



# Implementation Challenges in AUTOSAR Framework

By: Gireesh P Chandran

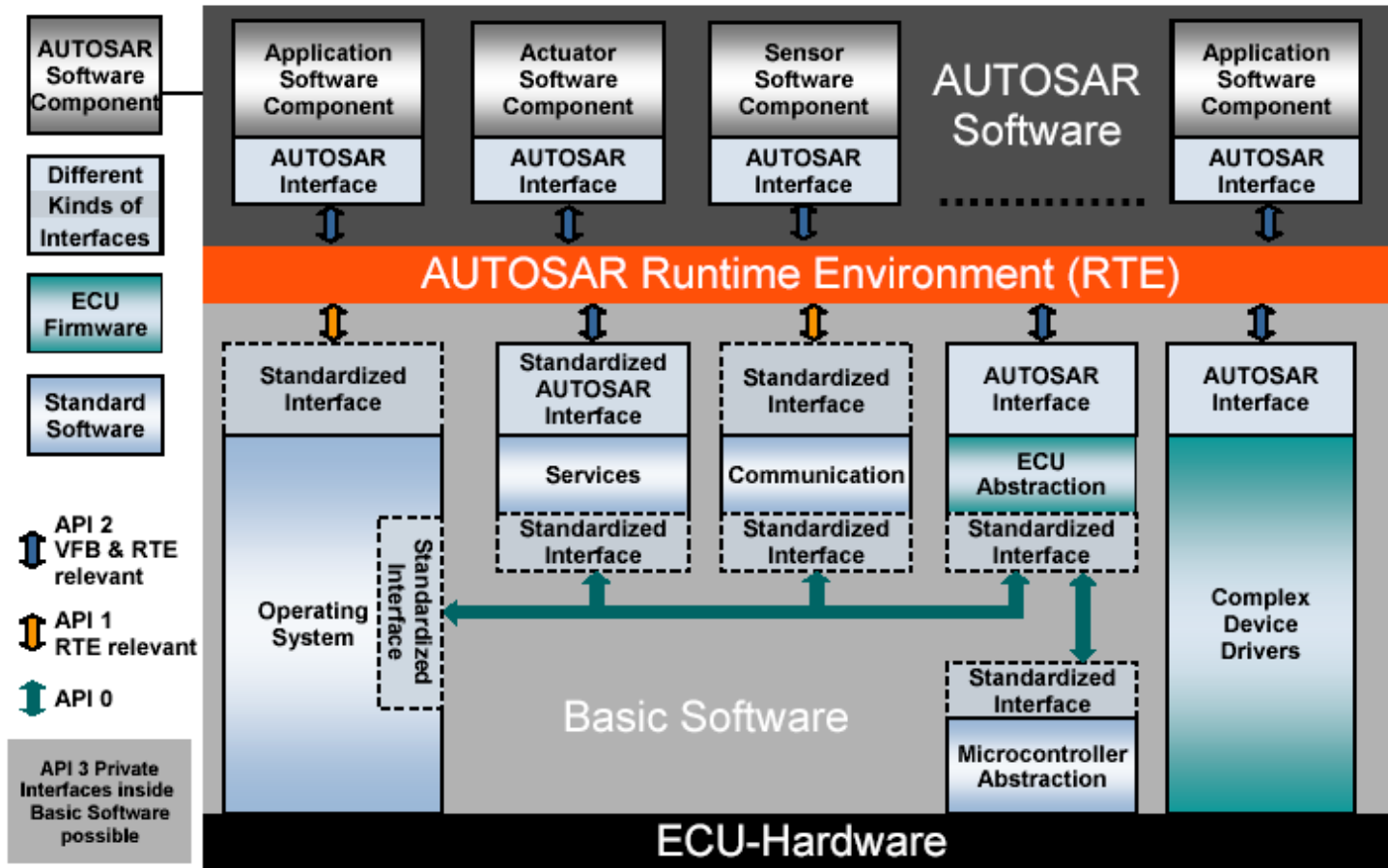
**TATA ELXSI LIMITED**  
engineering creativity

- Associate Member since 2006
  - MCAL Driver Development
  - BSW components development on ECAL and Services layer
  - AUTOSAR compliant component development service
  - AUTOSAR Authoring tool development service
- Premium Member (Phase II) from April 2008
  - WP 3.1 (Internal Validation)
  - WP 10.1 (Application Interface – Body)

The proposed presentation addresses the challenges faced in implementing a selected feature of Body Control Module in AUTOSAR infrastructure.

