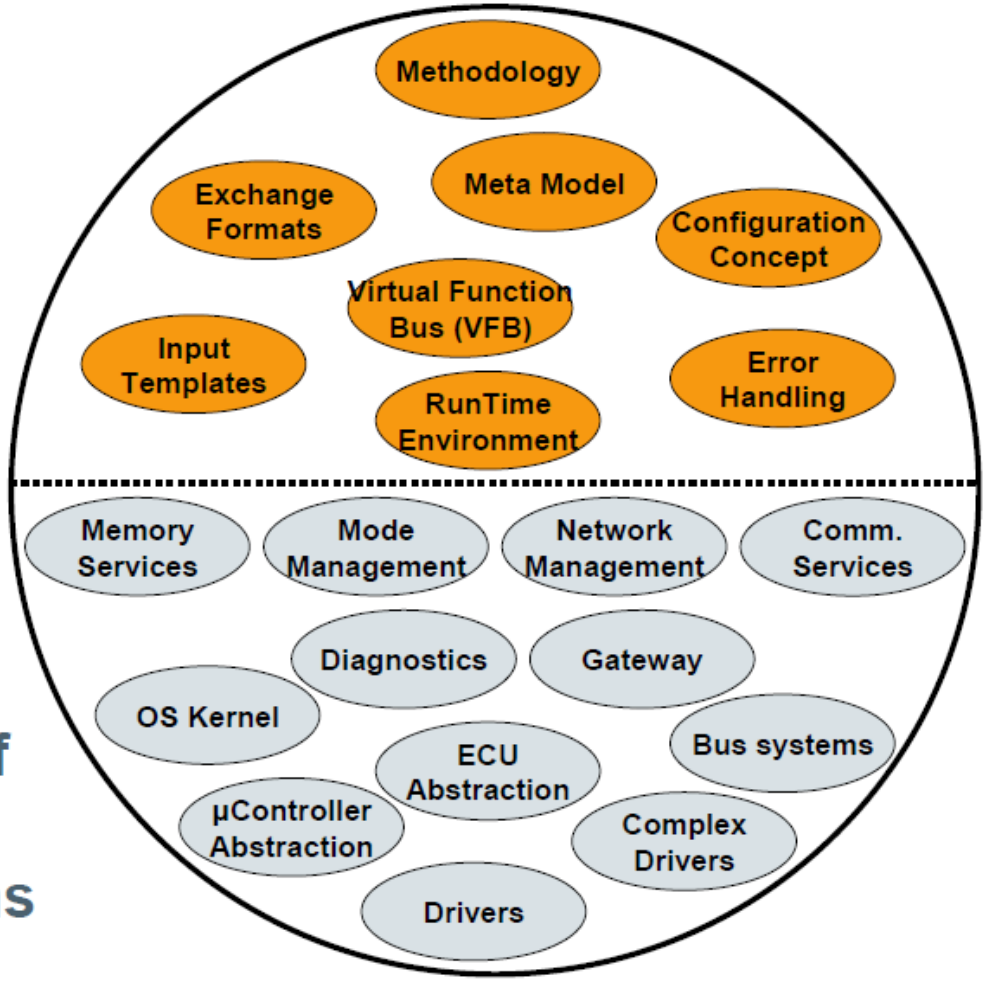


- Application Interfaces:

  Specification of interfaces of typical automotive applications from all domains in terms of syntax and semantics, which should serve as a standard for application software.

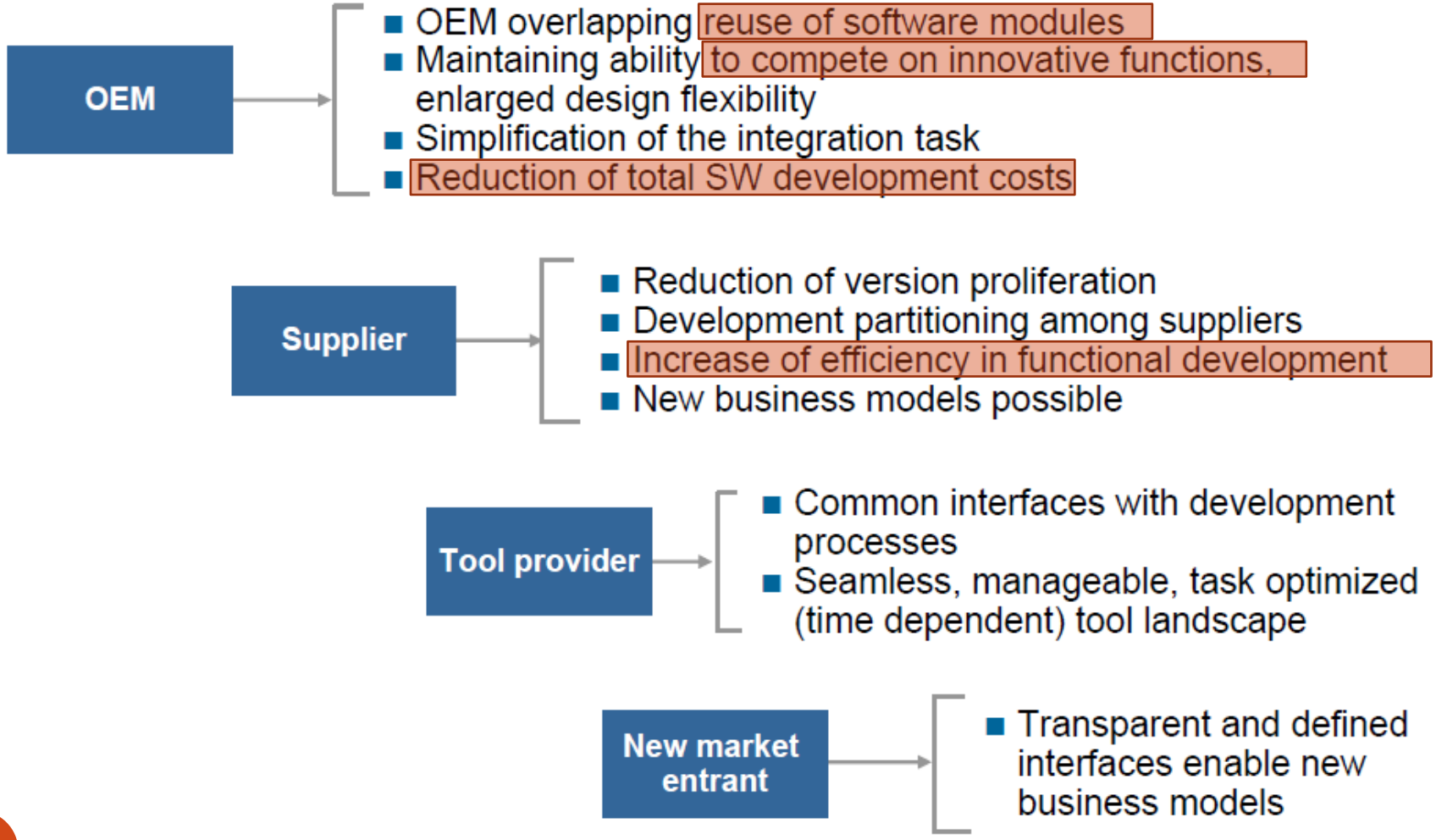
Périmètre technique

New concepts



Industry-wide consolidation of ‚existing' basic software designs

Principaux intérêts d'AUTOSAR



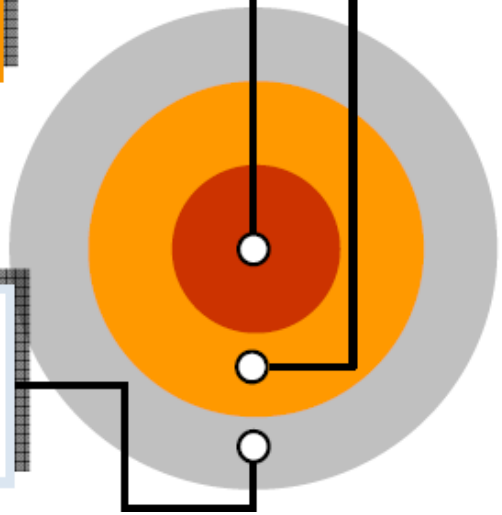
Consortium mondial

Core Partners
(OEM & Tier 1 Supplier)

- Organizational control
- Technical contributions
- Administrative control
- Definition of external Information (web-release, clearance, etc.)
- Leadership of Working Groups
- Involvement in Working Groups

Premium Members

- Leadership of Working Groups
- Involvement in Working Groups
- Technical contributions
- Access to current information



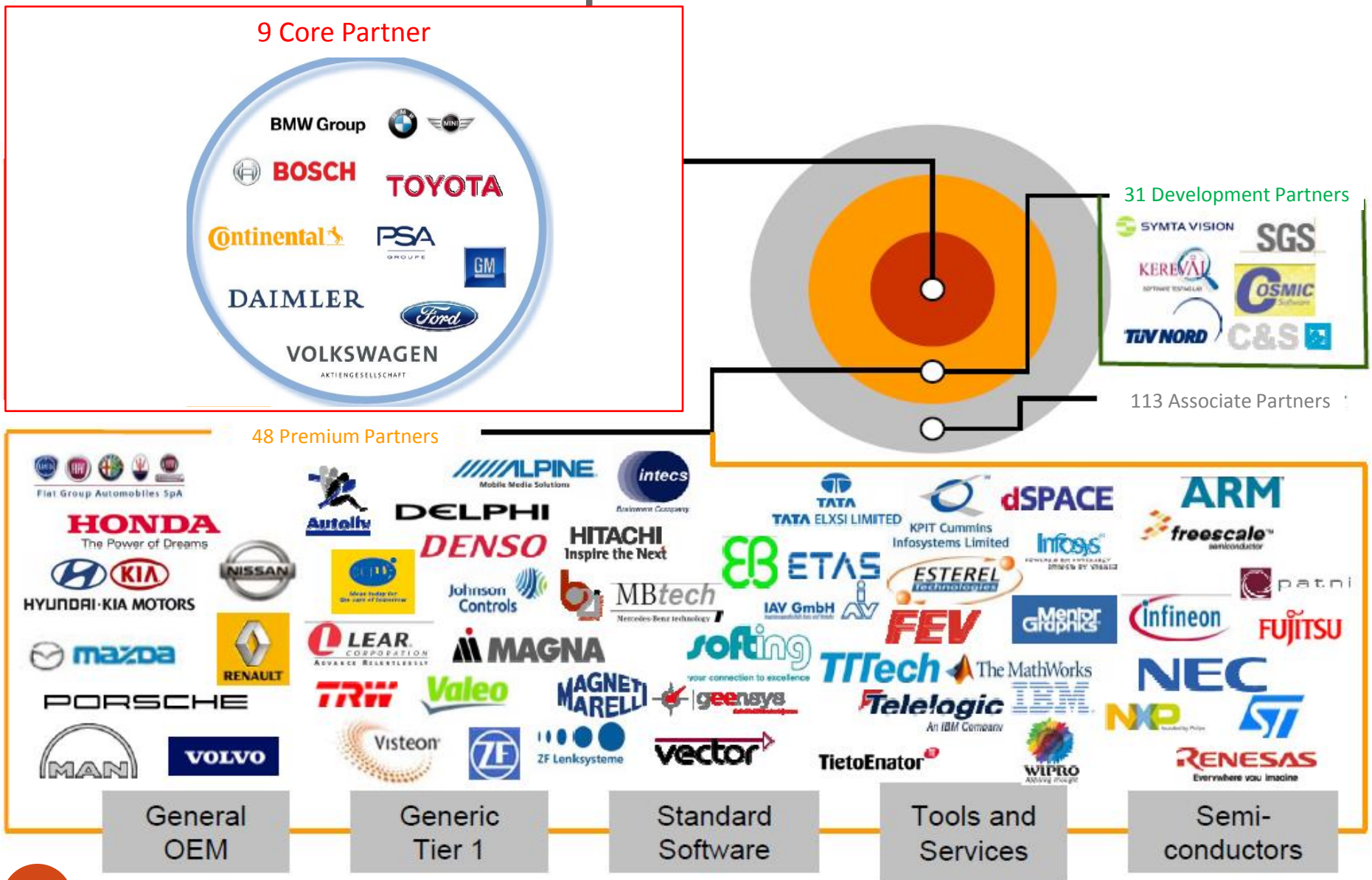
Associate Members

- Access to finalized documents
- Utilization of AUTOSAR standard

Support Roles

- *Development Members*
- *Attendees*

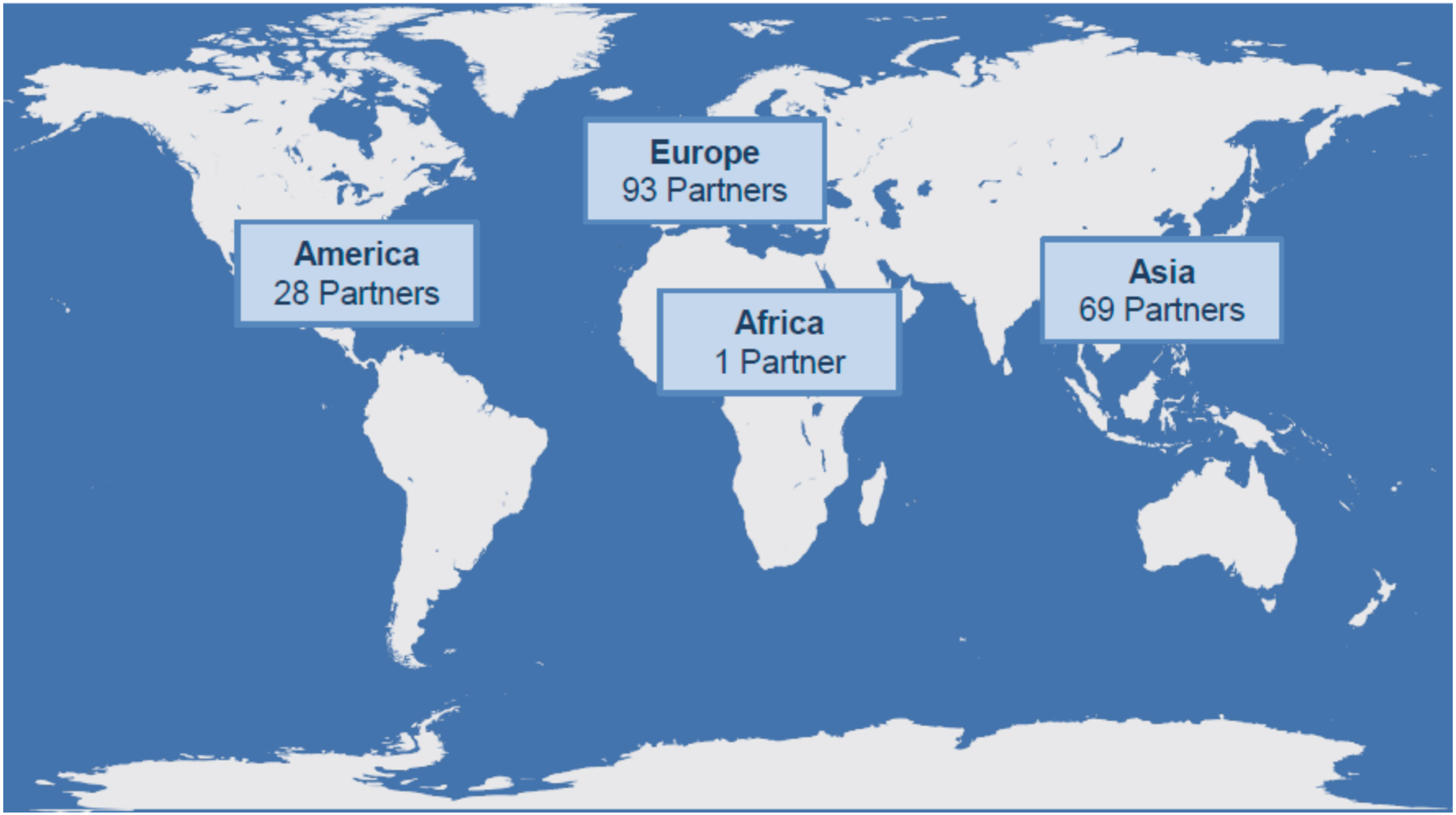
Les différents partenaires…



Status : Nov 2016

A.Baktash

Les partenaires dans le monde



Source : AUTOSAR.org 2015

Historique (1/4)

- Phase I (2003-2006)
  - AUTOSAR was founded as a development partnership in 2003
  - Release 1.0: AUTOSAR specifications for basic software below the runtime environment (RTE) level
  - Release 2.0 and 2.1: completion of Basic Software (BSW) components and the RTE
  - With Release 2.1 and Release 3.0/3.1, the majority of members started their series roll-out of AUTOSAR

Historique (2/4)

- Phase II (2007-2009)
  - Release 3.0: harmonization of the ECU wake-up and the network start-up
  - Release 3.1: incorporation of On-Board-Diagnostics (OBD) regulations support mechanisms
  - First ECUs based on AUTOSAR have entered production in 2008
  - Release 4.0: new features for functional safety and communication
  - Work on new car domains in the field of application interfaces: “Telematics/Multimedia/HMI” and “Occupants and Pedestrian Safety Systems”

