Dr. Eric Armengaud
VIF - Area E
Group leader embedded systems

January 10^{th}, 2011

Model-based development and test of distributed automotive embedded systems

bmvit Bundesministerium für Verkehr, Innovation und Technologie
bmwfi Bundesministerium für Wirtschaft, Familie und Jugend
FFG
COMET
Das Land Steiermark
SFG Im Auftrag des Wirtschaftslandesrates
TU Graz Graz University of Technology

virtual vehicle
Vehicle EE & Software

Electronics in car

Software Engineering for automotive embedded systems

The time-triggered architecture & FlexRay

Research activities @ Virtual Vehicle Competence Center

VIRTUAL VEHICLE in a nutshell:

Founded: **July 2002**

Current Staff: **150**

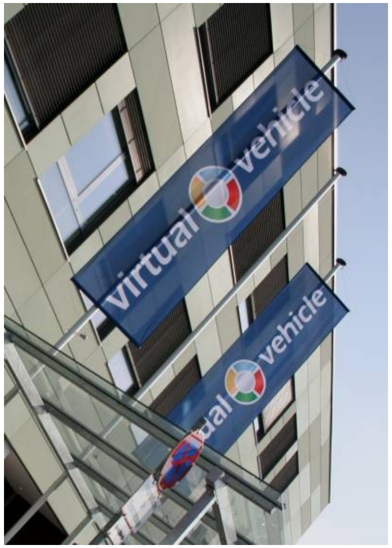
Turnover: **EUR 12 Mio.**

Shareholder:

TU Graz – Graz University of Technology	40%
AVL	19%
MAGNA STEYR	19%
SIEMENS	12%
JOANNEUM RESEARCH	10%

Managing Director: **Dr. Jost Bernasch**

Scientific Director: **Prof. Hermann Steffan**
(Vehicle Safety / Frank Stronach Institute, TU Graz)



Independent Research Platform

(not tied to specific bodies or corporations)

Applied Research and Scientific Services

Driven by the demand of leading companies

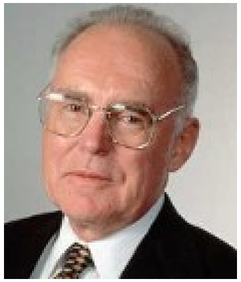
(> 50 industry partners)

Comprehensive international Research Network

(> 35 scientific partners and university institutes)

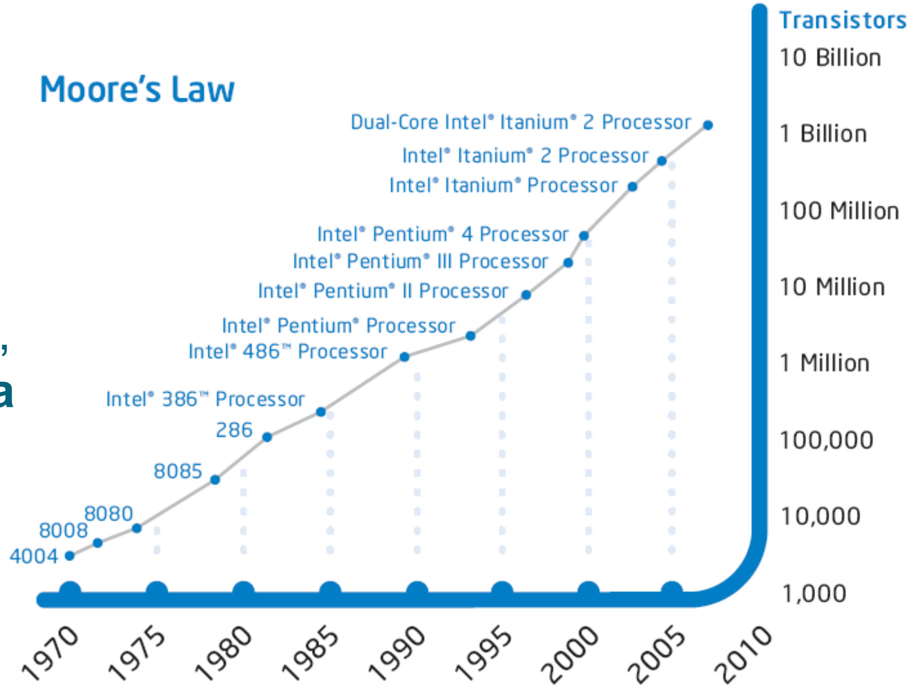
Extensive financial funding programs available

(no overhead as in customary funded projects)



"If the automotive industry had advanced as rapidly as the semiconductor industry we'd all be **driving a Rolls Royce**, it would do **half a million miles to the gallon** and it would be **cheaper to throw away than to park**"

And as a friend pointed out, Moore said, "it would only be a **half-inch long and a quarter-inch high**."



Source: AVL List

PAST → TODAY → FUTURE ?

Vehicles a decade ago

- A few embedded systems per vehicle

Vehicles nowadays

- Up to a few hundreds of computing devices per vehicle
- Multiple networks per vehicle

Advantage

- Safety-critical embedded systems have been **key innovation drivers**
- E.g. by-wire systems

Disadvantage

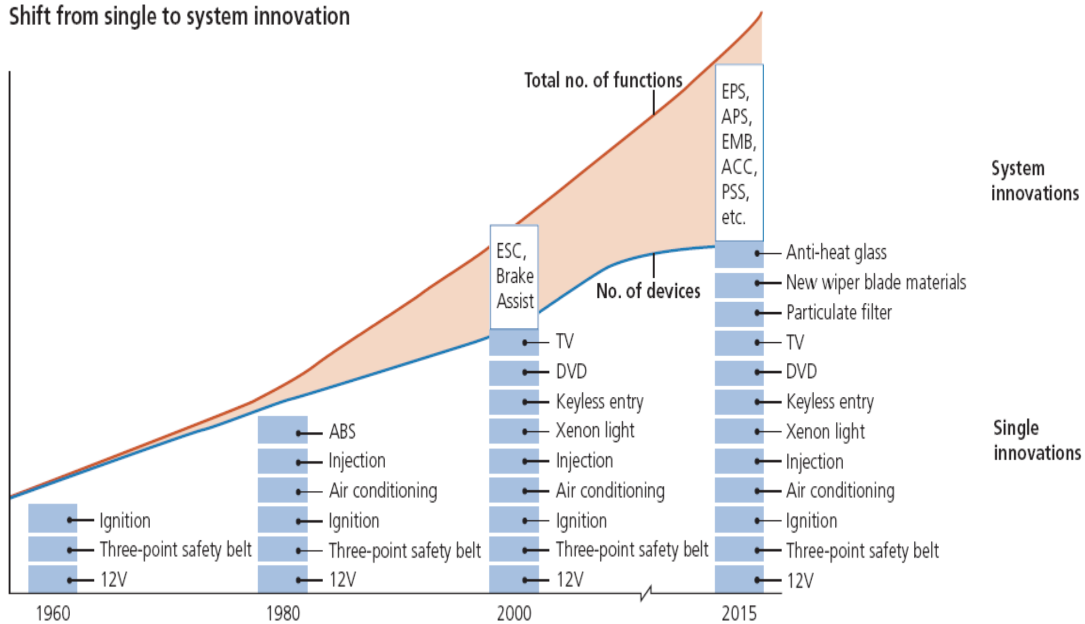
- Enormous **complexity** is challenging industry (**automotive, aerospace, rail, automation**)
- Increasing costs
- Affected product quality ➔ safety-critical

virtual vehicle
Vehicle EE & Software

R&D spending in automotive industry:

- In 2005 € 68 billion in research & development
  - 4,2 % of sales or € 783 per vehicle
- Additionally € 1.500 of cost reduction per vehicle forecasted (11 % of costs)
- Through 2015, R&D will rise to € 800 billion
- E&E will remain the most important enabler for automotive innovations

Shift from single to system innovation



Please note: ABS = anti-lock braking system, ESC = electronic stability control, EPS = electronic power steering, APS = adaptive power steering, EMB = electro-mechanical braking, ACC = adaptive cruise control, PSS = predictive safety systems

[H. Gall, austriamicrosystems]