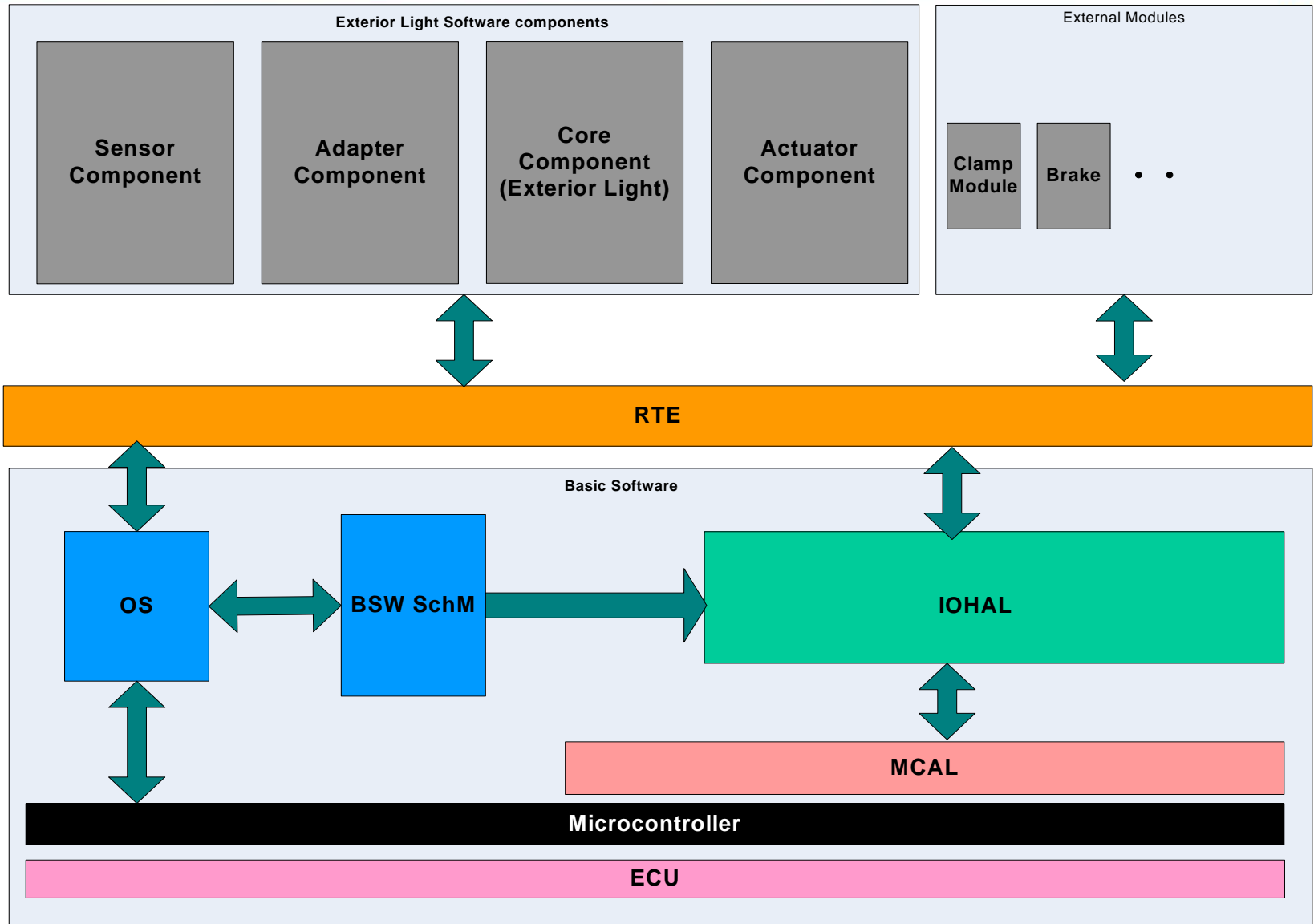
Implementation was based on an available working version of AUTOSAR.

# AUTOSAR Overview



“Cooperate on standards, compete on implementation”

# System Overview



- Software Component Description and Architectural Model
- RTE Contract Phase
- Implementation
- Development of Configuration Plug-ins
- BSW Configuration
- OS Configuration
- RTE Generation
- Integration

In the feature set implemented, software components were bundled in four groups as following:

- Sensor Component
- Adapter Component
- Core Component
- Actuator Component

Additionally IOHAL module is also modeled as a special software component as per IOHAL AUTOSAR Specification.



# Sensor Component



- Collects the user requests and provides filtered data via provided ports.
- The provided ports are standardized in AUTOSAR.
- The required ports are not standardized since this is ECU dependent.
- Required ports are interfaced with IOHAL SWC via Client\Server communication.



# Adapter Component



- Acts as a “translator” between OEM specific interfaces which are likely to change.
- Do preprocessing of the requested ports connected to sensor components and external components.
- Provide standardized mode requests and status signals.
- Both the provided and required ports are standardized in AUTOSAR.



# Core Component



- Handles feature's core functionality.
- Receives inputs from Adapter and other external modules.
- Both the provided and required ports are standardized in AUTOSAR.



# Actuator Component



- Drives the actuator outputs.
- Receives inputs from Core component .
- Both the provided and required ports are standardized in AUTOSAR.
- Provided ports interact with IOHAL (Client \ Server Communication) component via RTE to drive the outputs.



# IOHAL Software Component



- Treated as a special Software Component under RTE.
- Provides Read\Write services through Server ports for devices mapped under IOHAL.
- Provides data change notification to RTE via Send\ Receive ports.