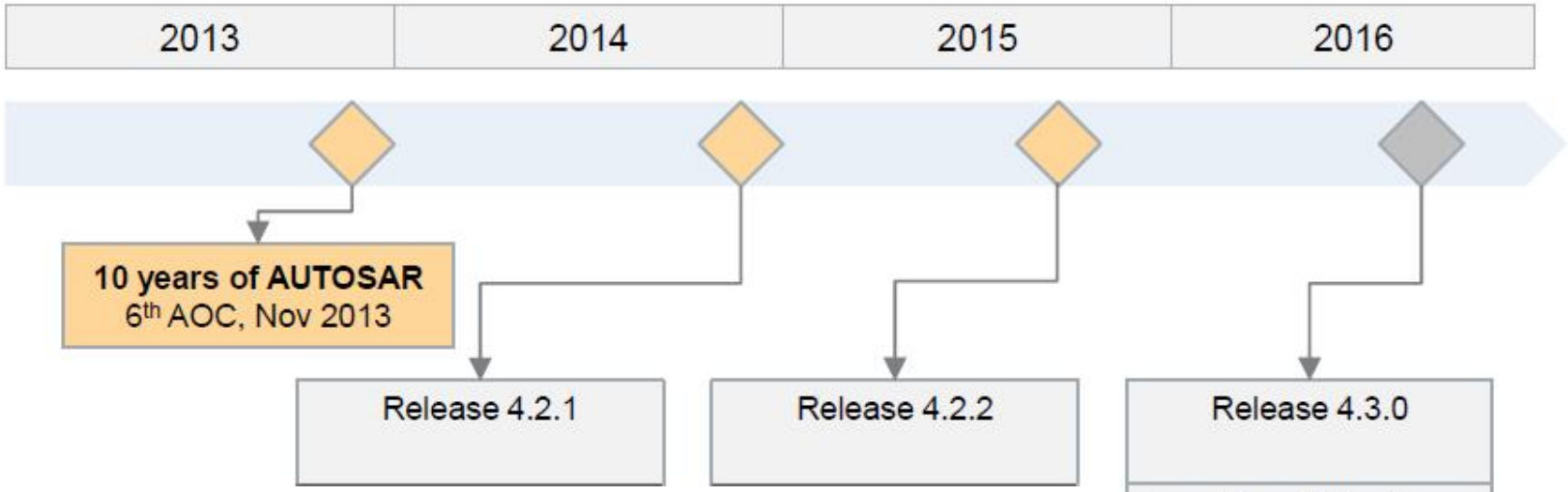
Historique (3/4)

- Phase III (2010-2012)
  - Release 3.2.1 and 4.0.3: incorporation of Partial Networking
  - General Objectives:
    - Maintenance of existing releases
    - Selective enhancement of the standard driven by market needs
    - Improve maintainability of the standard

Historique

- Phase I (2004-2006):
  - Développement de base de la norme (versions 1.0, 2.0 et 2.1)
- Phase II (2007-2009):
  - Extension de la norme en termes d'architecture et de méthodologie (versions 3.0, 3.1 et 4.0)
- Phase III (2010-2013):
  - Maintenance et améliorations spécifiques (versions 3.2, 4.1 and 4.2)
- Depuis 2013:
  - Specifications 4.2
  - Fonctionnement permanent :
    - Maintenir le standard
    - Améliorations spécifiques

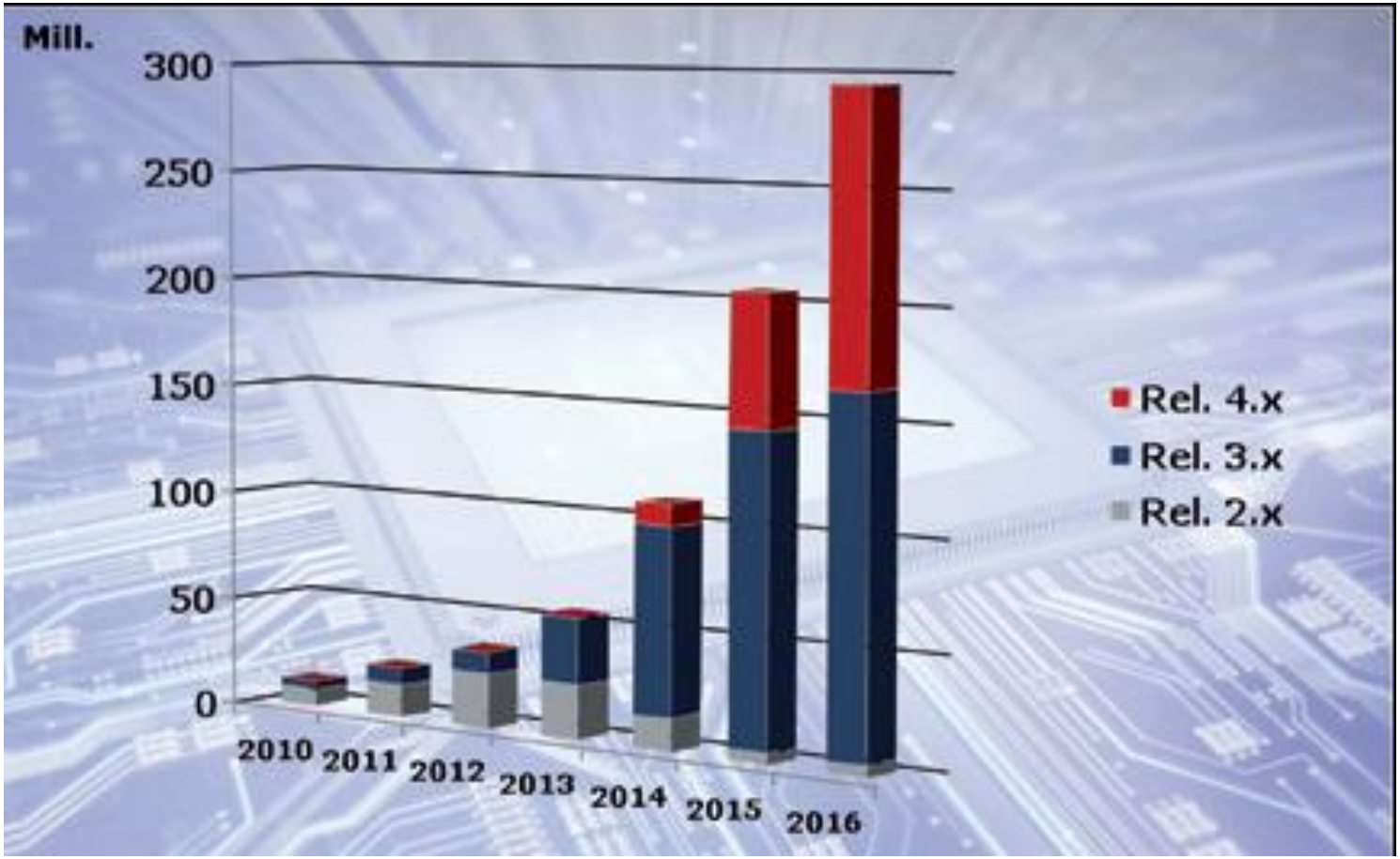
Historique (4/4)



- Post phase III (from 2013 onwards)
  - Selective, market-driven enhancements and extensions to the standard with focus on stabilization, limited number of parallel Releases and excellent match of series development requirements
  - Enhanced features for new technologies like multi-core processors, Ethernet/TCP/IP communication mechanisms and others

A.Baktash

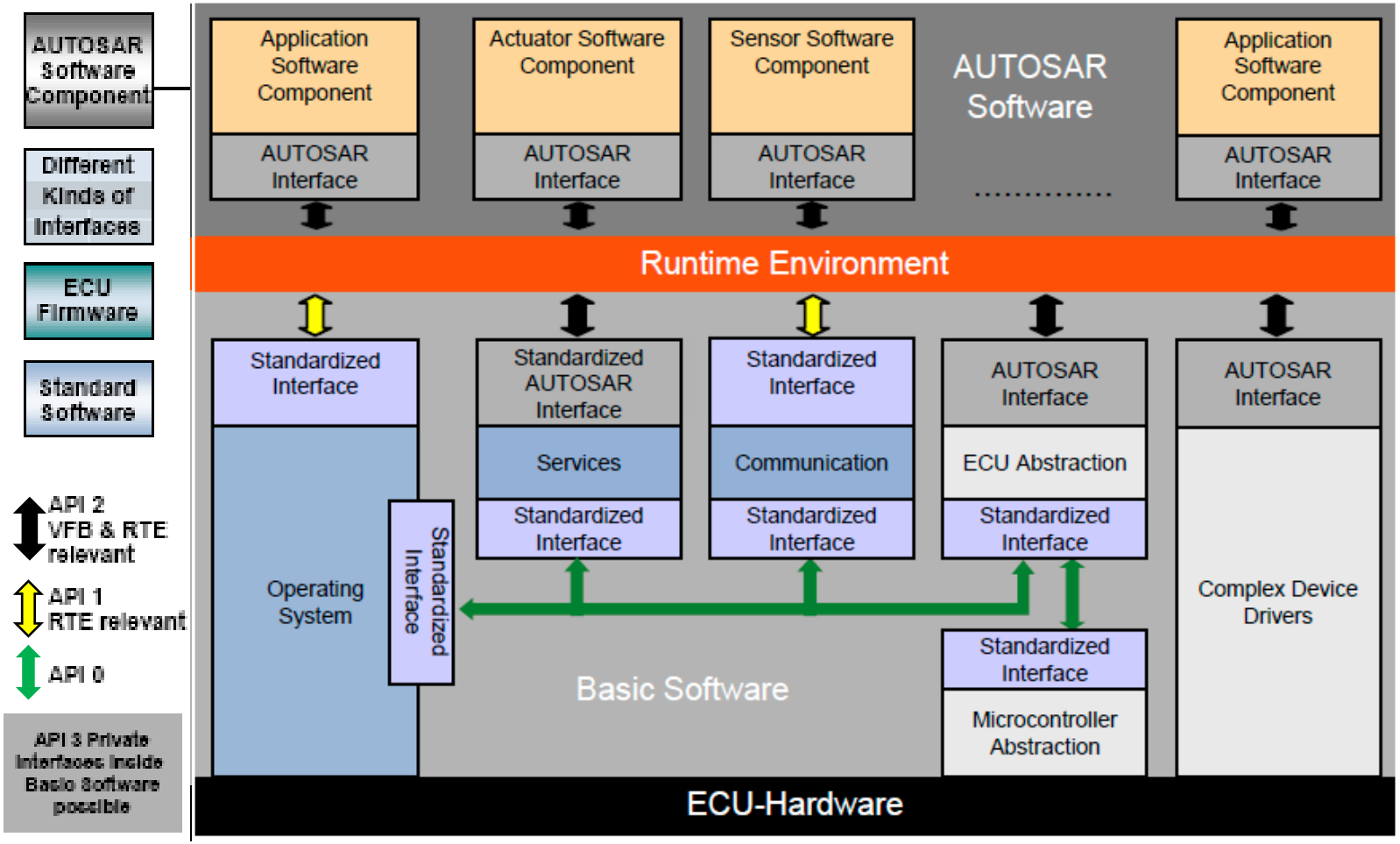
Nombre d'ECUs conformes AUTOSAR



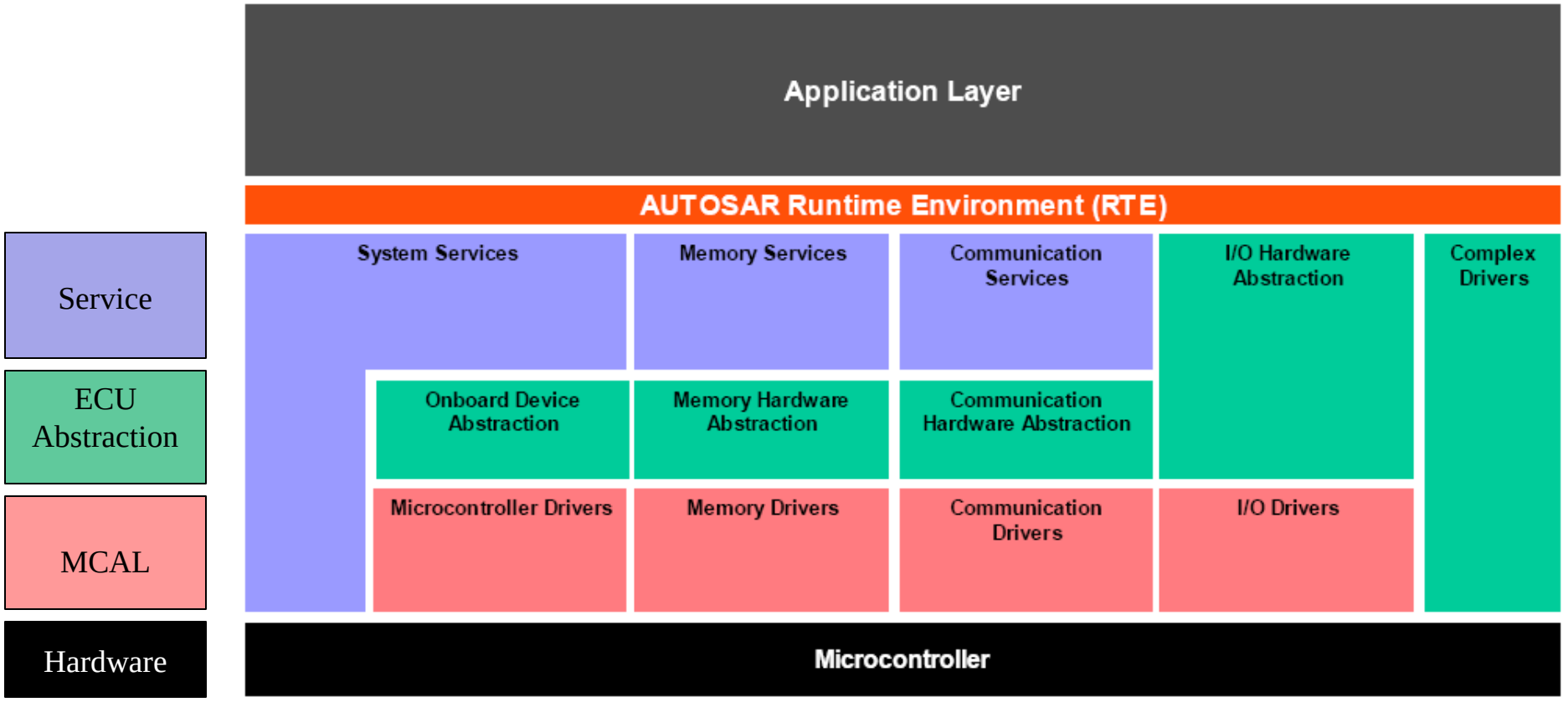
Agenda

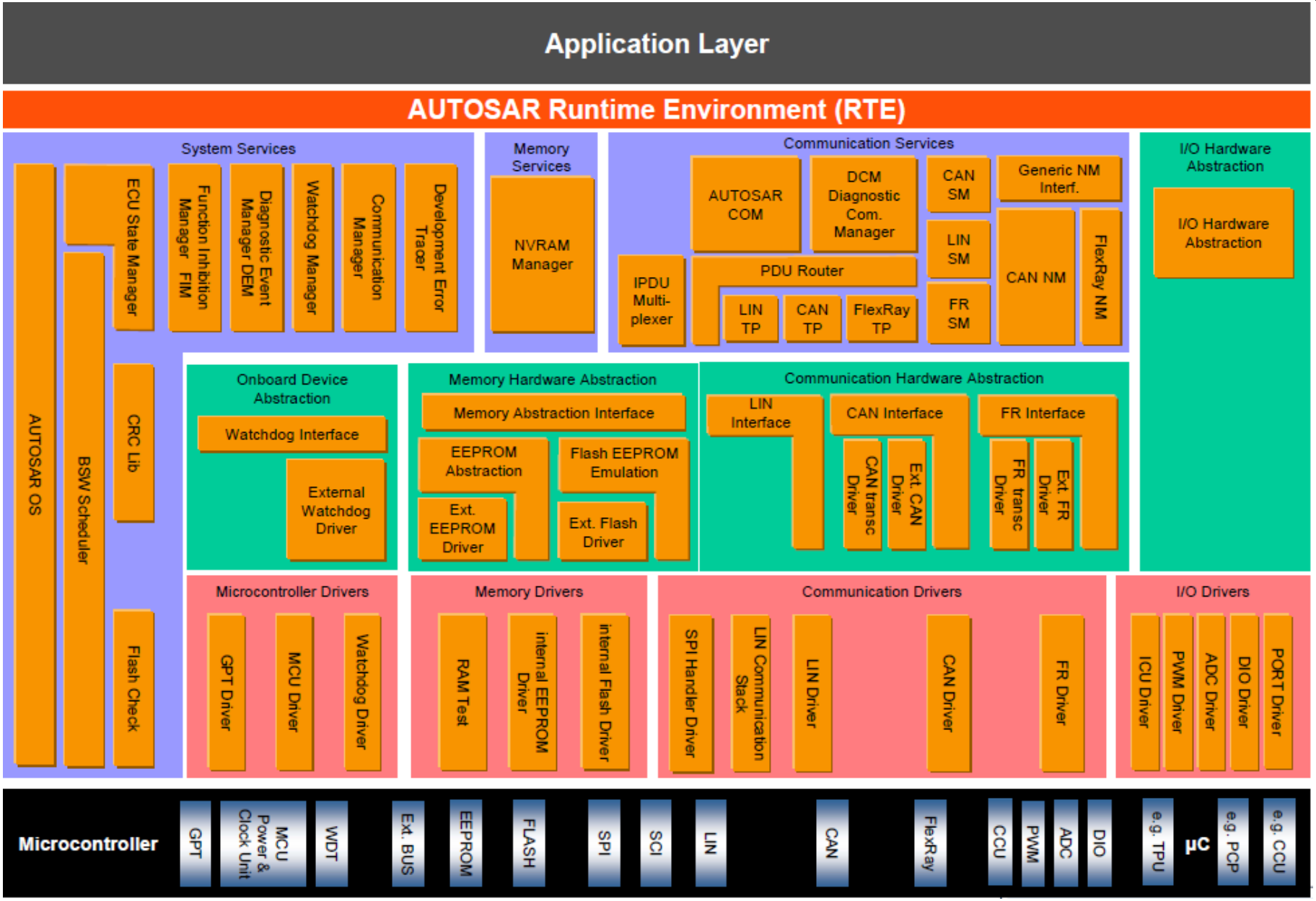
- Introduction et historique du consortium
- L'architecture logicielle AUTOSAR
- La méthodologie AUTOSAR
- Quelques « solutions » AUTOSAR
- Conclusion