80% of the innovation in cars comes from electronics – Software plays a key role

Evolution of the complexity

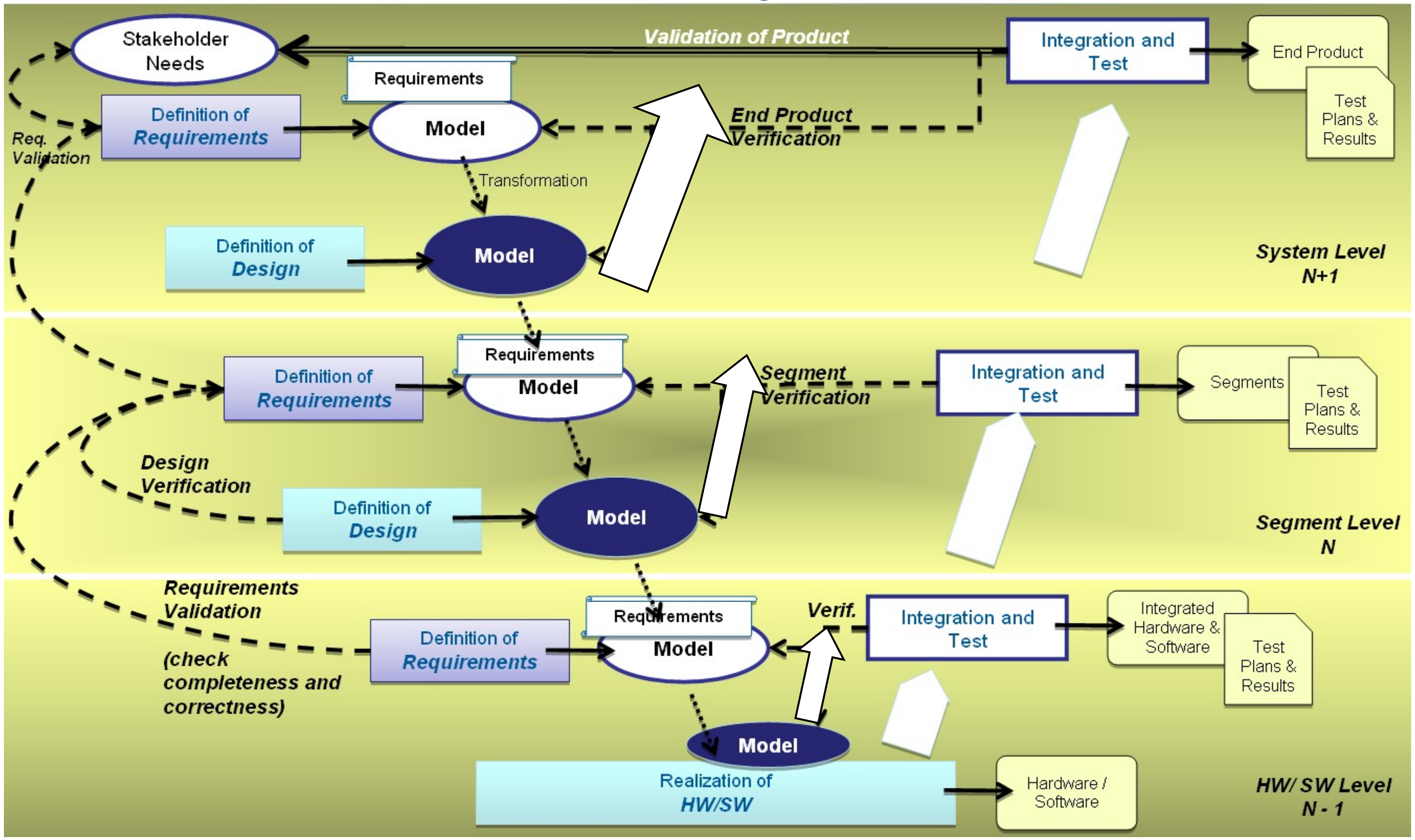
- Programming language: assembler in the 1970's, C in the 90's, Matlab/Simulink (ASCET) nowadays
- 100 millions lines of code
- up to 80 ECUs
- 2500 signals
- 65 millions cars and light commercial vehicles produced each year
- Large development teams regrouping different domains and different institutions

Requirements on automotive electronics

- High reliability
- Functional safety
- Real-time behavior
- Minimized resource consumption
- Robust design

virtual vehicle Vehicle EE & Software

The model-based approach for early validation and automatic translations



But, different kinds of models for different skills (dysfunctional models for safety, performance models for timing constraints, …)

[CESAR Project, O. Laurent, AIRBUS]

Methods for requirement engineering

- First description of the system; contract between OEM and supplier
- Requirements needs to be precise, unambiguous and complete
- Formalization of multi viewpoint, multi criteria and multi level requirements

Methods for component-based design

- Global understanding of the system for efficient analysis
- Provide traceability during system design and validation
- Design space exploration comprising multi-view, multi-criteria and multi level architecture trade-offs

Safety methods and processes

- Ensure the quality of a product via the execution of safety related activities and the definition of a standardized development process
- Provide traceability of the development process
- Formalization of the dev. process for analysis, reporting (certification) and automation (service orchestration)

Software Engineering

"Software engineering (SE) is a profession dedicated to designing, implementing, and modifying software so that it is of higher quality, more affordable, maintainable, and faster to build. It is a systematic approach to the analysis, design, assessment, implementation, test, maintenance and re-engineering of a software by applying engineering to the software"

[Wikipedia]

Requirements engineering:

- deals with understanding, documenting, communicating and implementing customer needs
- is required to reach a common understanding between the stakeholders
- is required during the entire development cycle (design, implementation, validation)

Related activities

- *Requirements elicitation*: find out the services the system should provide and the operational constraints
- *Requirements analysis and negotiation*: solve the conflicts between the requirements in order to reach a common understanding between the stakeholders
- *Requirements documentation and validation*: write down and check the requirements against correctness, completeness, consistency, verifiability, unambiguity, traceability…
- *Requirements management*: managing requirements changes (keep the requirement set consistent)

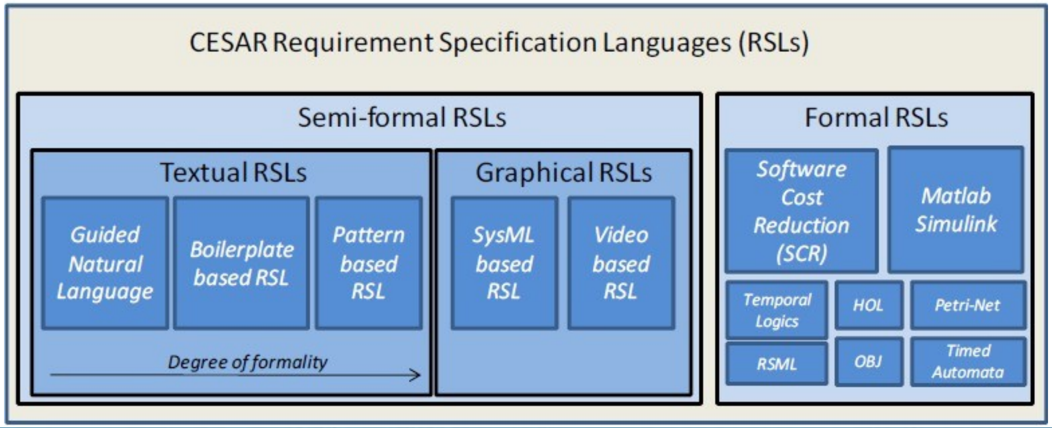
virtual vehicle
Vehicle EE & Software

Requirement Specification Language

- minimize amount of time to write requirements
- make requirements understandable and unambiguous
- minimize amount of time to validate requirements
- differences in:
  - **formality: formal/semi-formal/informal**
  - **illustration: textual/graphical/tabular**

Requirement Meta-Model

- capturing, managing and organizing requirements into a formalized structure
- providing the meta-model for each RSL
- providing the interoperability model for tools



[MEPAS Project, N. Marko]

Free text: no constraint
➔ no training required
e.g.: the system shall count time between eyelid movement and warn driver if the time is less than 2 sec

Guided natural language: limited vocabulary from a dictionary
➔ reduce ambiguity
e.g.: **driver**: person who drives the car // **warn**: inform the driver about an event

Structured textual: template for requirement description
➔ further reduce ambiguity, support transition to formal notations
e.g. IF <trigger> THEN <subject> SHALL DO <action list> WITHIN <time bound>