- Created by interconnecting different software components via their ports.
- The output of SWC modeling phase is two .arxml files.
  1. Software Component Description arxml
  2. System Description arxml

# SWC Description - Challenges



- Building expertise on SWC template and tool to develop SWC description.
- Handling of interfaces which are not mentioned in selected module's AUTOSAR specification.

# SWC Description - Challenges



- Issue with triggering a software component above RTE by a service component for e.g IOHAL.

Created a runnable in service software component and associated the port event report function with this runnable. Following settings were made in Runnable trigger and access mode attributes.

<b>SI No:</b>	<b>Parameter</b>	<b>Service SWC</b>	<b>Sensor SWC</b>
1	Runnable Trigger	None	On Data Reception
2	Access Mode	Direct Write	Direct Read

- Software Component Description and Architectural Model
- RTE Contract Phase
- Implementation
- Development of Configuration Plug-ins
- BSW Configuration
- OS Configuration
- RTE Generation
- Integration



- Generation of RTE APIs required for the implementation of software components.
- Generated using RTE generator tool.
- The Software Component Description .arxml file is the input for this phase.
- One application header file will be generated for each software component.

# RTE Contract Phase- Challenges



The generated interfaces did not support outputs of called operation prototypes as return values in Client \Server interfaces. This required modifications in the interface of operation prototypes of server ports.

- Software Component Description and Architectural Model
- RTE Contract Phase
- Implementation
- Development of Configuration Plug-ins
- BSW Configuration
- OS Configuration
- RTE Generation
- Integration

# Implementation



Development of software components involves hand-coding as well as modeling using Matlab. Also involves development of IOHAL and other ECU specific complex drivers.

The required ports were accessed using the APIs generated in RTE contract phase.

# Implementation - Challenges



SWC implementation was using MATLAB required mapping to RTE interface manually by creating Legacy code.

- Software Component Description and Architectural Model
- RTE Contract Phase
- Implementation
- Development of Configuration Plug-ins
- BSW Configuration
- OS Configuration
- RTE Generation
- Integration

# Development of Configuration Plug-ins



Configuration plug-ins were developed for ECU specific complex drivers and for IOHAL component.

This task would require building additional expertise to generate tool-specific configuration interface.

- Software Component Description and Architectural Model
- RTE Contract Phase
- Implementation
- Development of Configuration Plug-ins
- BSW Configuration
- OS Configuration
- RTE Generation
- Integration

BSW Components used in this project are:

- Port
- DIO
- ECU Specific Devices
- IOHAL
- BSW Scheduler
- OS
- MCU

- Since BSW components are procured from third party, debugging involved debugging of third-party software which may add additional efforts.
- Variant issues can arise even if the microcontroller belongs to same family.

- Software Component Description and Architectural Model
- RTE Contract Phase
- Implementation
- Development of Configuration Plug-ins
- BSW Configuration
- OS Configuration
- RTE Generation
- Integration

# OS Configuration



SI No:	Component	Scheduler	Trigger
1	Sensor Adapter	AUTOSAR OS	Triggered by IOHAL SWC upon a data change
2	Core Component Actuator Component	AUTOSAR OS	10ms Cyclic Task
3	IOHAL Cyclic	BSW Scheduler	10ms Cyclic Alarm Task

Task frames for Sensor, Adapter, Core Component and Actuator Component will be present in generated RTE file(Rte.C)

Task frame for IOHAL cyclic function will be present in auto generated BSW scheduler file (Schm\_tasks.c).

In addition to these tasks, alarms\resources\ schedule tables generated for the RTE functionality will also be present in the OS configuration.

- Software Component Description and Architectural Model
- RTE Contract Phase
- Implementation
- Development of Configuration Plug-ins
- BSW Configuration
- OS Configuration
- RTE Generation
- Integration

- Input is the System Configuration Description.
- The configuration file is imported to RTE generator tool.
- Runnables of the software components are mapped to appropriate tasks.
- RTE configuration is exported to OS to generate the OS objects.

# RTE Generation - Challenges



- Since generated Rte.c is cumbersome, it is difficult to debug the same.
- RTE can be validated only after complete integration.

- Software Component Description and Architectural Model
- RTE Contract Phase
- Implementation
- Development of Configuration Plug-ins
- BSW Configuration
- OS Configuration
- RTE Generation
- Integration

- Integration of the software was done in three levels:
  1. Integration of SWC, OS and RTE
  2. Integration of IOHAL and OS
  3. Integration of entire software
- Synchronization between RTE and OS.

- Due to the inter-dependency of the modules, integration would require configuration of additional modules.
- Low-level integration would require implementation of many stub functions which adds additional efforts.

# General Challenges



- Information on AUTOSAR implementation is spread over various documents. This would consume project execution time when an issue is faced.
- A document (e.g. a FAQ Document) to address possible problems that may occur during AUTOSAR implementation would be preferable.
- Improvement in performance would not be an easy task due to the presence of third-party supplied and auto generated software.

# Comparison with Conventional Development Methods



RAM Usage – Higher by 26%

ROM Usage – Higher by 22%

Throughput – Lesser by 14%

Thank You