L’architecture logicielle AUTOSAR

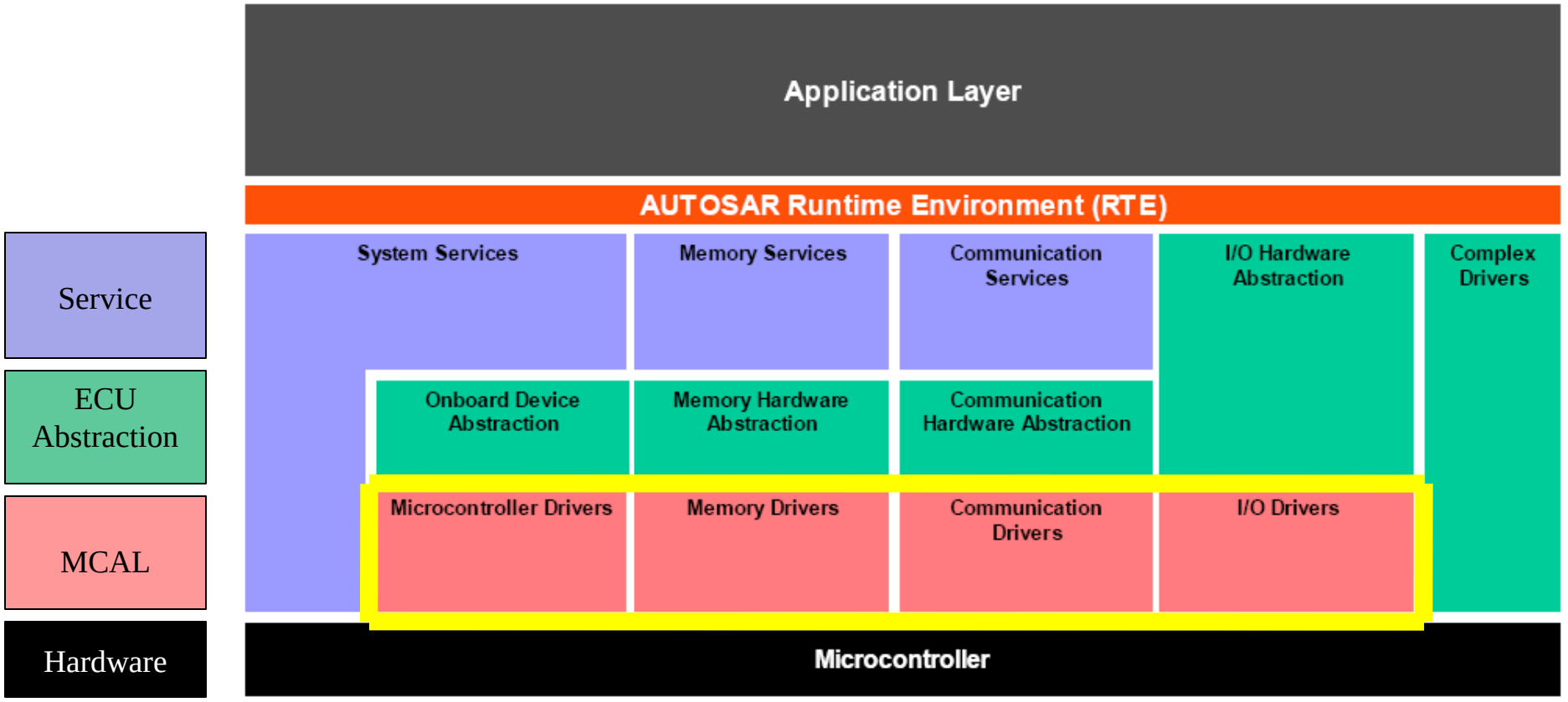


L’architecture logicielle AUTOSAR




Application Layer
AUTOSAR Runtime Environment (RTE)
System Services
AUTOSAR OS
BSW Scheduler
ECU State Manager
Function Inhibition Manager FIM
Diagnostic Event Manager DEM
Watchdog Manager
Communication Manager
Development Error Tracer
CRC Lib
Flash Check
Memory Services
NVRAM Manager
Communication Services
AUTOSAR COM
DCM Diagnostic Com. Manager
CAN SM
Generic NM Interf.
LIN SM
CAN NM
FlexRay NM
IPDU Multi-plexer
PDU Router
LIN TP
CAN TP
FlexRay TP
FR SM
I/O Hardware Abstraction
I/O Hardware Abstraction
Onboard Device Abstraction
Watchdog Interface
External Watchdog Driver
Memory Hardware Abstraction
Memory Abstraction Interface
EEPROM Abstraction
Flash EEPROM Emulation
Ext. EEPROM Driver
Ext. Flash Driver
Communication Hardware Abstraction
LIN Interface
CAN Interface
FR Interface
CAN transc Driver
Ext. CAN Driver
FR transc Driver
Ext. FR Driver
Microcontroller Drivers
GPT Driver
MCU Driver
Watchdog Driver
Memory Drivers
RAM Test
internal EEPROM Driver
internal Flash Driver
Communication Drivers
SPI Handler Driver
LIN Communication Stack
LIN Driver
CAN Driver
FR Driver
I/O Drivers
ICU Driver
PWM Driver
ADC Driver
DIO Driver
PORT Driver
Microcontroller
GPT
MCU Power & Clock Unit
WDT
Ext. BUS
EEPROM
FLASH
SPI
SCI
LIN
CAN
FlexRay
CCU
PWM
ADC
DIO
e.g. TPU
µC
e.g. PCP
e.g. CCU

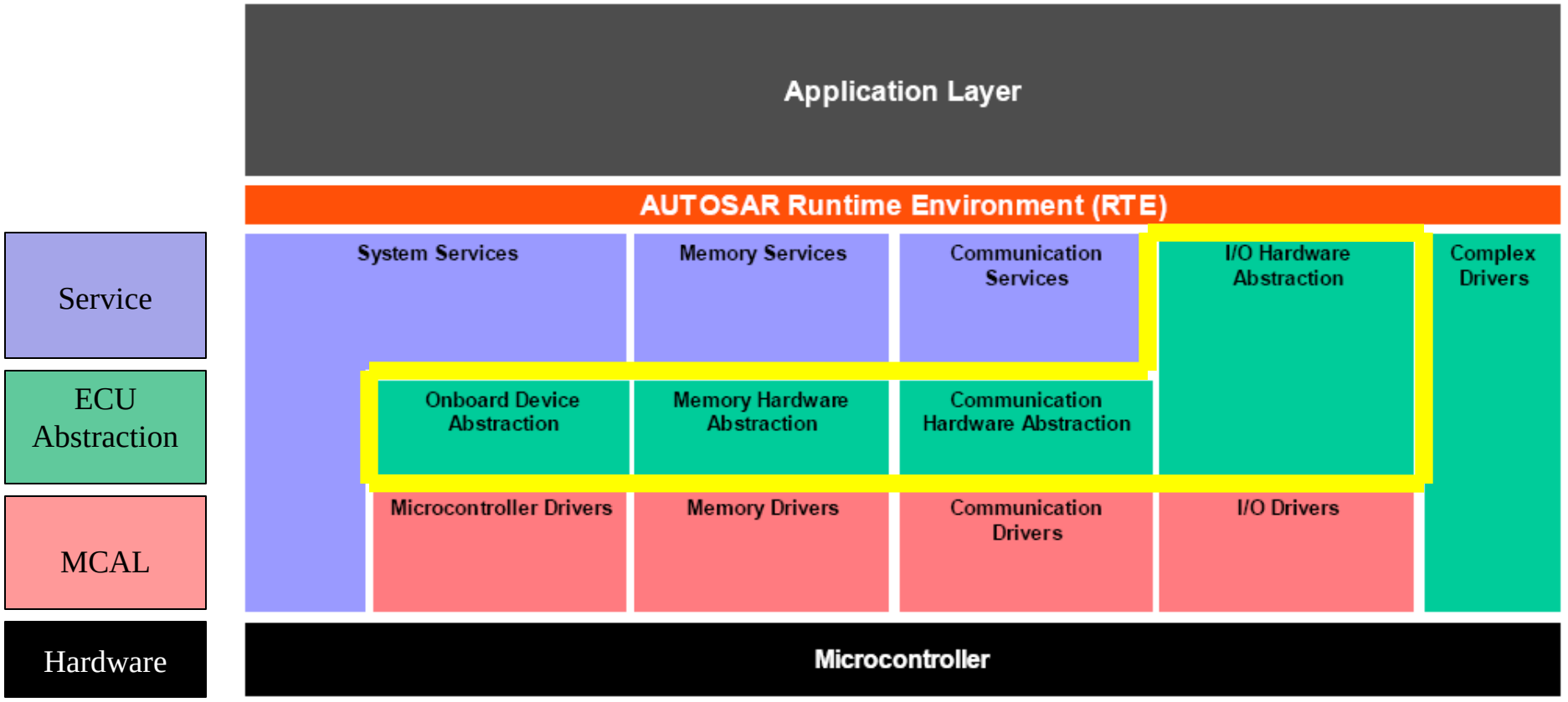
La couche Microcontroler Abstraction Layer (MCAL)



La couche Microcontroler Abstraction Layer (MCAL)

- Implémente les drivers des périphériques internes au microcontrôleur:
  - I/O drivers = DIO, ADC, PWM, etc…
  - Communication drivers = CAN, LIN, Flexray et SPI
  - Memory drivers = Eeprom interne, Flash interne, etc…
  - Microcontroler drivers = Timer, Watchdog interne, etc…
- Accède directement aux registres du microcontrôleur
- Permet de rendre les couches supérieures indépendantes du microcontrôleur

La couche ECU abstraction

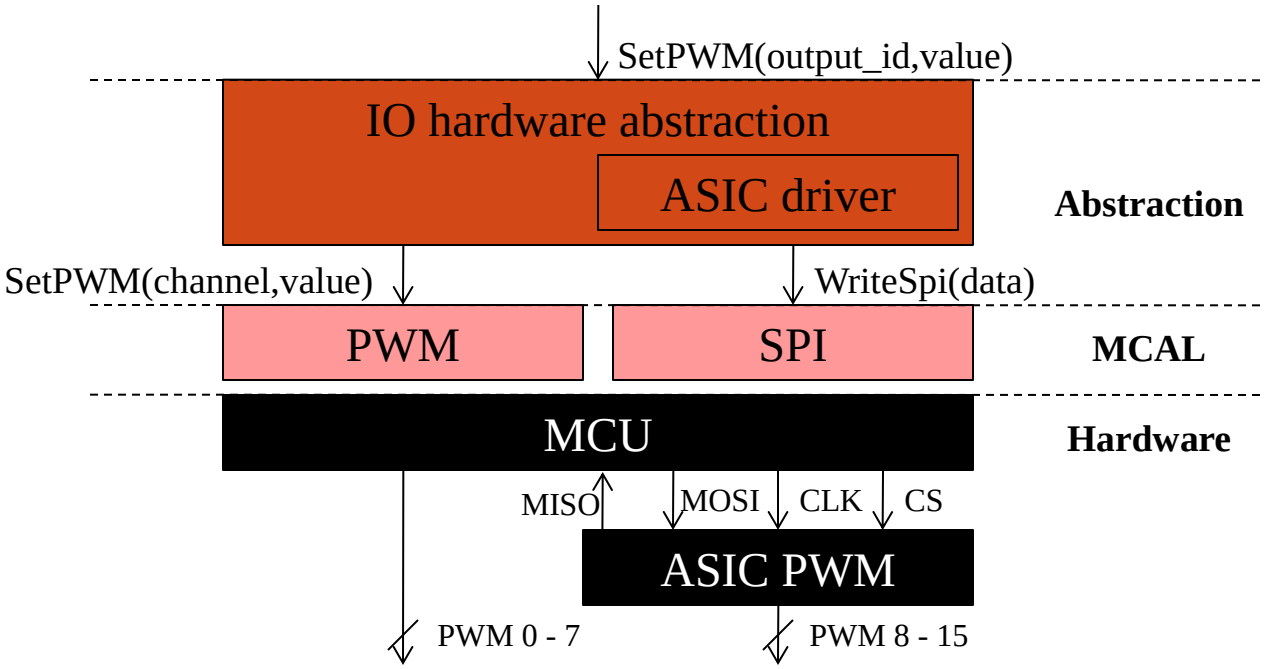


La couche ECU abstraction

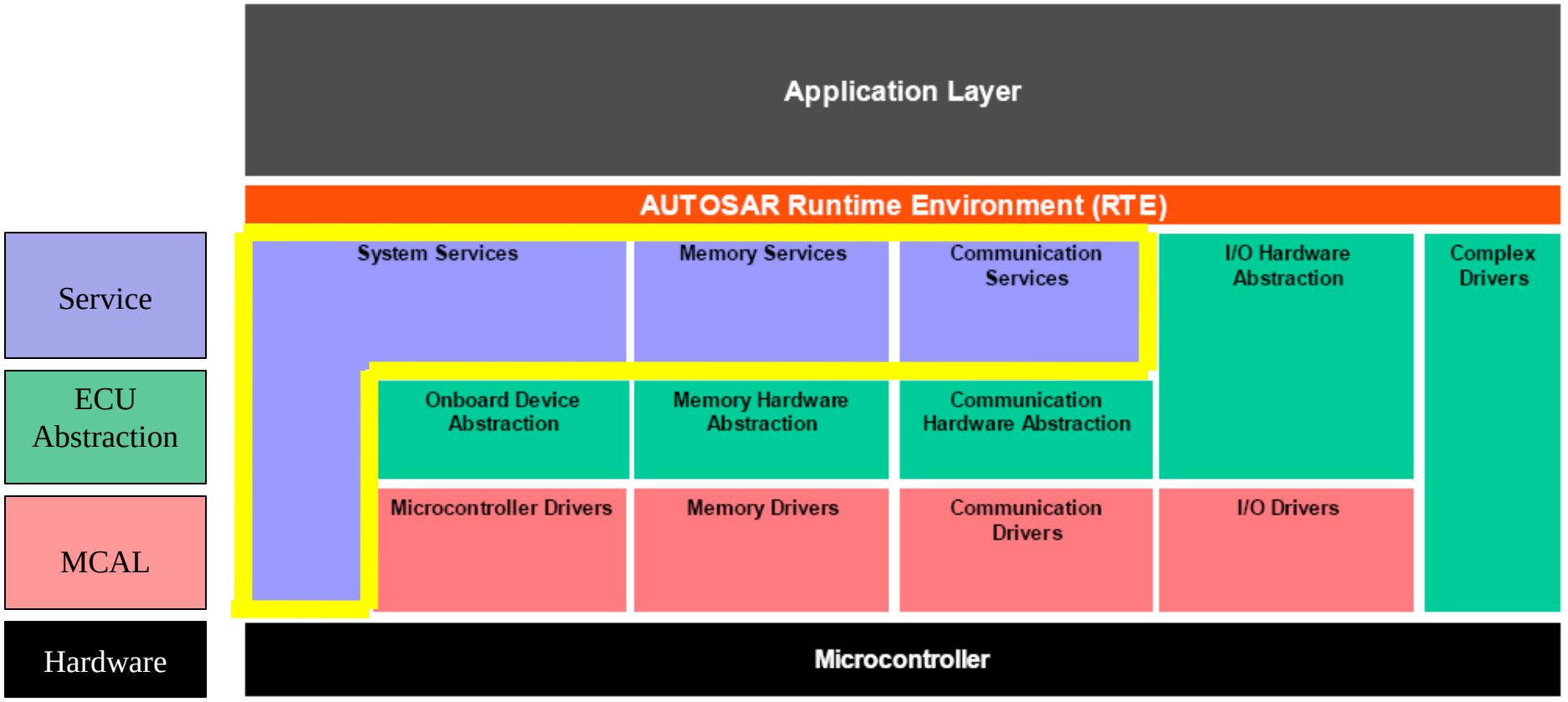
- Implémente les drivers des périphériques externes au microcontrôleur
- Utilise les services du MCAL pour accéder aux E/S du microcontrôleur
- Permet d'accéder aux E/S de l'ECU et autres périphériques indépendamment de leur routage et de leur localisation (interne, externe, …)
- Permet de rendre les couches supérieures indépendantes de l'ECU