Semi-formal model-based: formal and precise syntax while their semantics are imprecise and allow different interpretation
➔ support the analysis of the requirements
e.g.: UML modeling

Formal model-based: method for definite, orderly and methodical requirement definition
➔ most precise requirement definition
e.g. Petri nets, timed automata

Traceability: „Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction, …"

post-requirements traceability links

- satisfies
- verify
- realize

pre-requirements traceability links

- explicit traceability links
  - owns
  - hasRationale
  - hasSource
- possible operations performed on requirements
  - refine
  - decompose
  - copy
  - depend

Motivation

- Describe in a formalized way the different artefacts of a system
  ➔ improved specification of the system
- Explicitly link the different artefacts together
  ➔ improved analysis and optimization capabilities of the system
- Provide a computer-based framework
  ➔ Support engineers during development activities and improve tool interaction

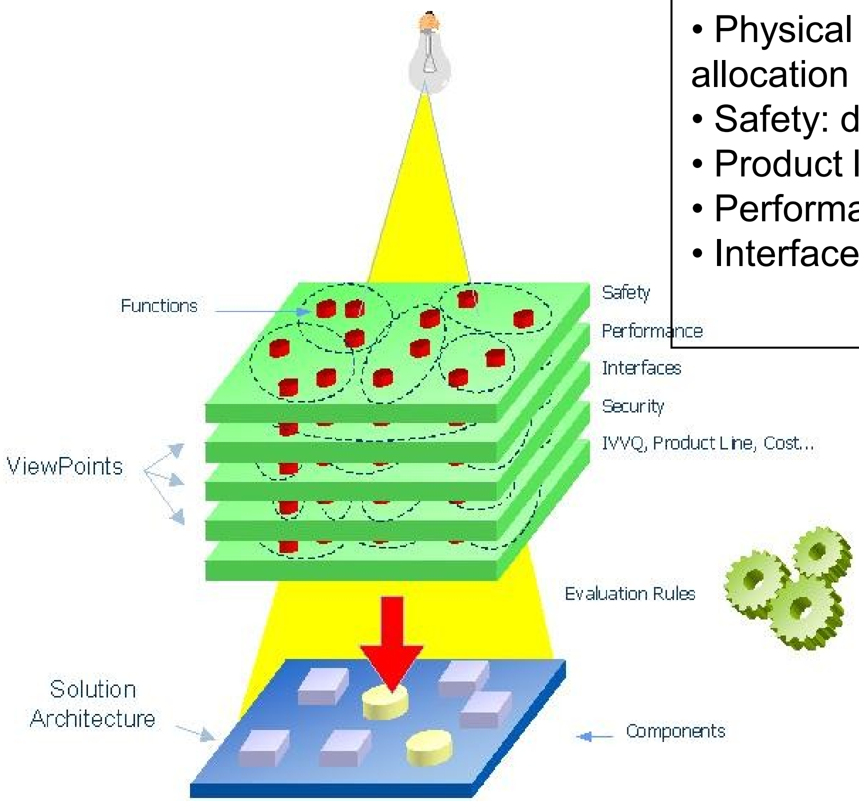
Some (non-functional) modeling languages for automotive domain

- EAST-ADL: architecture description language tailored for the automotive domain (www.atesst.org)
- AUTOSAR: AUTomotiv Open System Architecture (www.autosar.org)
- FIBEX: Field Bus Exchange Format (www.asam.net)
- TIMMO: Timing Model (www.timmo.org)

A modeling language supports your development work but will NEVER take away the intellectual work of creating and understanding your system

possible views :
• Operational: focus on the system missions
• Functional: focus on the functional aspects of the system
• Logical: define system architecture, define abstract components, allocation of functions on them, behaviour of components and interfaces between components
• Physical : define concrete hardware and software components, allocation of functions on hardware and software components
• Safety: define the dysfunctional aspects of the system
• Product line: define the variability points
• Performance: define the system performance
• Interface: define the interfaces of the system components

**The first challenge:
Identify the relevant views**

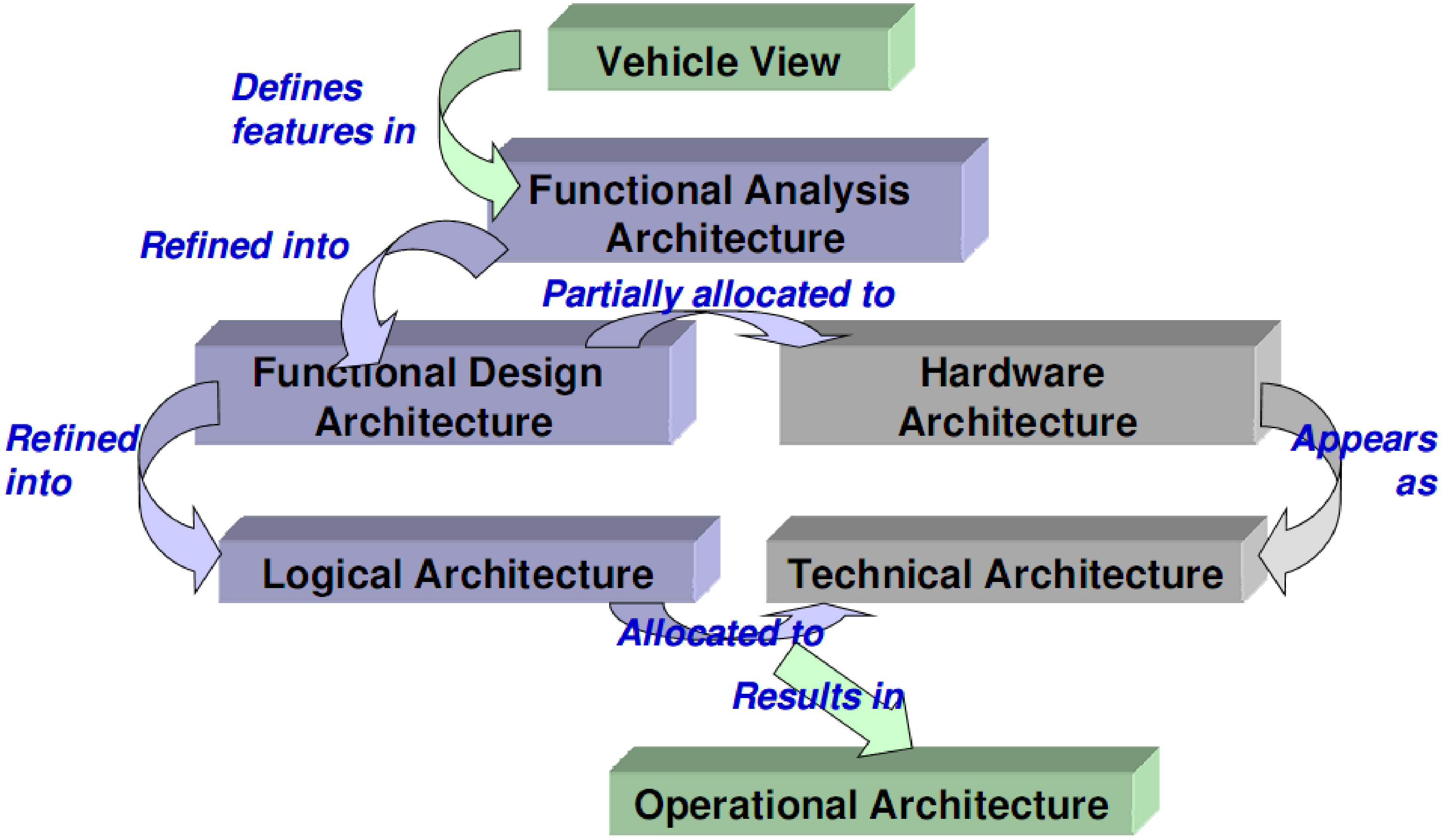


The multi-views prism
[CESAR Project, O. Laurent, AIRBUS]