Exemple IO hardware abstraction

- ECU avec 16 sorties PWM:
  - 8 channels directement connectés au microcontrôleur
  - 8 channels gérés par un ASIC externe piloté par SPI



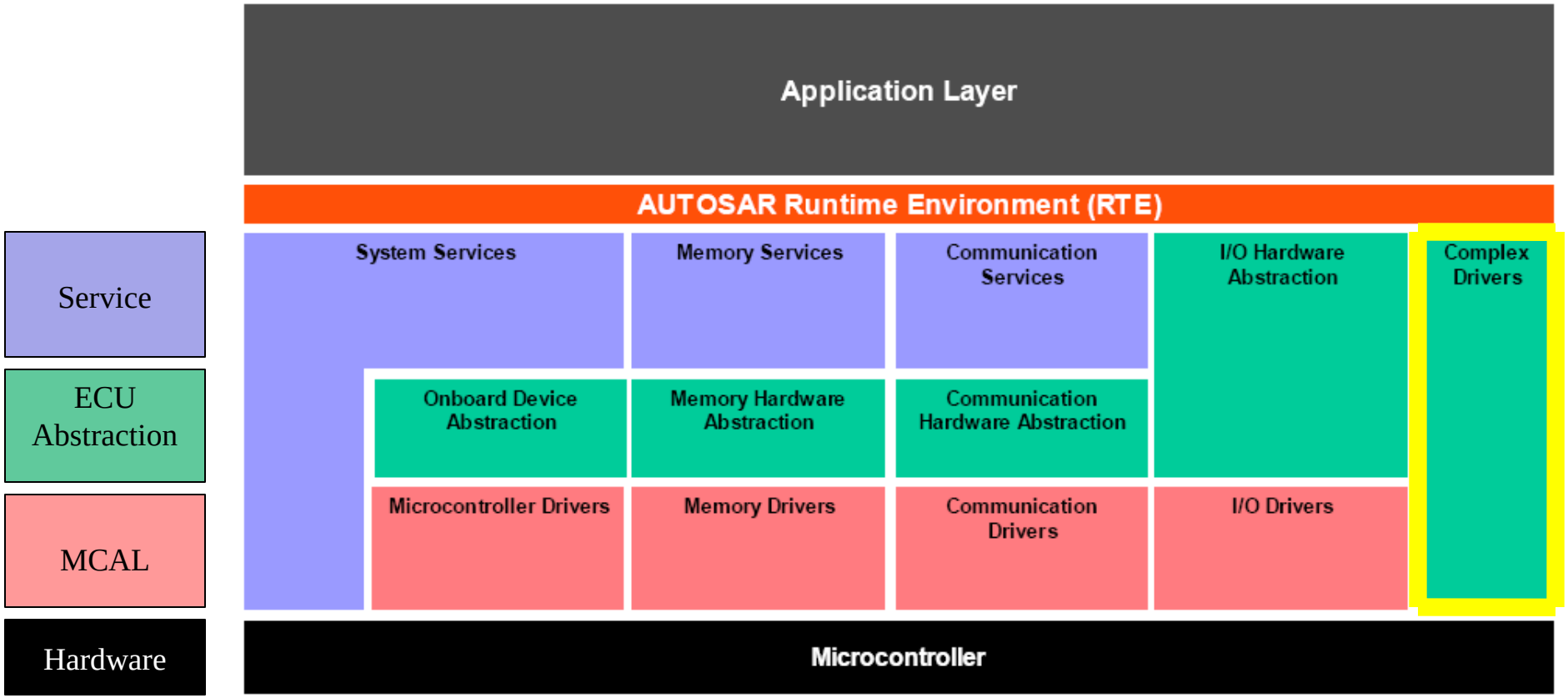
La couche de service



La couche de service

- Logiciel standard qui est indépendant du microcontrôleur et de l'ECU
- System service = OS, Watchdog Manager (WDM), ECU state manager, COM manager, BSW Scheduler…
- Communication service = Network Management (NM), Diagnostic Communication Manager (DCM), COM, …
- Memory service = Non Volatil Memory (NVM = gestion CRC, redondance EEPROM), …

Les Complex Device Driver (CDD)

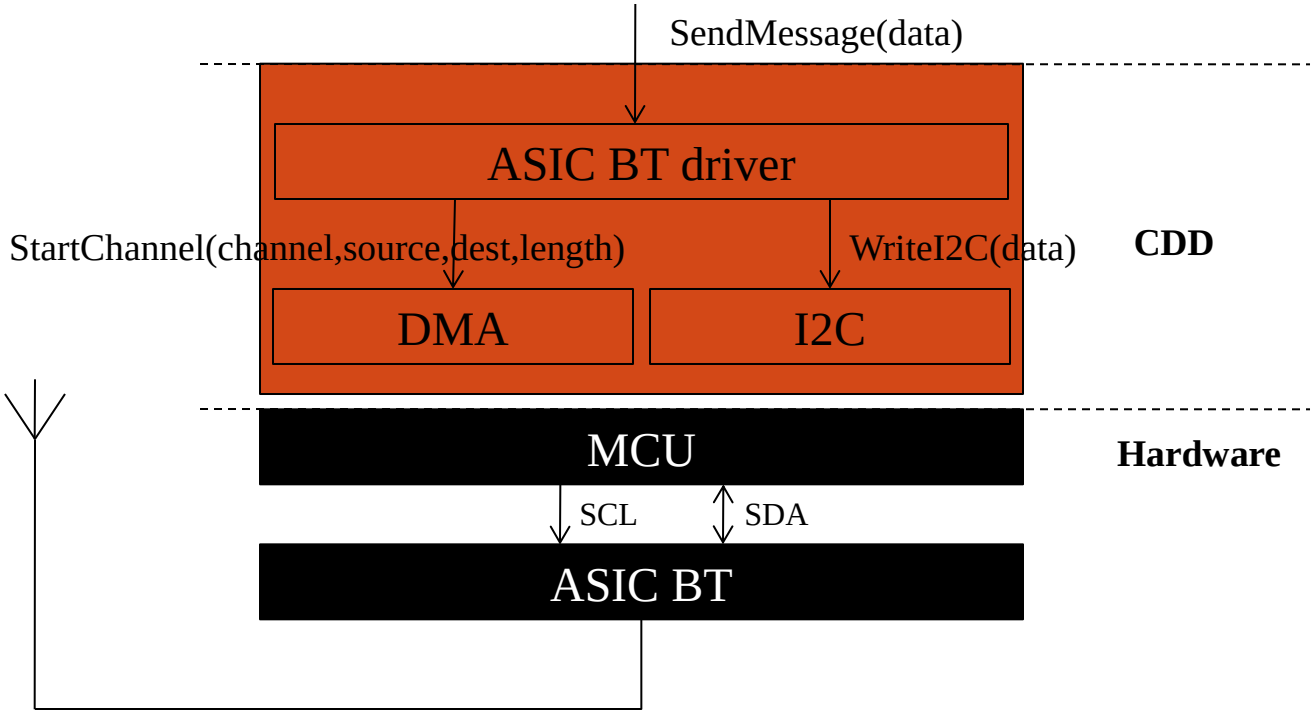


Les Complex Device Driver (CDD)

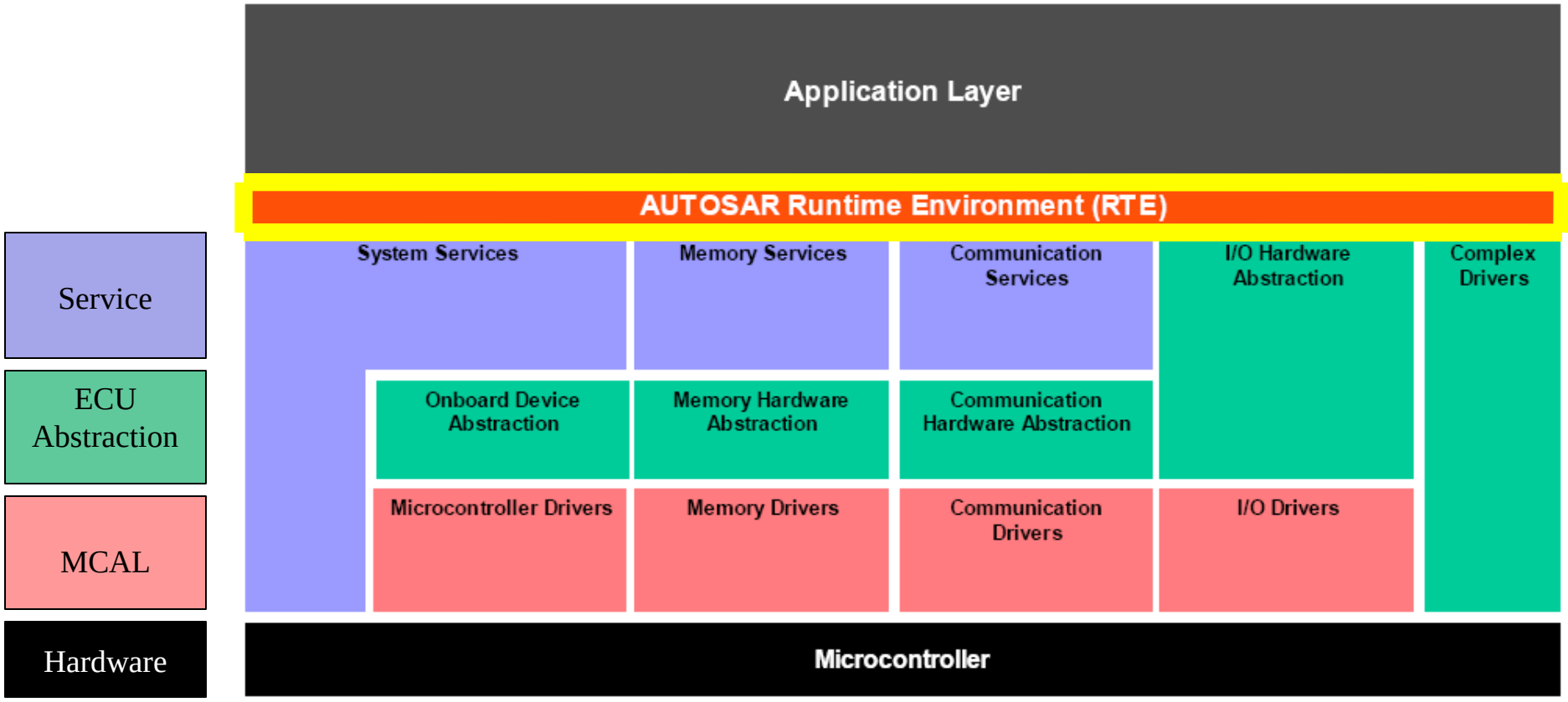
- Implémente les drivers des périphériques internes ou externes « exotiques »
- Accède directement aux registres et interruptions du microcontrôleur
- Logiciel non-standard dépendant du microcontrôleur et/ou de l'ECU
- Permet de s'affranchir des drivers standards AUTOSAR si la performance est critique

Exemple Complex Device Driver

- Module bluetooth I2C + canal DMA:



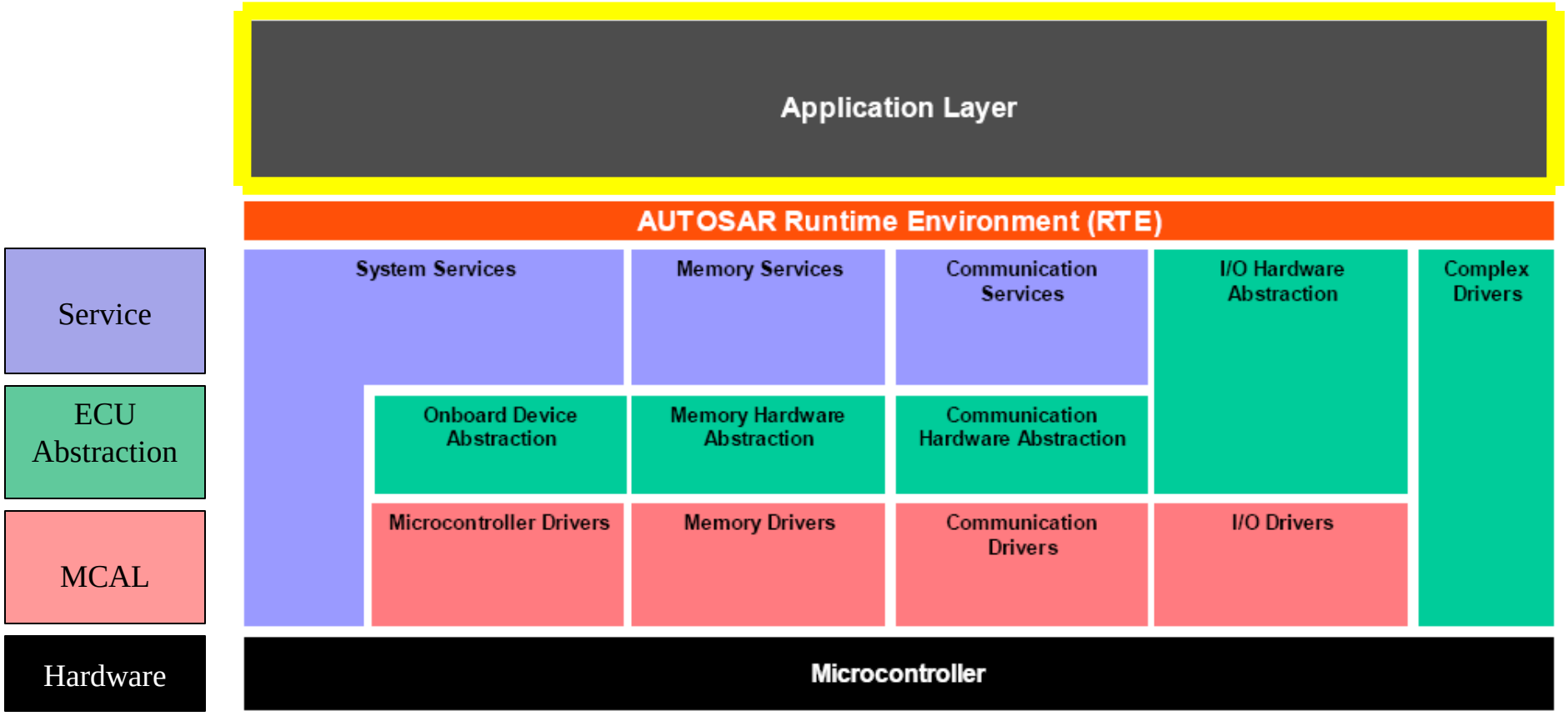
Le Run Time Environement (RTE)



Le Run Time Environement (RTE)

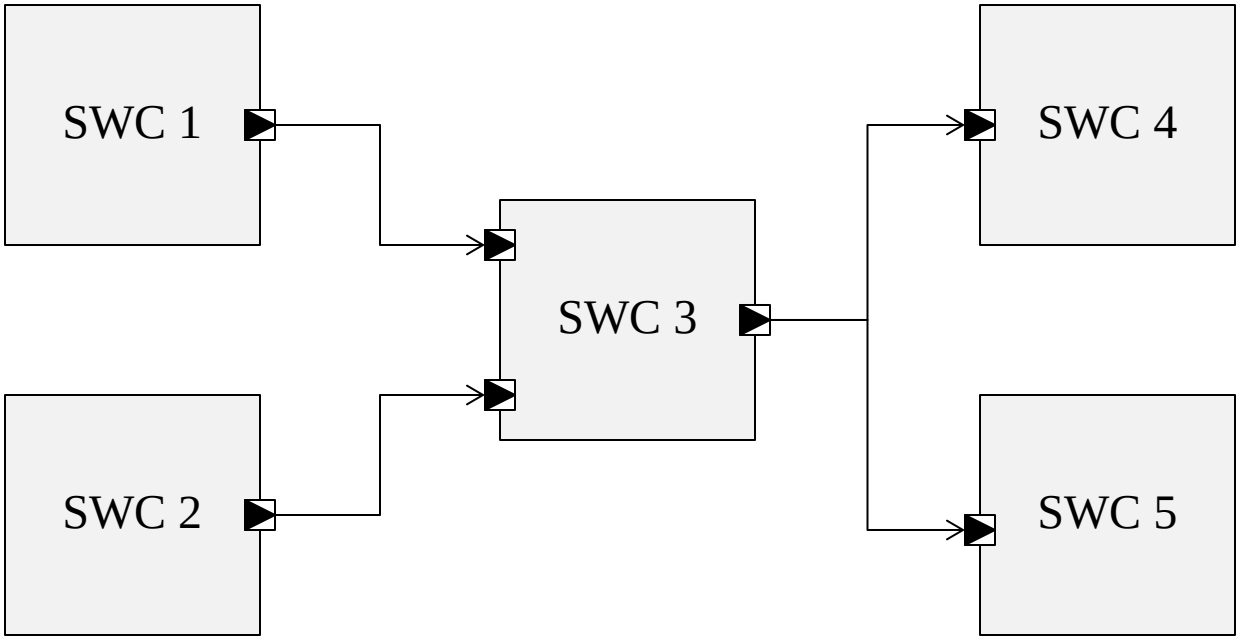
- Permet de rendre les composants applicatifs indépendants de l'ECU dans lequel ils sont déployés
- Le code RTE est généré en fonction des SWC qu'il doit supporter
- Le RTE est en charge de:
  - L'exécution des SWC
  - La communication entre les SWC
  - L'accès au BSW par les SWC