**The second challenge:
Propose the appropriate foundations to share common data between the different views**

virtual vehicle
Vehicle EE & Software

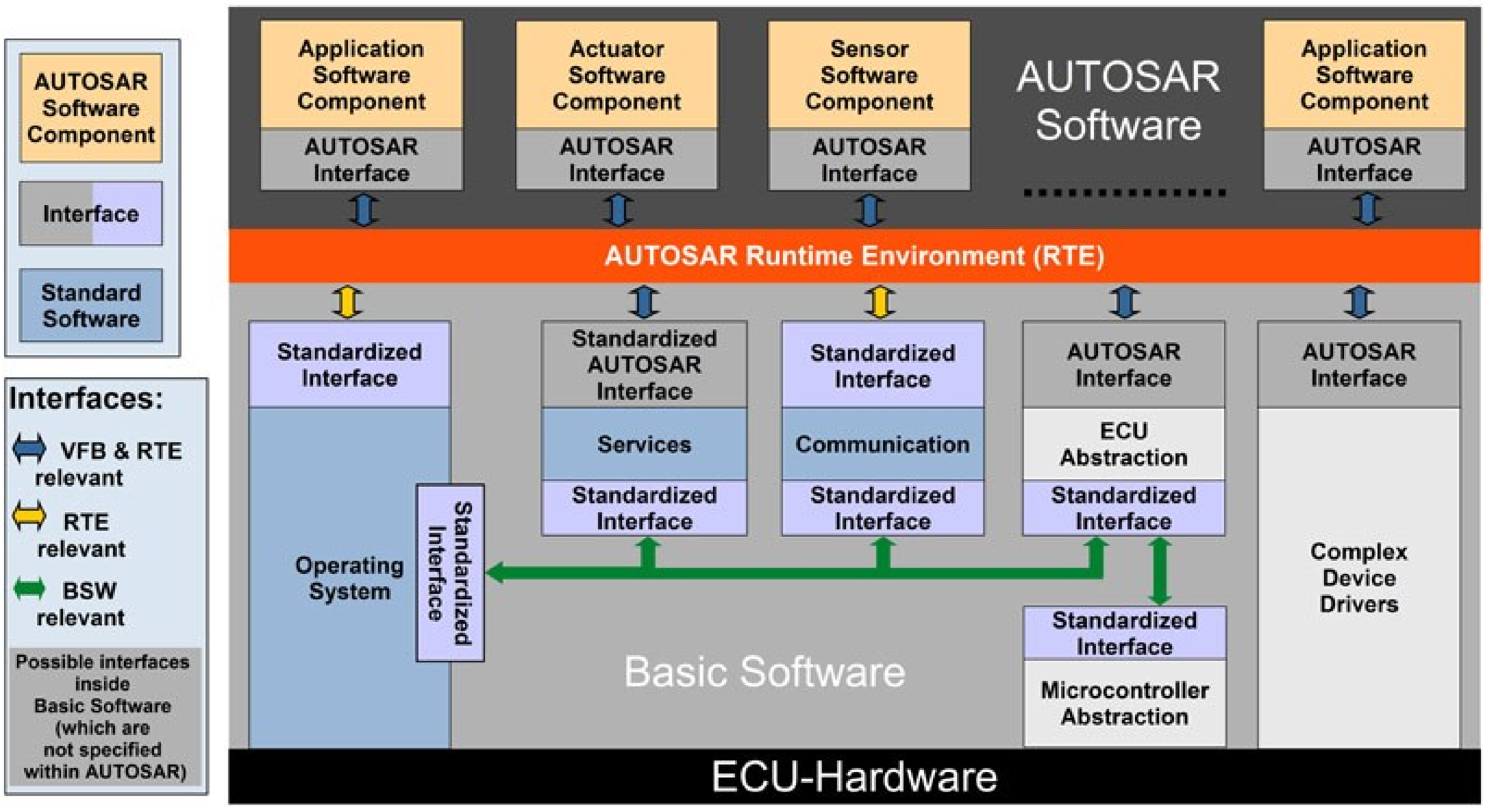
EAST-ADL is an architecture description language with improved means for capturing the requirements, characteristics and configurations of cooperative systems and the related analysis and V&V.



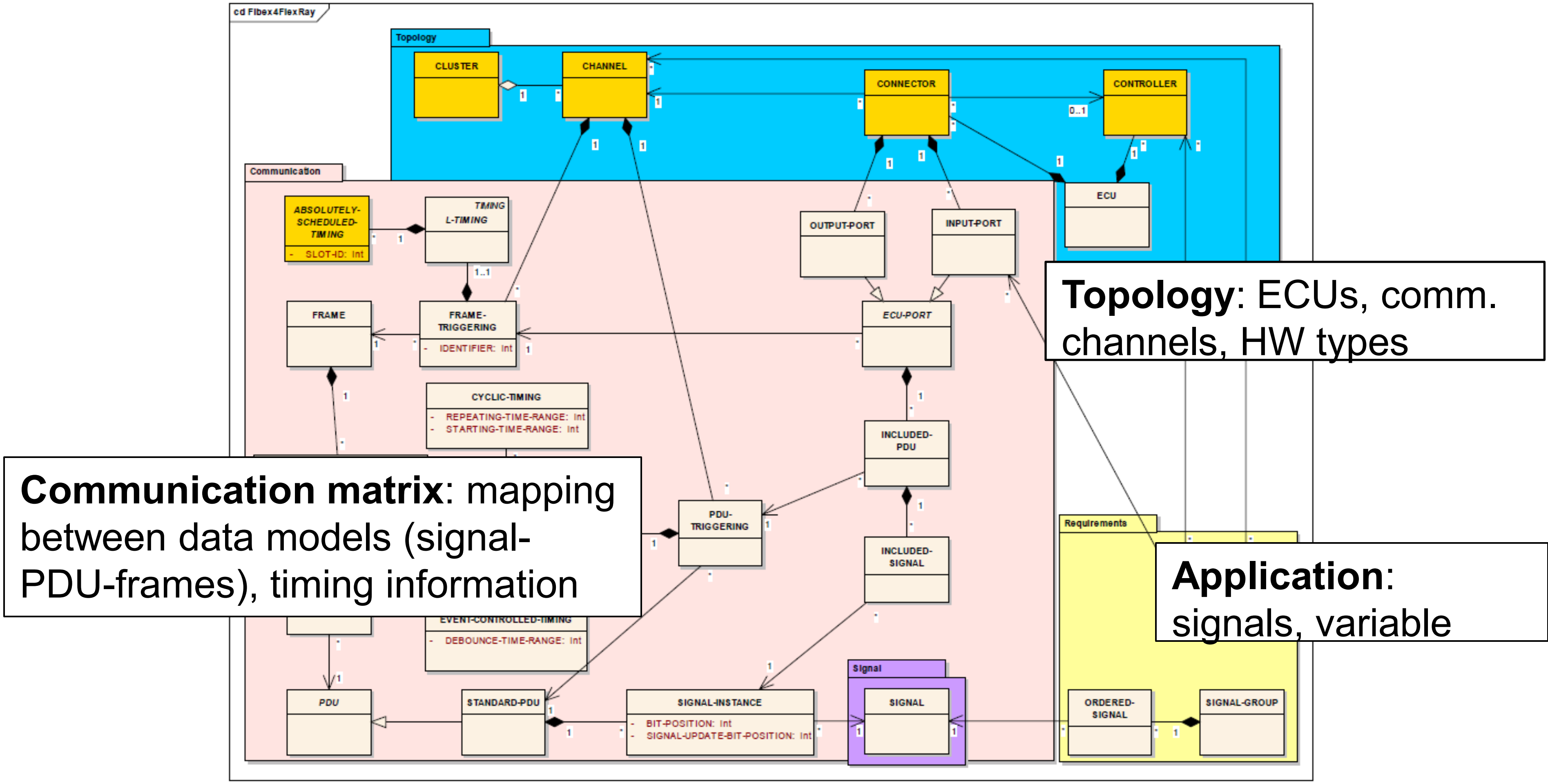
virtual vehicle
Vehicle EE & Software

AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers.

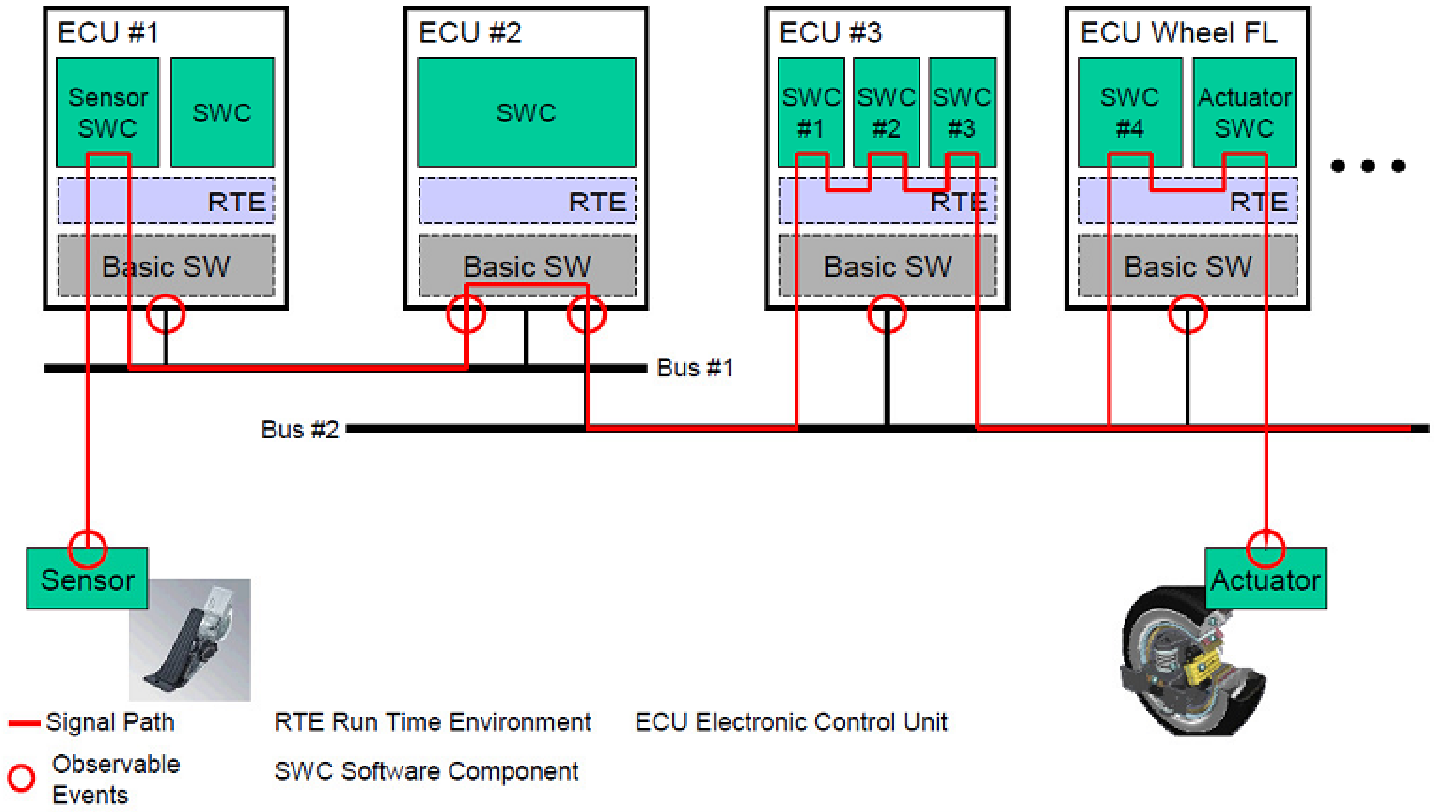
Interfaces
Components and interfaces view (simplified)



FIBEX is an XML-based standardised format used for representing the networks used in the automobile. It has extensibility required for the various network protocols used.



TIMMO develops a common, standardized infrastructure for the handling of timing information during the design of embedded real-time systems in the automotive industry. This shortens the development cycle and increases its predictability.

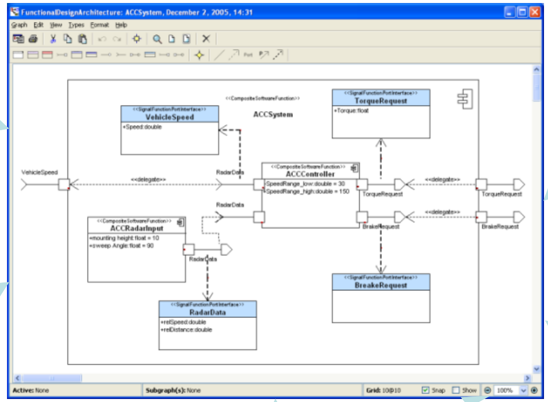


Seamless modeling - vision

Requirements

ID		Requirement	Priority
HL_Req_1	1 Calling the elevator	False	
HL_Req_2	A potential passenger can be on any of the floors and can call an elevator by pressing either the up or button to call the elevator.	True	High
HL_Req_3	The potential passenger waits for the doors to open before entering into the elevator. The potential passenger now becomes a passenger	False	
HL_Req_6	2 In the elevator	False	
HL_Req_7	Once in an elevator, a passenger can select the floor or a number of floors where he wants to go to. Modif	True	Medium
HL_Req_8	Each elevator will have a list of floors to visit : Once the elevator has been called by a potential passenger or a passenger has selected a destination, then the elevator will move to the appropriate floor.	False	
HL_Req_9	3 Elevator at selected floor	False	
HL_Req_10	When the elevator has arrived at a floor and the doors have opened, then the passenger can exit the elevator. xxxxx	True	High

system architecture specification and management (EAST-ADL)



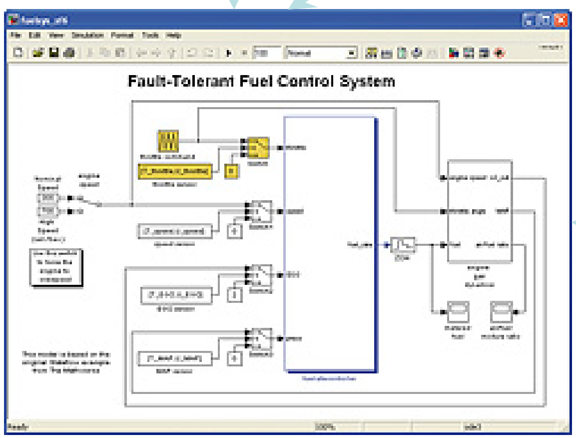
Safety analysis (HiP-HOPS)

Table 1 Effects of failures of the braking system

Failure	*Effect*	*Severity*	*Notes*
NoBrake4	Loss of braking in all wheels	Catastrophic	100% loss of braking function
NoBrake3	Loss of braking in three wheels	Catastrophic	80% loss of braking function
NoBrakeFront	Loss of braking in front wheels	Catastrophic	65% loss of braking function, 30% loss of car stability
NoBrakeRear	Loss of braking in rear wheels	Critical	35% loss of braking function, 30% loss of car stability
NoBrakeDiag	Loss of braking in diagonally opposite wheels	Critical	50% loss of braking, 15% loss of stability, 15% loss of steering
NoBrake1	Loss of braking in single wheel	Critical	Marginal unless car is braking while on a curved trajectory, in which case it may drift off course
ValLow4	Insufficient braking in all wheels	Catastrophic	Major loss of braking function
ValHigh4	Excess braking in all wheels	Critical	Excess braking can lead to loss of steerability if wheels lock, but stability is mostly maintained

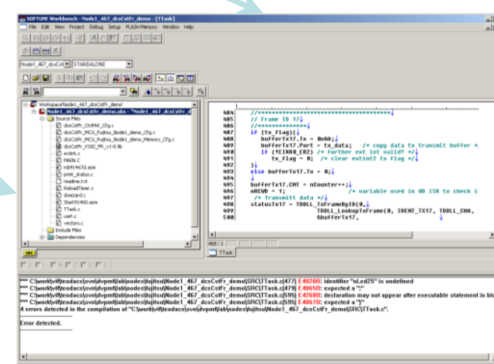
Source: Papadopoulos and McDermid (1999)

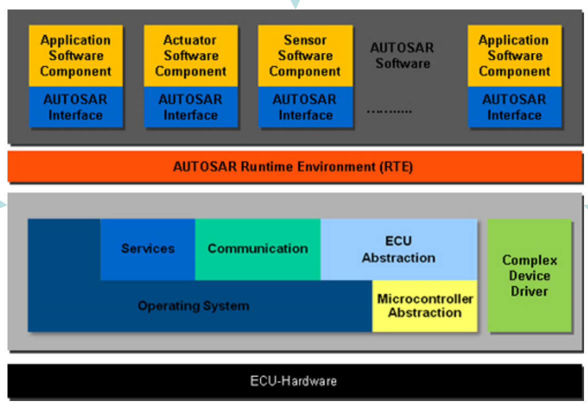
behavioral modeling (Matlab / Simulink)



Further static analysis

IDE





architecture modeling (AUTOSAR)

The ECUs are deployed for safety-relevant operations (e.g., car movement, power distribution, vehicle stability), where a failure can harm people, environment or property and has therefore to be avoided.