L’application



L’application

- Est composée de composants logiciels (SWC) interconnectés

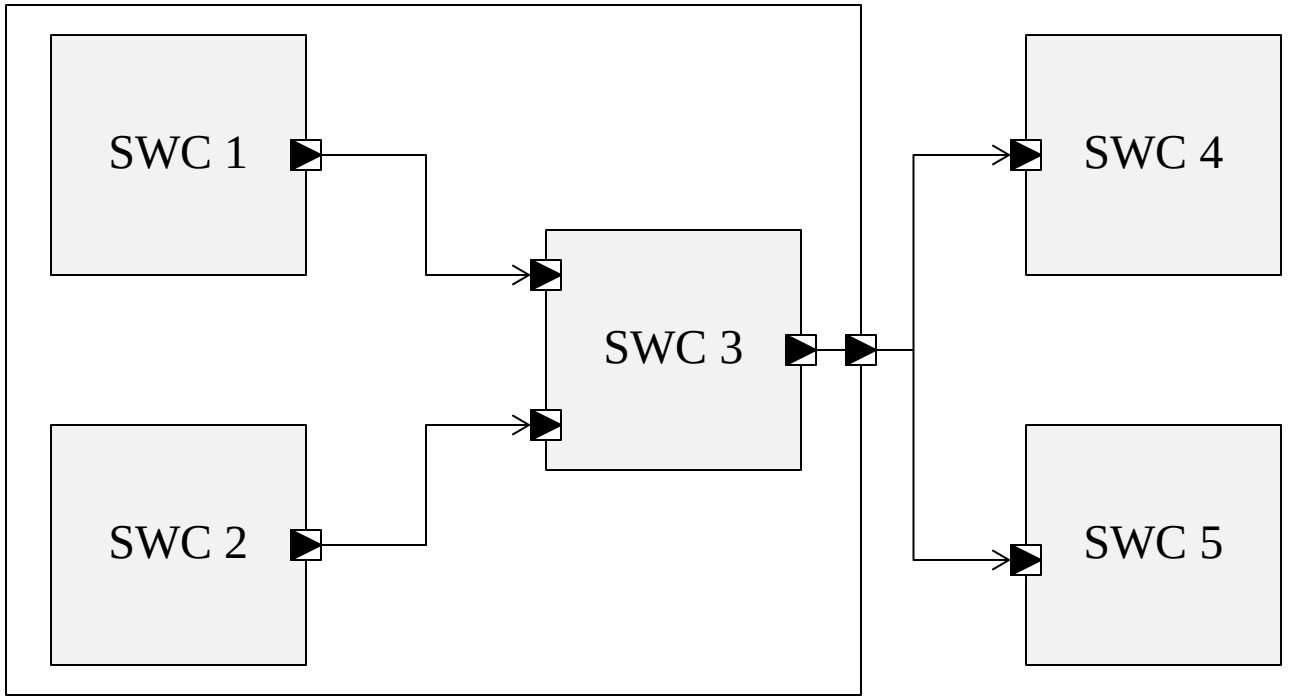


Les SWC

- Chaque SWC implémente une fonction spécifique
- L'implémentation s'appuie sur l'API générée par le RTE
- Ainsi l'implémentation d'un SWC est indépendante:
  - Du microcontrôleur et de l'ECU dans laquelle il s'exécute
  - De la localisation des autres SWC avec lesquels il est connecté (même ECU, autres ECU reliée par CAN, LIN, etc…)

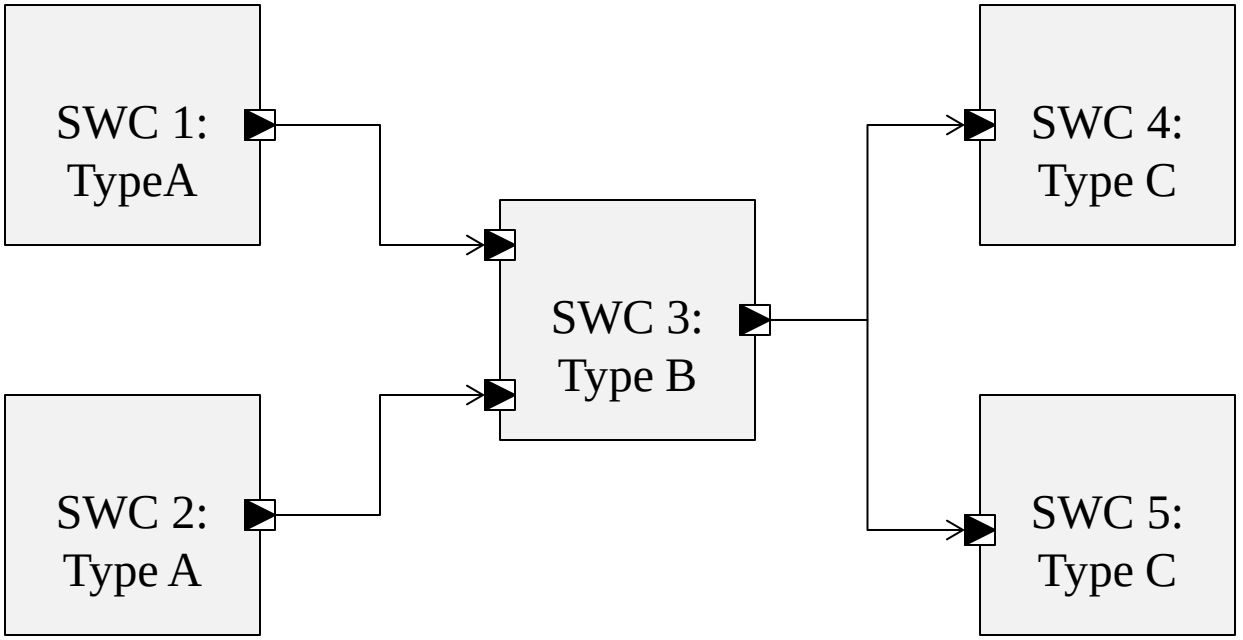
Composition de SWC

- Plusieurs atomic SWC peuvent être regroupés pour former une composition



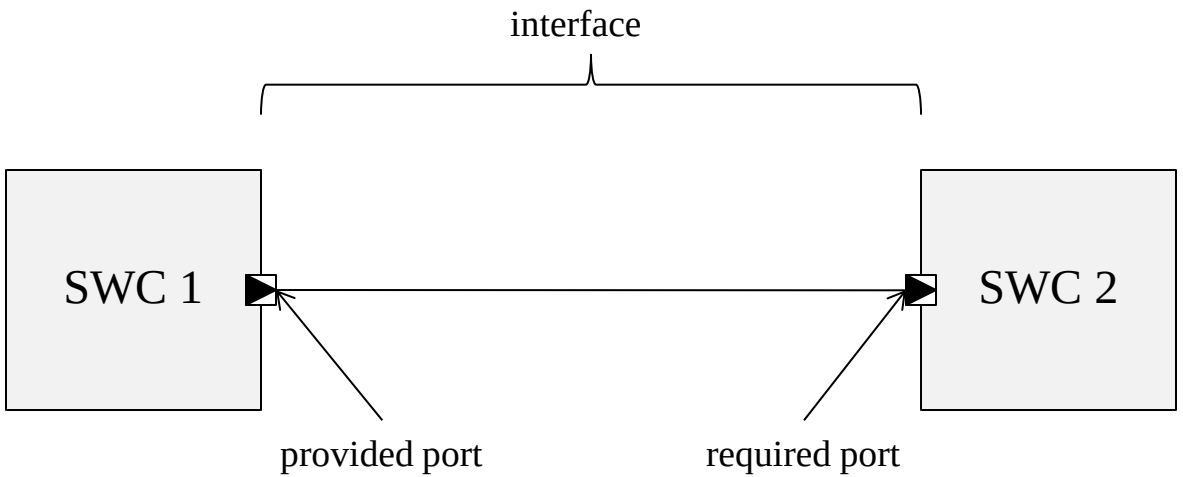
Instanciation multiple d'un SWC

- Plusieurs instances (prototypes) d'un même type de SWC peuvent être crées



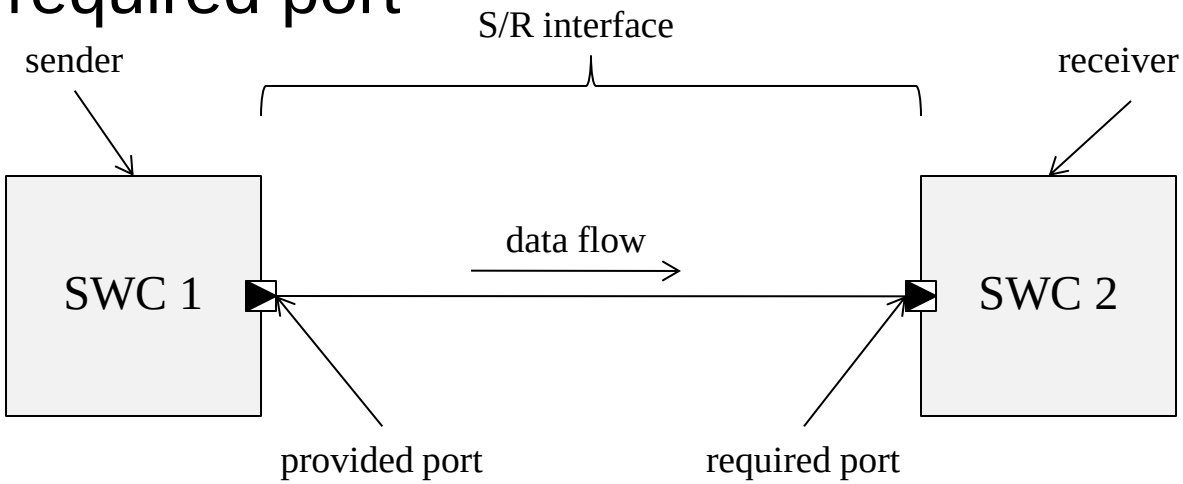
Communication entre SWC

- Les SWC communiquent au travers de ports formant des interfaces



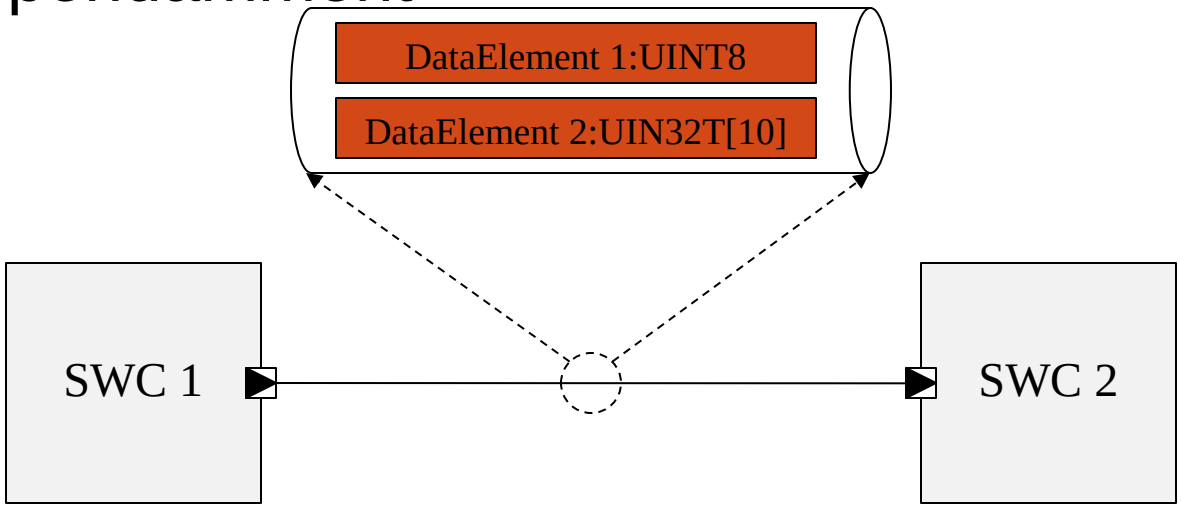
Sender/Receiver Interface

- Permet l'échange de données entre SWC
- Le sender émet les données au travers de son provided port
- Le receiver reçoit les données au travers de son required port



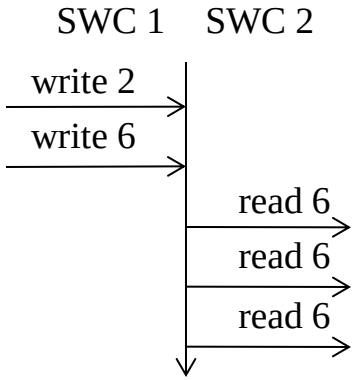
Sender/Receiver Interface

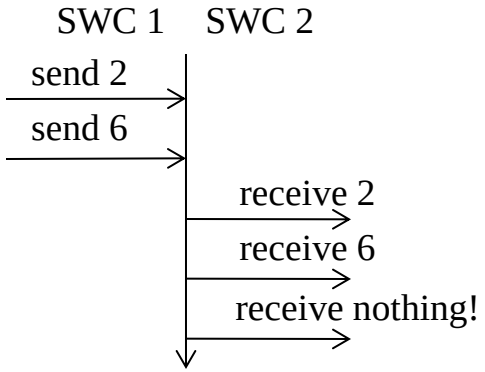
- Une S/R interface peut transporter 1 à N DataElement
- Chaque DataElement est géré indépendamment



Sender/Receiver Interface

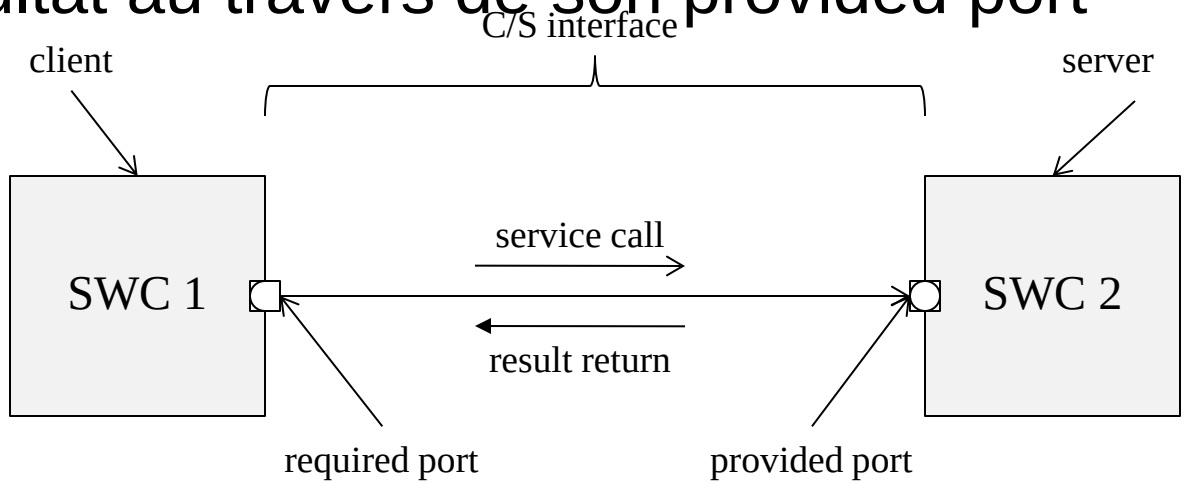
- Data sematic
  - Last is best
  - IsQueued == FALSE
  - Rte_Read_<port>_<data>
  - Rte_Write_<port>_<data>
  - Ex: Speed
- Event semantic
  - FIFO (on receiver side only)
  - IsQueued == TRUE
  - Rte_Send _<port>_<data>
  - Rte_Receive _<port>_<data>
  - Ex: JDD