IEC 61508: "Functional safety of electrical/electronic/programmable electronic safety-related systems"

- Basic functional safety standard applicable to all kinds of industry
- Published 1998, since 2001 as European norm
- Covers the entire development cycle (16 phases covering analysis, realization, operation
- Central concepts *risk* and *safety function*
- Philosophy
  - zero risk can never be reached
  - safety must be considered from the beginning
  - non-tolerable risks must be reduced (ALARP - as low as reasonably practicable)

ISO 26262: "Road vehicles – Functional safety" (www.iso.org)

- Based on IEC 65801
- Defines safety process for the development of road vehicles
- Draft International Standard – will be released in 2011

Safety

- Freedom from unacceptable risk
- Risk: combination between probability and severity of a failure

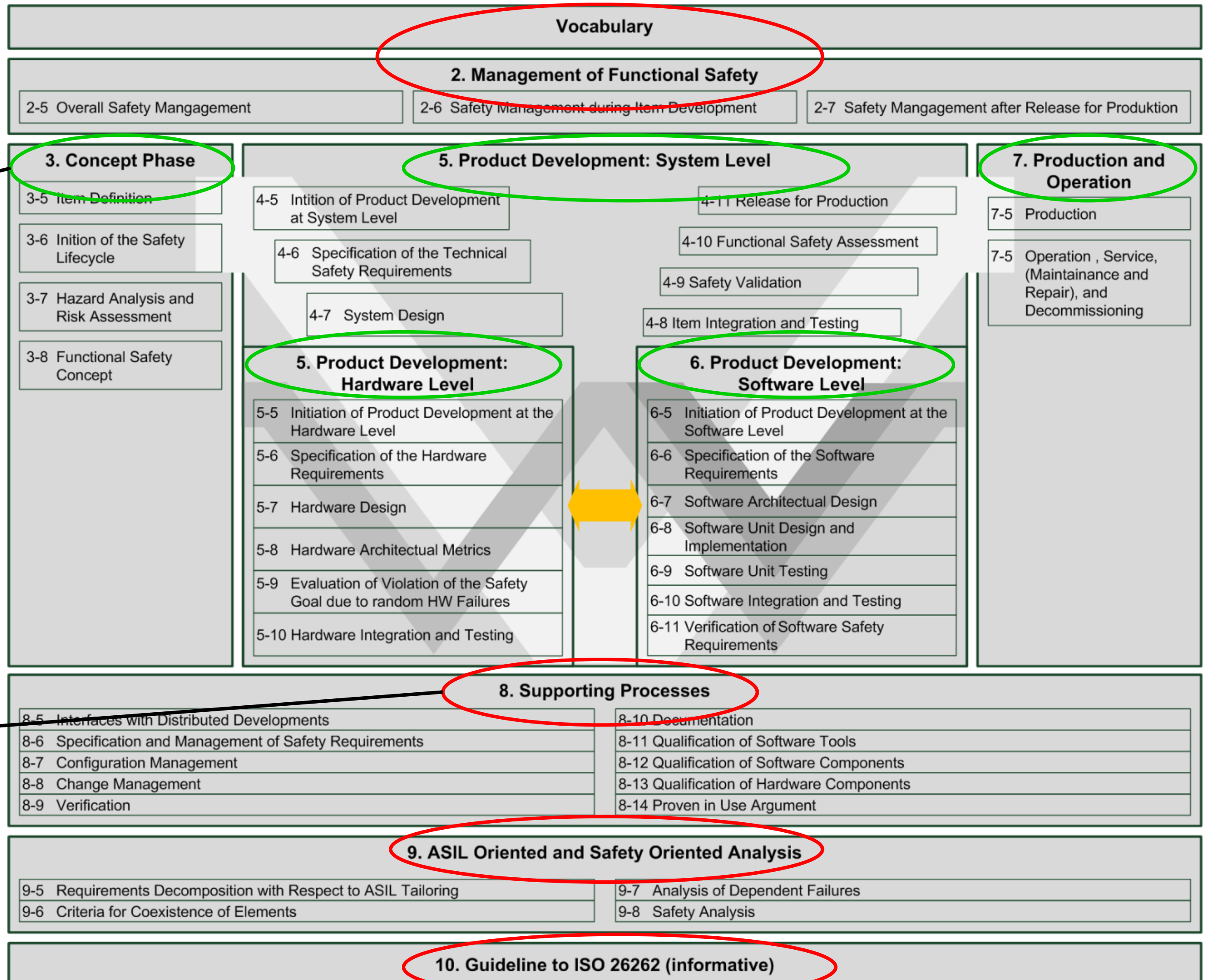
Safety related project activities

- **Risk**: Hazard and risk analysis
  e.g. "what could happen if"
- **Safety**: safety concept
  e.g. "what is the safe state"
- **Safety functions**: safety requirements
  e.g. "How to provide the safe state"
- **SIL decomposition**: Implementation and processes
  e.g. "what SIL (Safety Integrity Level) applies for individual units"

virtual vehicle
Vehicle EE & Software

Standardized development processes including safety-related activities

Accompanying processes



[ISO DIS 26262]

Safety concept

- Is required to analyze systematically the risks of the controlled system in its environment
- Provides additional requirements to the system
- Has a direct influence on the system architecture and functionality (safety functions)

Traceability of the development process

- It must be ensured that the standardized development process has been followed (audit from external companies)
- The tool chain must be reasonably reliable (classification and qualification activities)

Needs for Software Engineering

- Development activities are part of the ISO 26262 (“What” - which kind of test, review, analysis)
- However specific development methods are not part of the ISO 26262 (“How”)
- Systematic approach for designing, implementing, and modifying the software is required to improve system quality while minimizing the costs

Time-triggered architectures for complex control applications

"… in the ***event-triggered*** *approach, all communication and processing activities are initiated whenever a significant change of state, i.e., an event (e.g., interrupt), is noted. In the* ***time-triggered*** *approach, all communication and processing activities are initiated at predetermined points in time."*

[Real-Time Systems, Kopetz, 1997, Kluwer Acacemic]

Example: Volkswagen Golf

1976: Golf I

0 ECU's

2006: Golf VI

Up to 48 ECU's

2004: Golf V

Up to 35 ECU's

1983: Golf II

5 ECU's

1991: Golf III

11 ECU's

1998: Golf IV

18 ECU's

virtual vehicle
Vehicle EE & Software

Snapshot 2004: the VW Phaeton

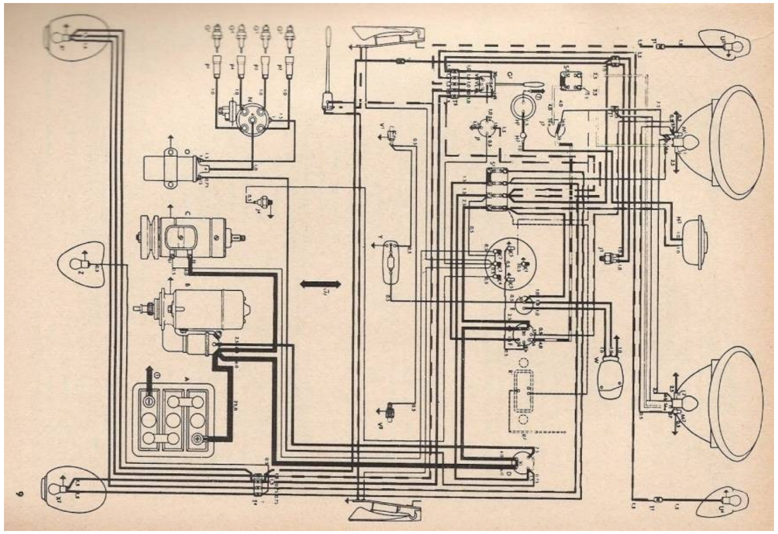
- 2110 cables
- 3860 meters cable
- Weight: 64kg
- 70 ECUs