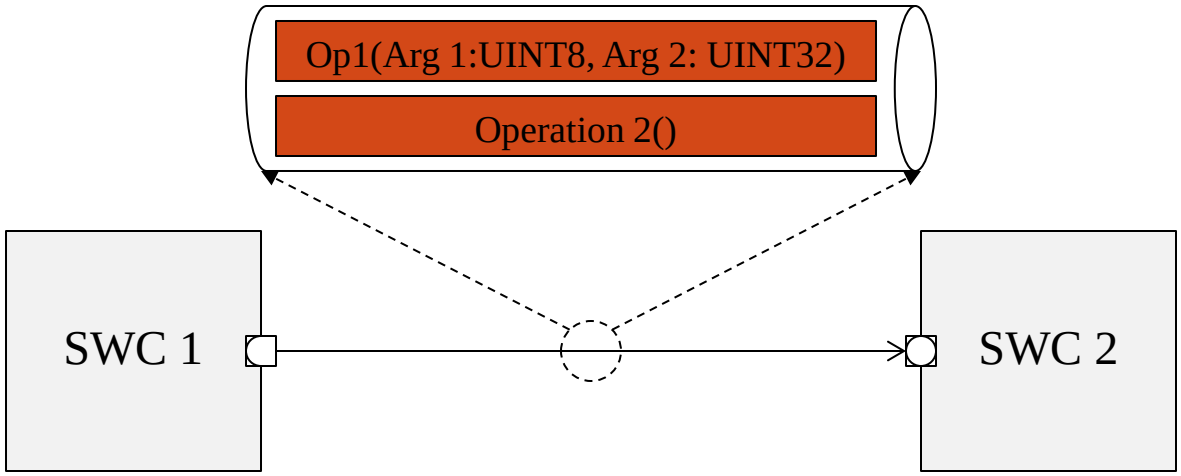
Client/Server Interface

- Permet l'appel de services entre les SWC
- Le client appelle le service au travers de son required port
- Le server exécute le service puis retourne le résultat au travers de son provided port



Client/Server Interface

- Une C/S interface peut contenir 1 à N operation
- Chaque operation contient 0 à N arguments (IN, OUT, INOUT)
- Chaque operation est gérée indépendamment

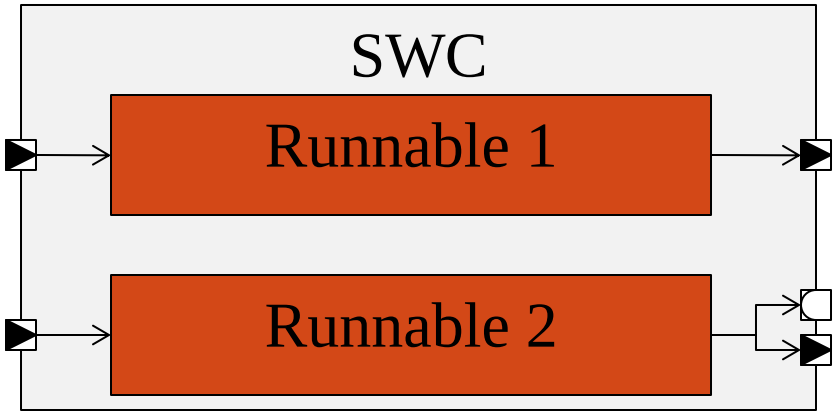


Client/Server Interface

- L’appel à un service peut être:
- Synchrone:
  - exécution immédiate, valeur de retour disponible immédiatement
  - Rte_Call_<port>_<op>()
- Asynchrone:
  - exécution différée, valeur de retour disponible ultérieurement
  - Rte_Call_<port>_<op>()
  - Rte_Result_<port>_<op>()

Runnable

- Les SWC sont composés de runnable
- Ce sont les points d'entrée pour l'exécution des SWC
- Le contexte d'exécution de chaque runnable est indépendant



RTE Event

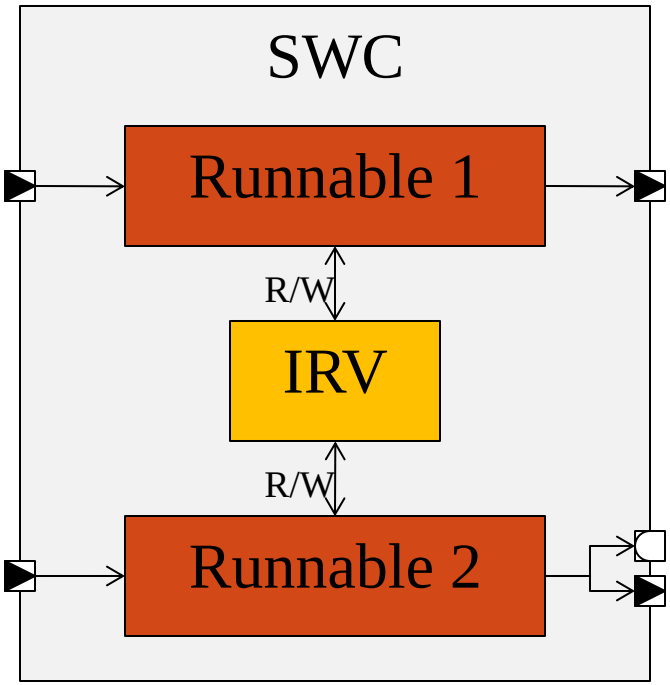
- Les runnables peuvent être exécutés suivant plusieurs types d'événements RTE:
  - TimingEvent
  - DataReceivedEvent
  - DataReceiveErrorEvent
  - DataSendCompleteEvent
  - OperationInvokedEvent
  - AsynchronousServerCallReturnsEvent
  - ModeSwitchEvent

Access point

- Les runnables accèdent aux ports du SWC
  - Data access read
  - Data accedd write
  - Data access receive
  - Data access send
  - Asynchronous server call point
  - Synchronous server call point

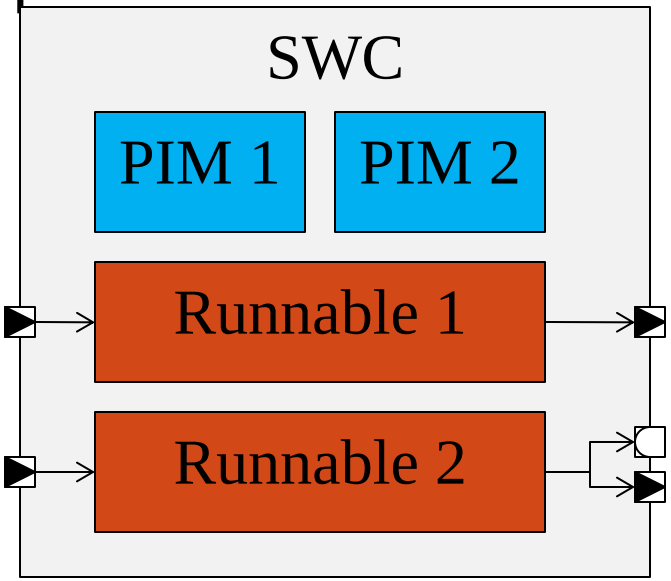
Inter Runnable Variable (IRV)

- Les runnable d'un même SWC communiquent au travers d'IRV
  - Rte_IrvWrite_<runbl>_<irv>()
  - Rte_IrvRead _<runbl>_<irv>()



Per Instance Memory (PIM)

- Si instanciation multiple d'un SWC il est nécessaire de déclarer des PIM pour stocker indépendamment les variables d'état de chaque instance du composant
  - Rte_Pim_<pim_name>()



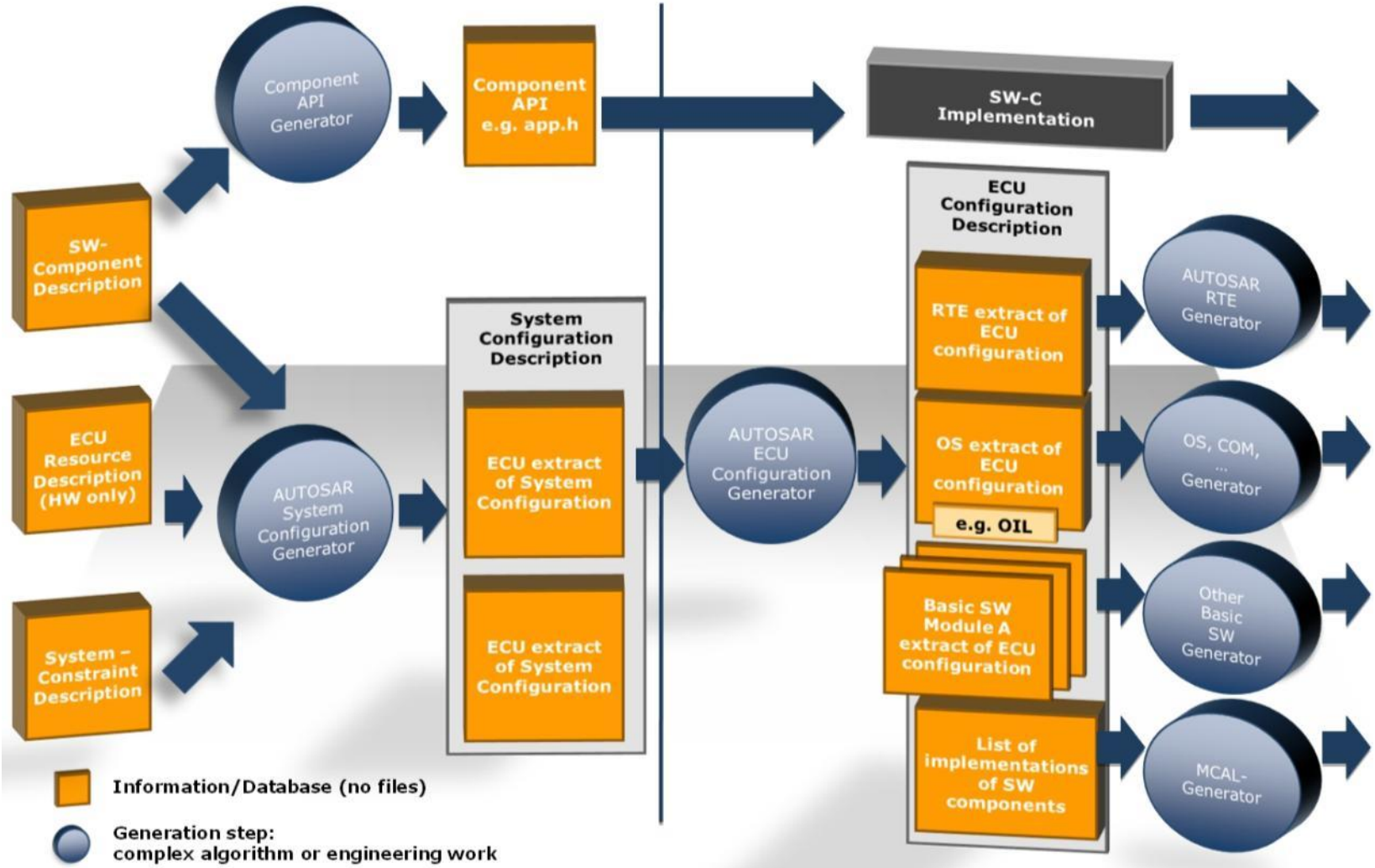
Exclusive Area

- La gestion des accès concurrentiels est encapsulée dans l'API RTE uniquement pour les accès aux:
  - Ports
  - IRV
- Si nécessaire, les SWC peuvent déclarer des exclusive area et y entrer/sortir explicitement:
  - Rte_Enter_<exclusive_area_name>()
  - Rte_Exit_<exclusive_area_name>()

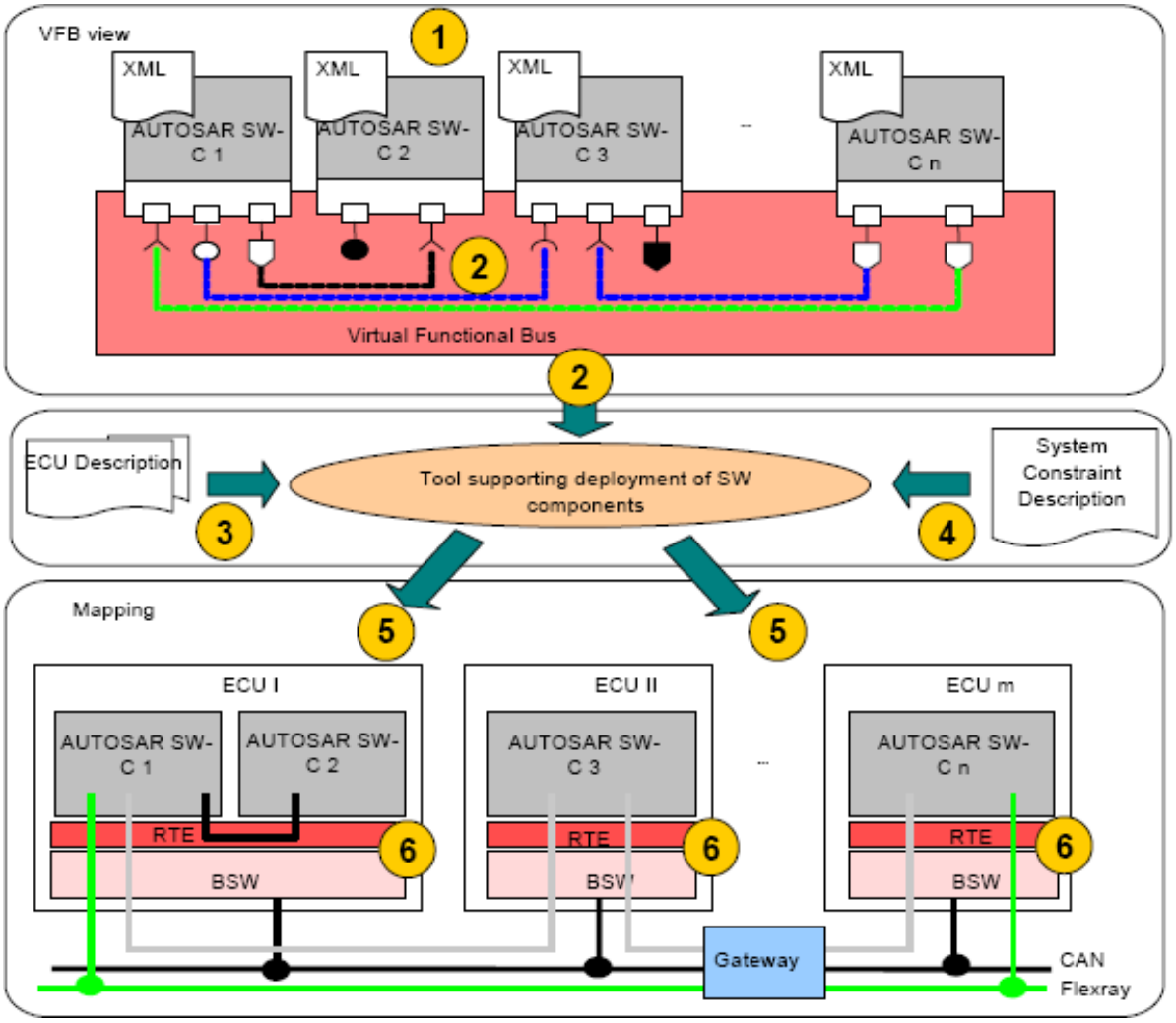
Agenda

- Introduction et historique du consortium
- L'architecture logicielle AUTOSAR
- La méthodologie AUTOSAR
- Quelques « solutions » AUTOSAR
- Conclusion