Advantage

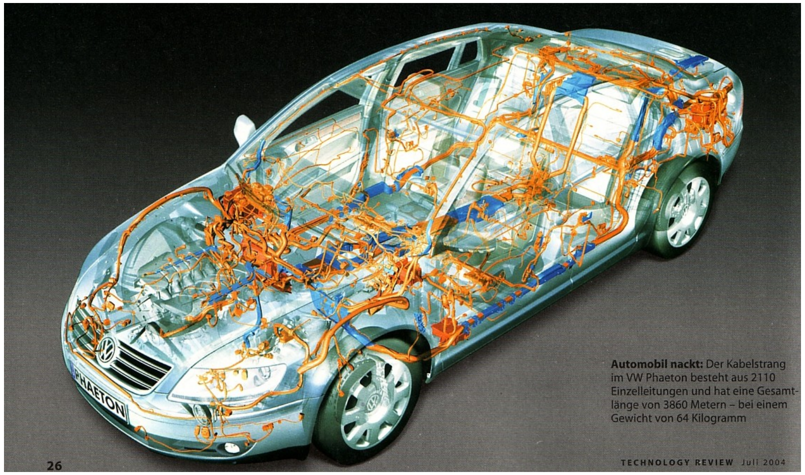
- Safety-critical embedded systems have been key innovation drivers
- E.g. by-wire systems

Disadvantage

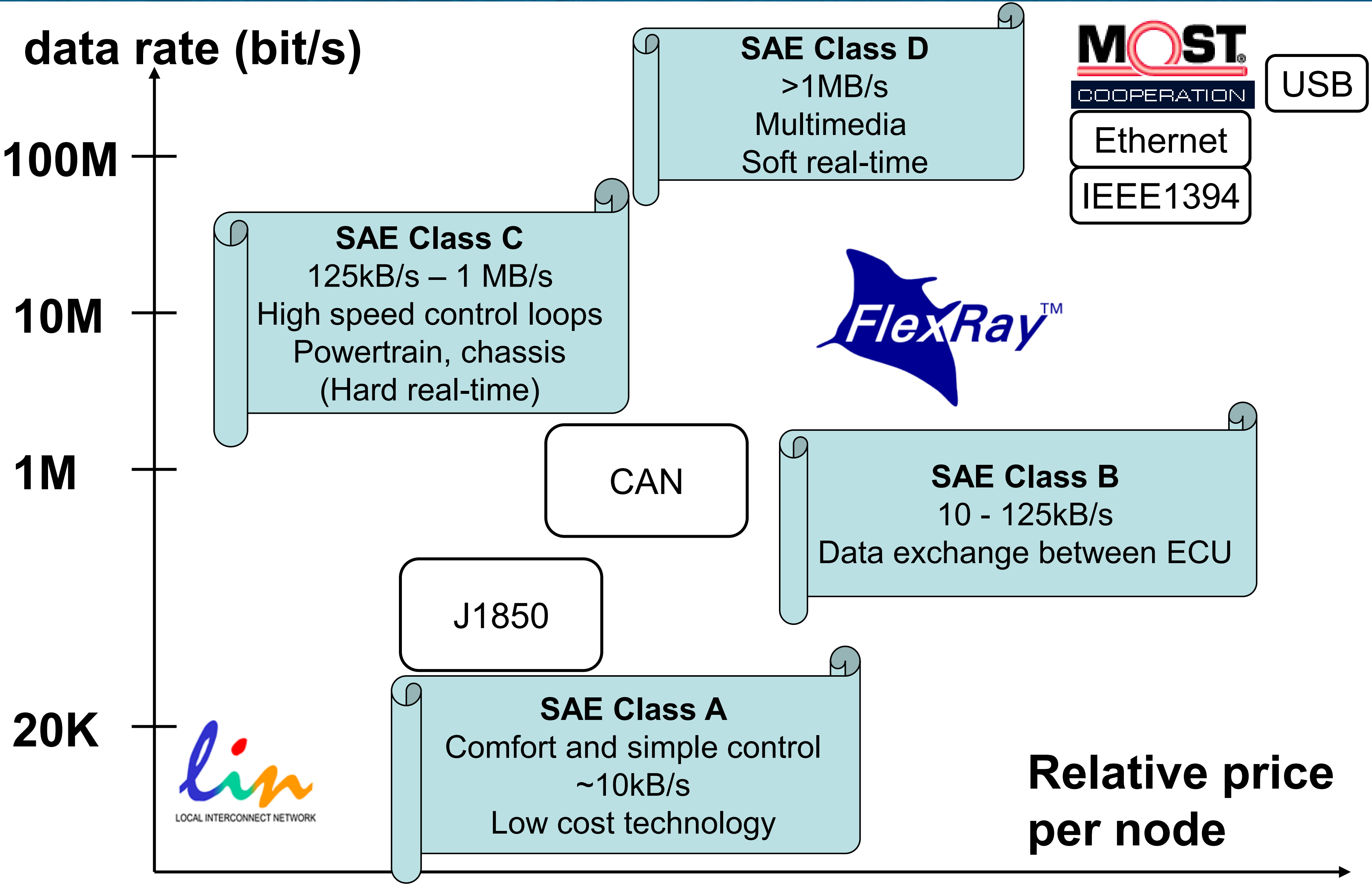
- Enormous complexity is challenging industry (automotive, aerospace, rail, automation)
- Increasing costs
- Affected product quality ➔ safety-critical



Source: Volkswagen Beetle, 1960



Source: Technology review, July 2004


data rate (bit/s)
100M
10M
1M
20K
SAE Class D
>1MB/s
Multimedia
Soft real-time
MOST COOPERATION
USB
Ethernet
IEEE1394
SAE Class C
125kB/s – 1 MB/s
High speed control loops
Powertrain, chassis
(Hard real-time)
FlexRay™
CAN
SAE Class B
10 - 125kB/s
Data exchange between ECU
J1850
SAE Class A
Comfort and simple control
~10kB/s
Low cost technology
lin
LOCAL INTERCONNECT NETWORK
Relative price
per node

Automotive electronics organized as complex distributed systems

- Local connection between sensors, processors and actuators
- Information dissemination within the car
- Point to point connection inefficient (reliability, weight)

System complexity difficult to manage

- Number of ECU, intensity of the communication
- Different technologies
- Complexity of the application

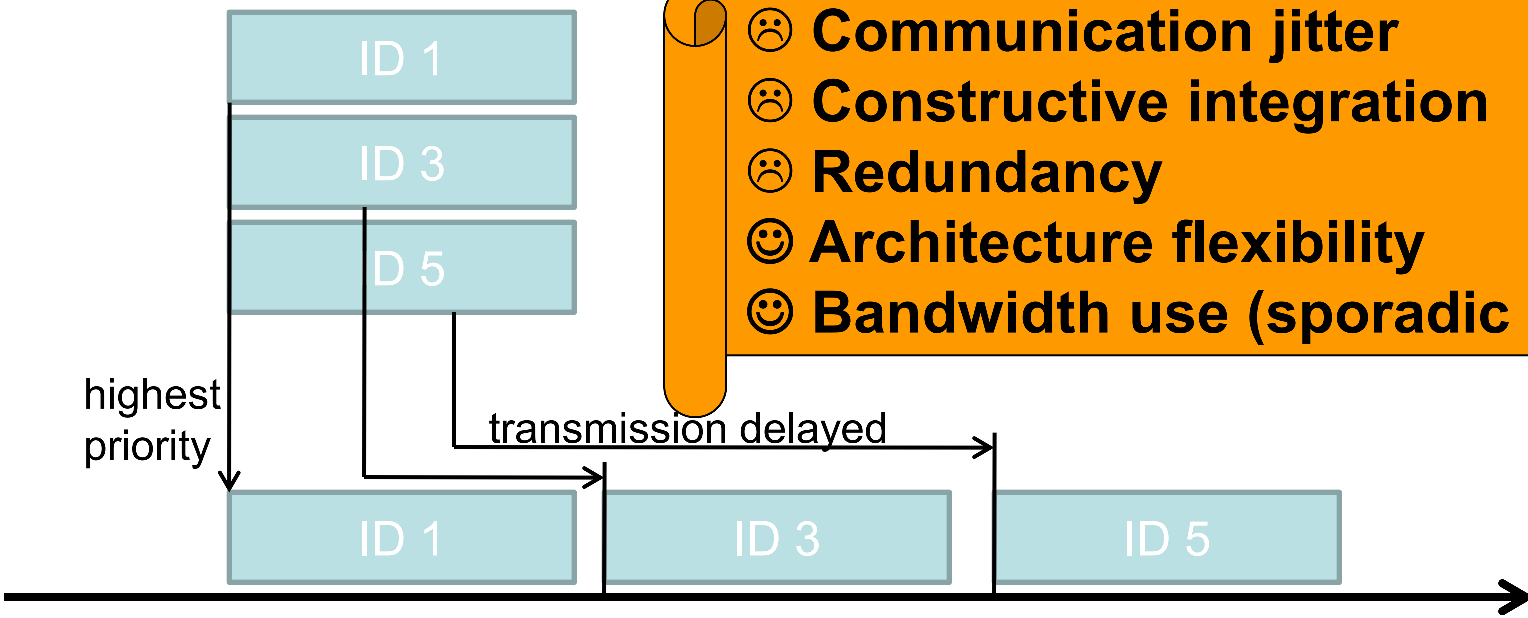
The system can not be assumed fault-free