La méthodologie AUTOSAR

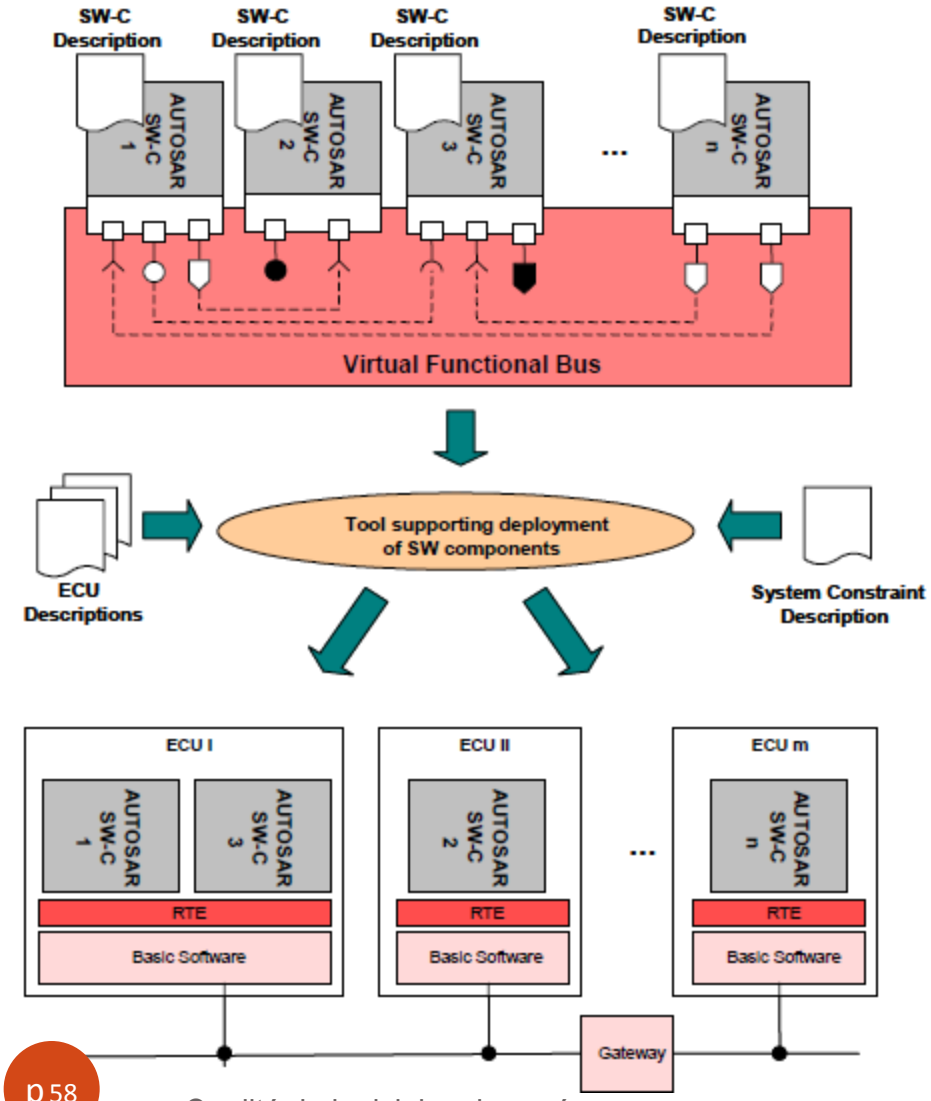


La Methodologie AUTOSAR



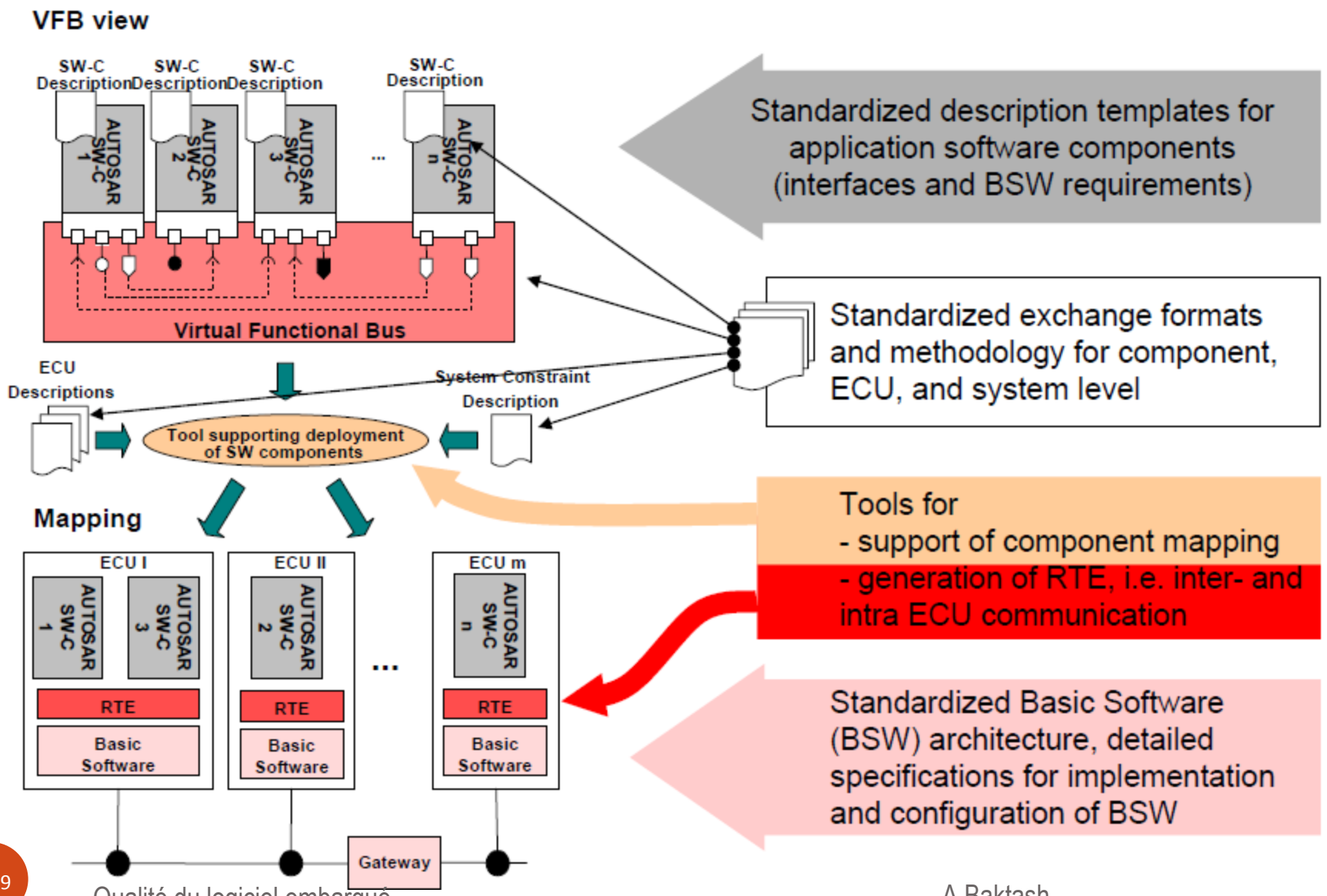
1. Software Component (SW-C) description
2. Integration of the SW-C via the Virtual Functional Bus (VFB)
3. ECU Description
4. System Constraint
5. Mapping of SW-C on specific ECUs
6. Configuration of Basic Software Modules (BSW) and Runtime Environment (RTE)

Following the AUTOSAR Methodology, the E/E architecture is derived from the formal description of software and hardware components.

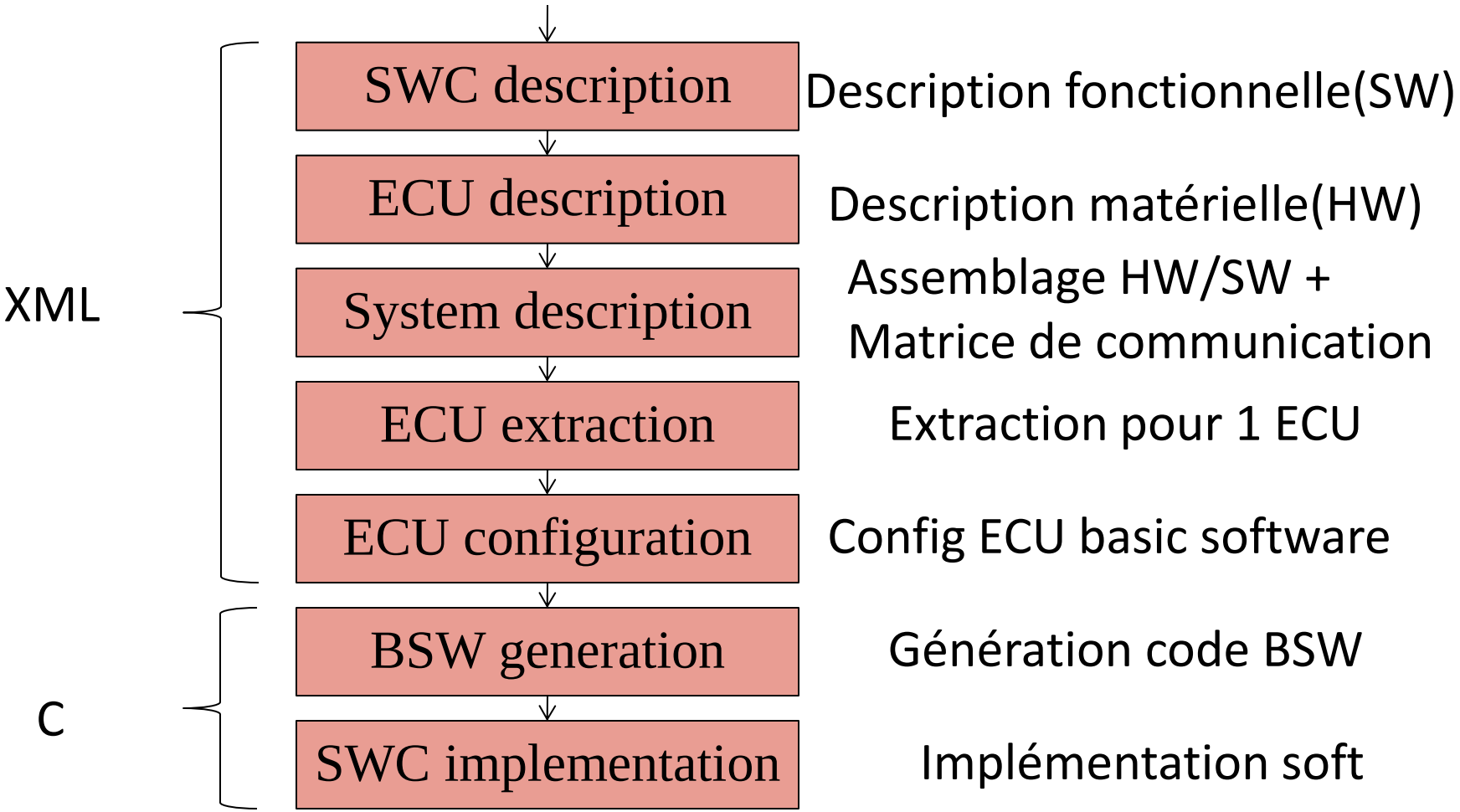


- Functional software is described formally in terms of “Software Components” (SW-C).
- Using “Software Component Descriptions“ as input, the „Virtual Functional Bus“ validates the interaction of all components and interfaces before software implementation.
- Mapping of “Software Components” to ECUs and configuration of basic software.
- The AUTOSAR Methodology supports the generation of an E/E architecture.

AUTOSAR Methodology
Derive E/E architecture from formal descriptions of soft- and hardware components

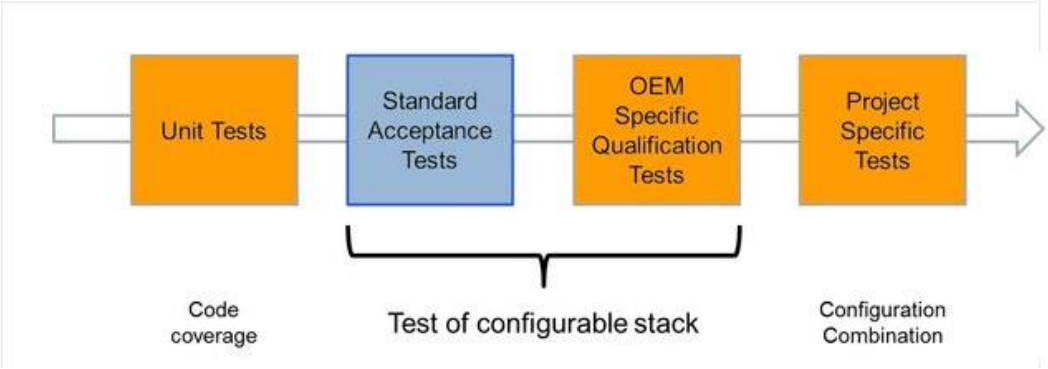


La Methodologie AUTOSAR



Spécification de tests d'acceptances standardisés

- **Objective**
  - Minimiser l'effort (et le coût) des tests



- **Plus-value**
  - Tests de développement et maintenance commune
  - Extensibilité des tests en dehors des modules AUTOSAR
  - Échange de résultats de tests fiables
    - Entre différents stack d'implémentation du fournisseur
    - Entre fournisseur et constructeur
    - Entre constructeurs

Agenda

- Introduction et historique du consortium
- L'architecture logicielle AUTOSAR
- La méthodologie AUTOSAR
- Quelques « solutions » AUTOSAR
- Conclusion

Vector

AUTOSAR Tools

Design of software components	DaVinci Developer
Configuration and generation of AUTOSAR basic software and RTE	DaVinci Configurator Pro
Testing AUTOSAR software components on your PC	DaVinci Component Tester
Multibus-tool for testing, simulation, diagnostics and analysis of networks and distributed systems	CANoe

AUTOSAR Basic Software

ECU basic software (RTE and BSW)	MICROSAR
Tools and basic software to evaluate Vector's AUTOSAR solution	AUTOSAR Evaluation Bundle

dSpace

Embedded Success dSPACE

Quand vous utilisez des outils dSPACE pour développer un logiciel de calculateur compatible AUTOSAR, vous avez à disposition des solutions de développent automobile globales. Avec SystemDesk® et TargetLink®, vous pouvez aussi bien modéliser des calculateurs simples que des réseaux entiers comportant des calculateurs et des bus du fichier travail. Avec le nouveau RTI AUTOSAR Package, vous pouvez utiliser les composants logiciels AUTOSAR pour le prototypage rapide de lois de commande en important vos composants logiciels AUTOSAR d'origine basés sur code C, dans l'environnement MATLAB®/Simulink® et dans le système temps réel dSPACE.

- SystemDesk
  Pour la conception, l'implémentation et l'intégration d'architectures complexes de système et de systèmes logiciels distribués
- SystemDesk RTE Generation Module
  Pour la génération d'un code C de production qui implémente un environnement exécutable (RTE) compatible avec AUTOSAR
- SystemDesk V-ECU Generation Module
  The dSPACE SystemDesk V-ECU Generation Module enables the generation of virtual ECUs and offline simulation
- TargetLink AUTOSAR Module
  TargetLink bridges the gap between model-based design and AUTOSAR-compliant software development.
- RTI AUTOSAR Package
  Utilisation des composants logiciels AUTOSAR dans un environnement MATLAB®/Simulink®

Mentor Graphics

VSX TOOLS BY MENTOR GRAPHICS

Volcano VSx is an integrated tool suite for top-down vehicle system and ECU design. VSx covers automotive software and electronic systems design, virtual verification, testing, and configuration. Despite tight integration, each tool can be individually used and supports standard AUTOSAR import/exports.

Volcano Vehicle Systems Architect (VSA)

The **Volcano Vehicle Systems Architect** is a systems design tool for AUTOSAR-based systems. It enables engineers to design, explore, and compare electronic and SW architectures. Volcano Vehicle Systems Architect ›

Volcano VSA COM Designer

The **Volcano VSA COM Designer** is a network design tool that enables users to design automotive CAN, LIN and FlexRay networks in keeping with modern systems engineering practices, while managing timing requirements, variants, configurations, and releases. Volcano VSA COM Designer ›

Volcano Vehicle Systems Integrator (VSI)

The **Volcano Vehicle Systems Integrator** is an AUTOSAR software execution environment designed specifically for the development of vehicular embedded systems. It provides early validation of software functionality on virtual electronic control units (ECUs) on the desktop. Volcano Vehicle Systems Integrator ›

Volcano VSTAR

Volcano VSTAR AUTOSAR Basic Software (BSW) stack provides a fully AUTOSAR 4.0 compliant and scalable middleware for ECU design, which abstracts the application from the HW-dependent layer. VSTAR software optionally includes complete communication stacks for LIN, CAN, and FlexRay implementations. Volcano VSTAR ›

Dassault Systèmes

Concevoir des applications et des systèmes AUTOSAR

Face à l'importance croissante des ECU dans les automobiles modernes et autres produits intelligents, la complexité des produits électroniques / électriques ne cesse d'augmenter, et leur processus de développement se complexifie également. AUTOSAR Builder est une suite d'outils ouverte, basée sur Eclipse, pour la conception et le développement de systèmes et de logiciels conformes au standard AUTOSAR.

AUTOSAR Builder est le socle d'environnements spécifiques et dédiés, prenant en charge différents rôles et contraintes d'intégration de processus. Cette suite d'outils comprend plusieurs plug-ins interopérables, correspondant aux différentes phases du processus de développement AUTOSAR, qui produisent des descriptions et informations standards basées sur la méthodologie et le standard AUTOSAR.

Points forts de la solution :

- AUTOSAR Builder est basé sur le système ouvert Artop (**www.artop.org**)
- Permet l'importation de descriptions existantes de conceptions basées sur des modèles et la génération de code conforme à AUTOSAR
- Traite les différentes phases du processus de conception AUTOSAR
- Peut être totalement intégré à votre processus de développement

Elektrobit

- EB tresos product line
  - EB tresos Studio
  - EB tresos AutoCore
  - EB tresos WinCore
  - EB tresos OsekCore
  - EB tresos Debug and Trace
  - EB tresos Inspector
  - EB tresos Busmirror
  - Bus interface hardware
- AUTOSAR
- Functional Safety
- ECU Software Engineering
- ECU Product News

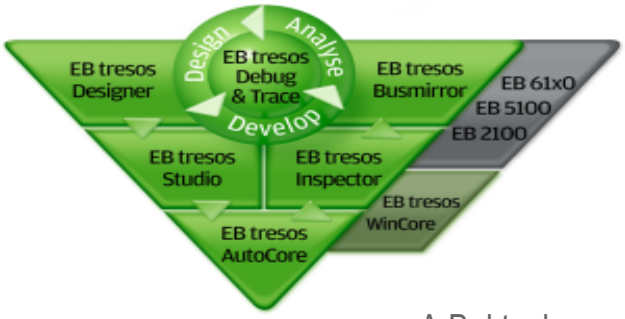
AUTOSAR BASIC SOFTWARE

Configuration, validation and generation

EB tresos offers tools and basic software that help develop state-of-the-art ECUs. EB is uniquely positioned to offer a complete solution encompassing: basic software configuration, debugging, cluster emulation and bus analysis products and services in the portfolio.

EB tresos product line

- EB tresos Studio - embedded software configuration, validation and generation
- EB tresos AutoCore - AUTOSAR compliant basic software core
- EB tresos WinCore - AUTOSAR compliant basic software core that runs on Win32
- EB tresos OsekCore - OSEK/VDX compliant basic software core
- EB tresos Debug & Trace - AUTOSAR debugging at runtime
- EB tresos Inspector with EB 61×0 or EB 2100 - measurement and analysis in FlexRay, CAN and LIN networks
- EB tresos Busmirror with EB 61×0, EB 2100 or EB 5100 - FlexRay and CAN cluster emulation

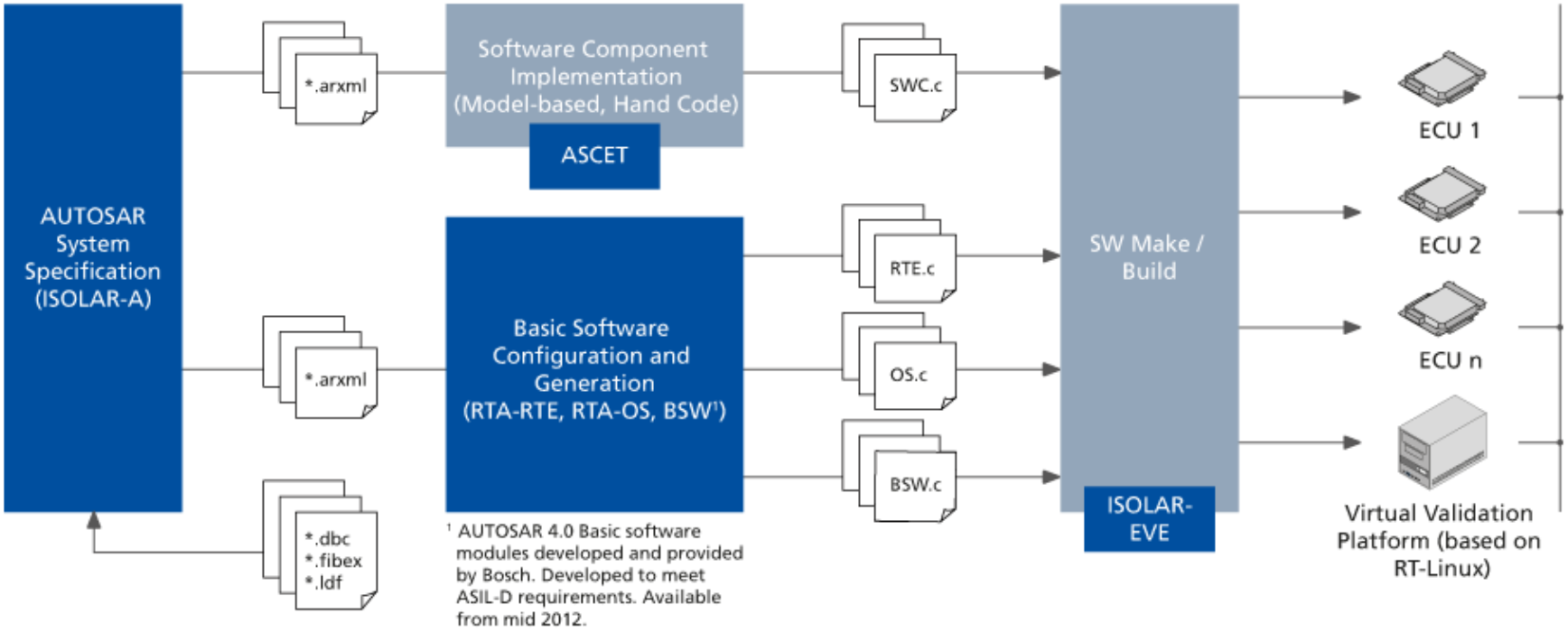


Open AUTOSAR and OSEK compliant basic software

ETAS

DRIVING EMBEDDED EXCELLENCE

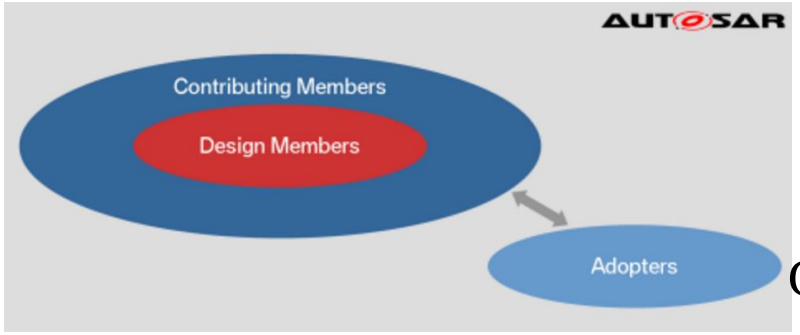
ETAS



The AUTOSAR Tool Platform (Artop):

- implementation of common base functionality for AUTOSAR development tools.
- including its source code,
- Available free of charge to all AUTOSAR members and partners.

- The Artop development process is transparent and based on a community approach driven by AUTOSAR members and partners. The community that develops Artop is organized as the Artop User Group.



Design Members :

BMW
Continental
Dassault Systems
ETAS4Itemis
PSA

Contributing Members:

ArcCore
Berner & Mattner
Mentor Graphics
Tata Elxsi
Vector

Agenda

- Introduction et historique du consortium
- L'architecture logicielle AUTOSAR
- La méthodologie AUTOSAR
- Quelques « solutions » AUTOSAR
- Conclusion

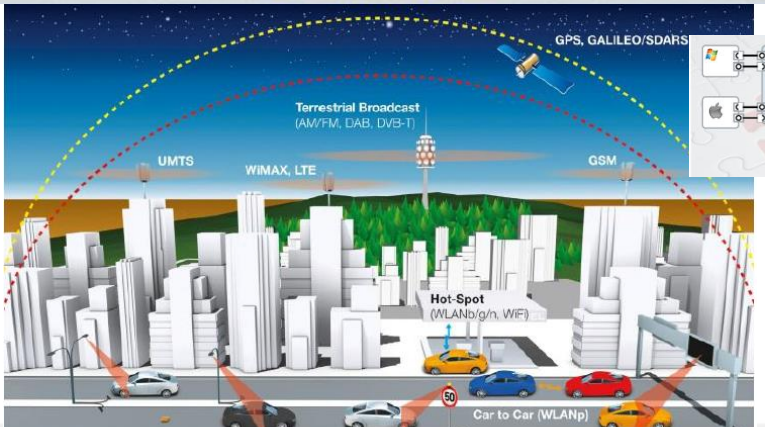
Conclusion

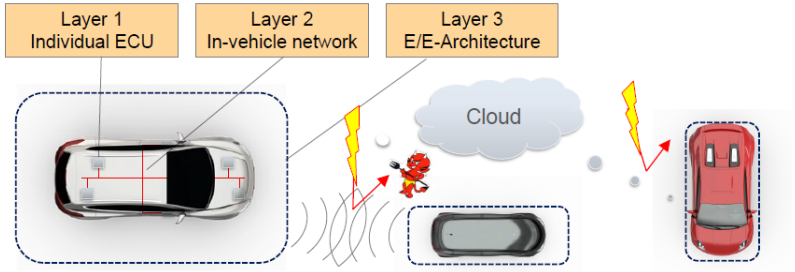
- Key success factors
  - Short development cycles
  - Front loading of validation
  - Precision and quality of the standard
  - Early availability of implementation
  - Interoperability and increased quality
- Key enabler on the way to the self-driving car.
- standard of choice for new technologies:
  - Autonomous driving
  - Interconnectivity
- Worldwide standard

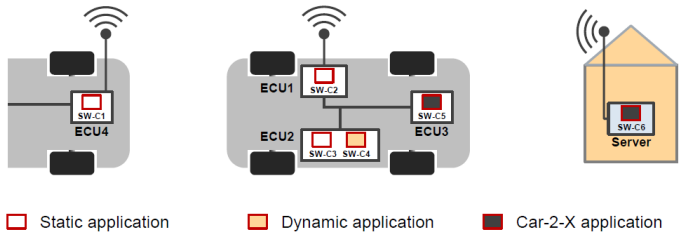
Et la suite de l'AUTOSAR?

Highly automated driving

Car-2-X applications







Vehicle in the cloud

Increased connectivity

A.Baktash

Technology Drivers

Ethernet

- High bandwidth
- Communication system is not limiting aspect any more
- Switched network
- Efficient point-to-point communication
- Efficient transfer of long messages

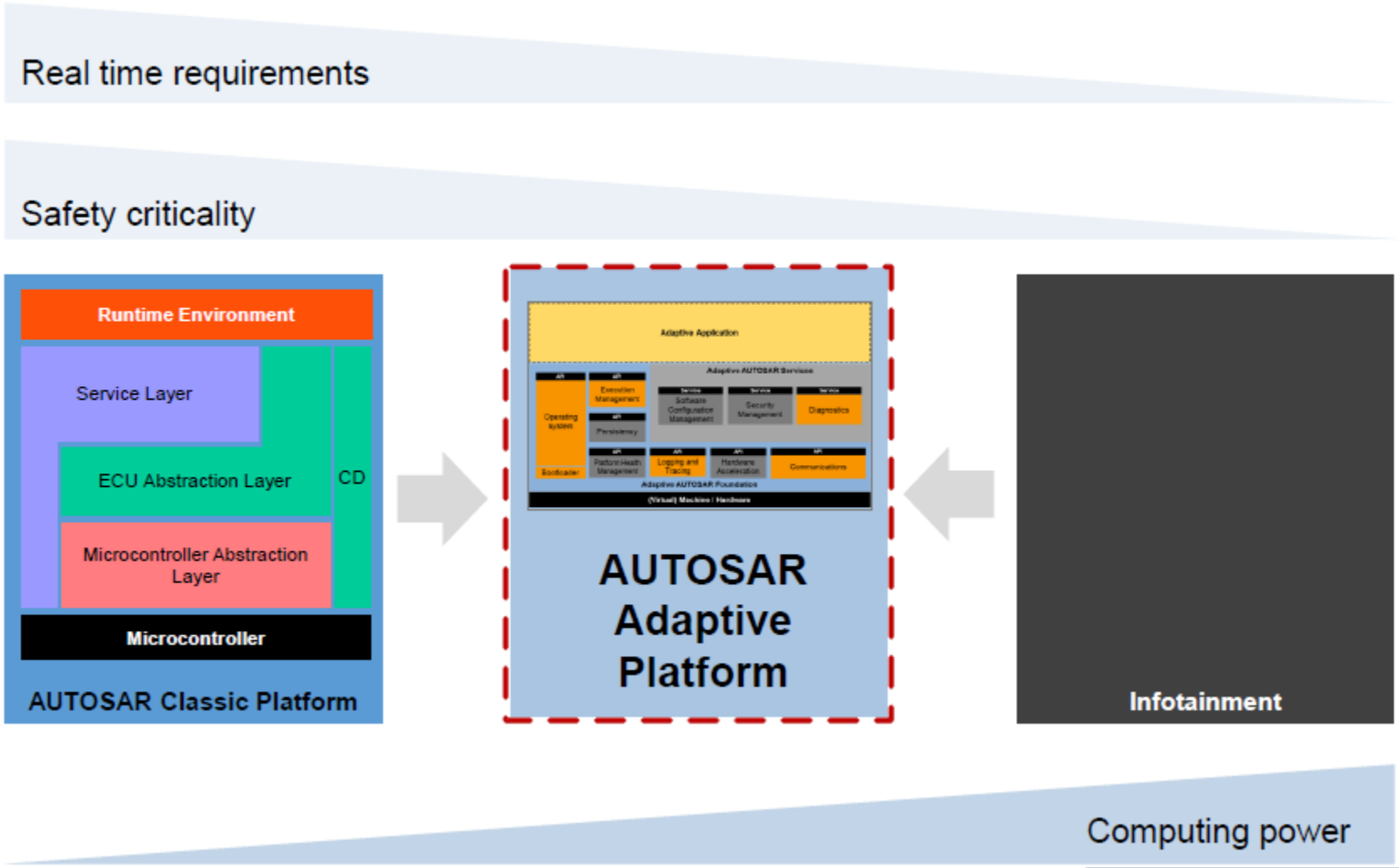
Processors

- Switch from microcontroller to processors with external memory (and maybe filesystems)
- Many core processors
- Parallel computing
- „Cheap“ availability of computing power

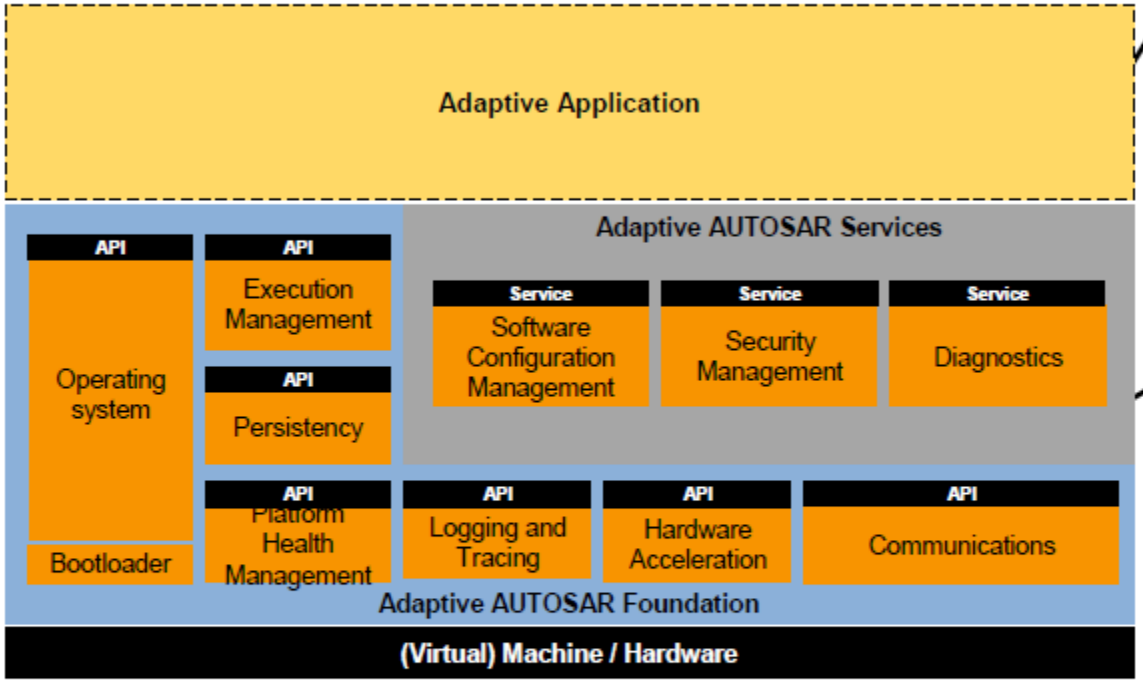
Heterogeneous architectures

- Special purpose processors

AUTOSAR adaptive platform



Architecture Adaptive Platform



Adaptive AUTOSAR API:
APIs and services exposed to Applications by functional clusters.

Adaptive AUTOSAR specification:
Behavior of software platform from Application and Network perspective.

Organized in functional clusters, not specification of internal architecture!

Functional Clusters:

- Assemble functionalities of the Adaptive Platform
- Define clustering of requirements specification
- But, do not constrain the SW architecture of a platform implementation

➔ **No definition of modules**

A.Baktash

http://Autosar.org

AUTOSAR AUTomotive Open System ARchitecture

Please note that the prolongation for the AUTOSAR agreements has started. To continue in AUTOSAR please contact admin@

Home | About | Partners | Specifications | User Groups | Events & Publications | Media | Contact

You are here: Home

Home

Welcome to the AUTOSAR development partnership

AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide development partnership of car manufacturers, suppliers and other companies from the electronics, semiconductor and software industry.

AUTOSAR

- Paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness
- Is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation"
- Is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality
- Facilitates the exchange and update of software and hardware over the service life of the vehicle

Media

Munich, November, 2012

The AUTOSAR (AUTomotive Open System ARchitecture) development partnership has defined its goals for post phase III organization beginning in 2013.

read more

Publications

AUTOSAR - The worldwide Automotive Standard for E/E systems

read more

Upcoming Events

AUTOSAR World Tour 2013

read more

Previous Events

Beijing, November, 2012

The 5th AUTOSAR Open Conference took place on Monday November 26, 2012 at CNCC, Beijing, China

The presentations are available here

Sitemap | Privacy Policy | Imprint

Merci de votre attention…

Avez-vous des questions?

Contact:
ali.baktash@smile.fr
ali.baktash@autoliv.com

Présentation inspirée de documents rédigés par:
- Christophe Brunschweiler, Smile
-Simon Fürst, BMW
-Rémy Dziemiaszko, Smile