- High temperature range and thermal gradients
- High humidity, splashes from oil, petrol, chemicals…
- Conducted emissions (electric motors) and radiated emissions (power lines, radio or TV transmitters)

Event-triggered architecture

- **System activity triggered by an event**
- **Priority based communication (CAN)**

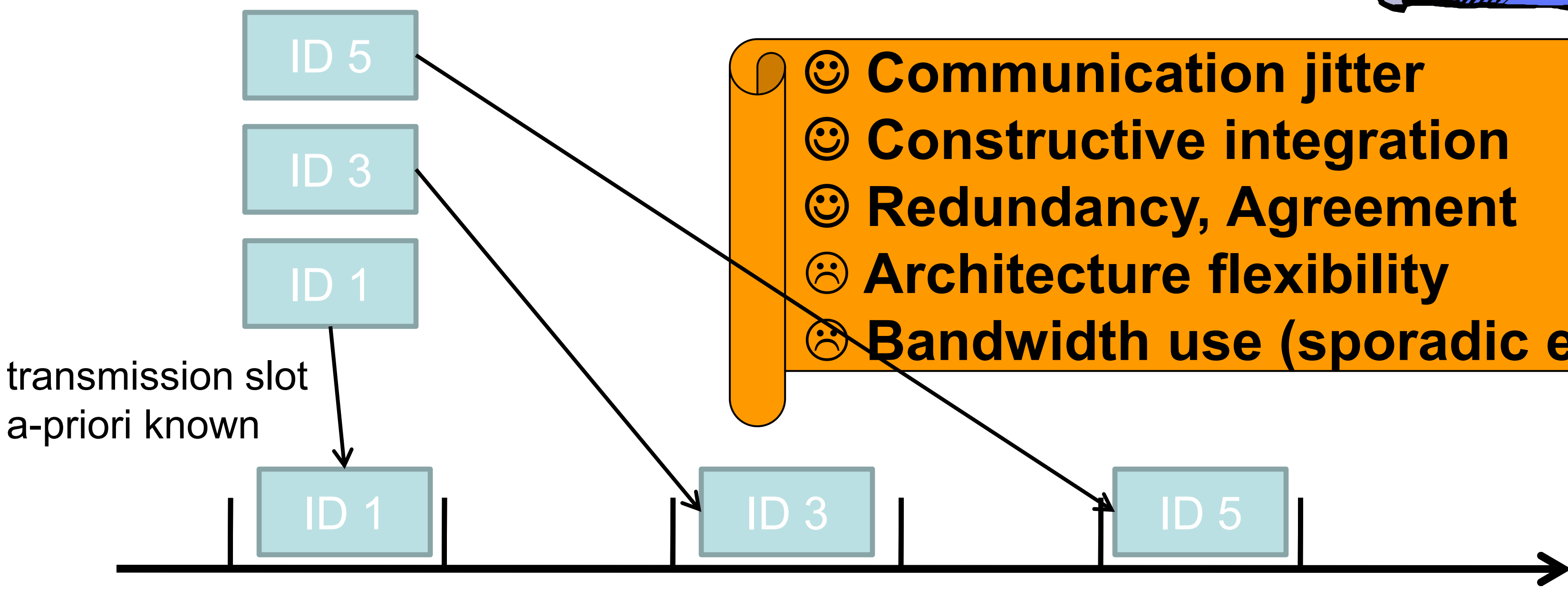
☹ Communication jitter
☹ Constructive integration
☹ Redundancy
☺ Architecture flexibility
☺ Bandwidth use (sporadic events)



Time-triggered architecture

- Action derived from progression of time
- Static, periodic, a-priori known schedule
- Global notion of time

- ☺ Communication jitter
- ☺ Constructive integration
- ☺ Redundancy, Agreement
- ☹ Architecture flexibility
- ☹ Bandwidth use (sporadic events)



Event-based communication

- A communication is triggered for **each new event** – i.e. major state change (e.g. temperature increase of +5 degree)
- Each event (communication) has to be detected and processed in the same time order it arrived
- Optimal use of the bandwidth
- Not robust – lost of message might lead to system inconsistencies

Status-based communication

- Periodic communication for **updating system state** (e.g. temperature is currently 55 degree)
- Events (communication elements) might be missed or processed in different time order than reception time
- Worse-case use of the bandwidth
- Robustness: lost of message only induce additional processing delays – no system inconsistencies

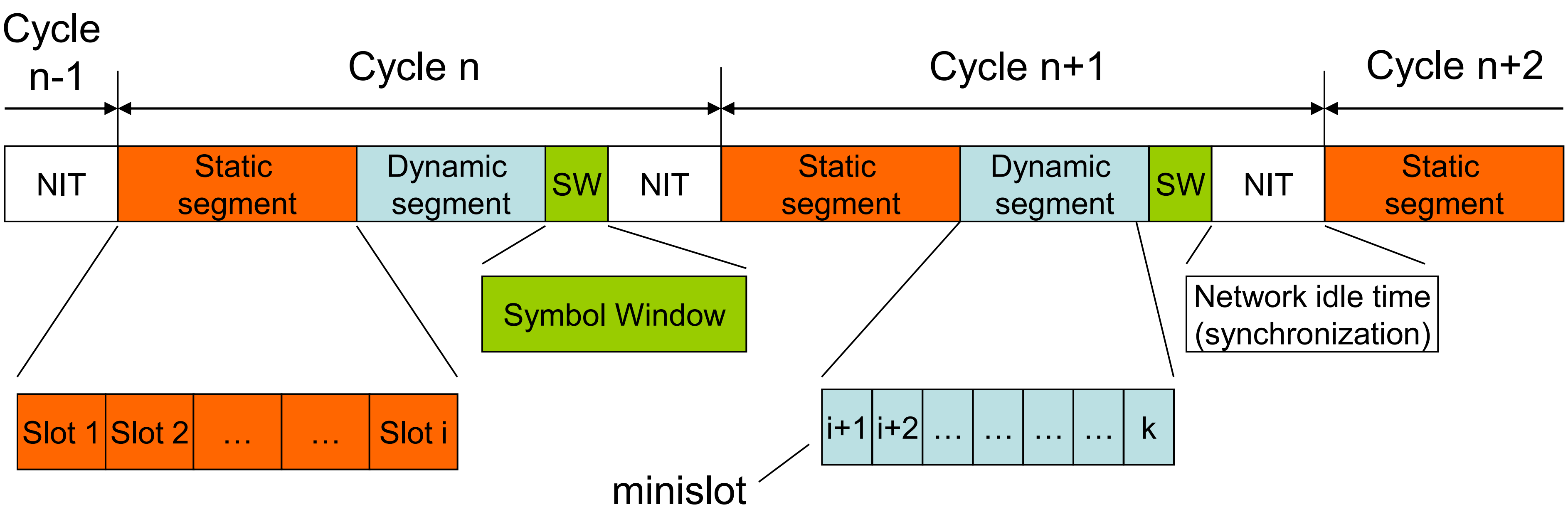
FlexRay

Overview

virtual vehicle
Vehicle EE & Software

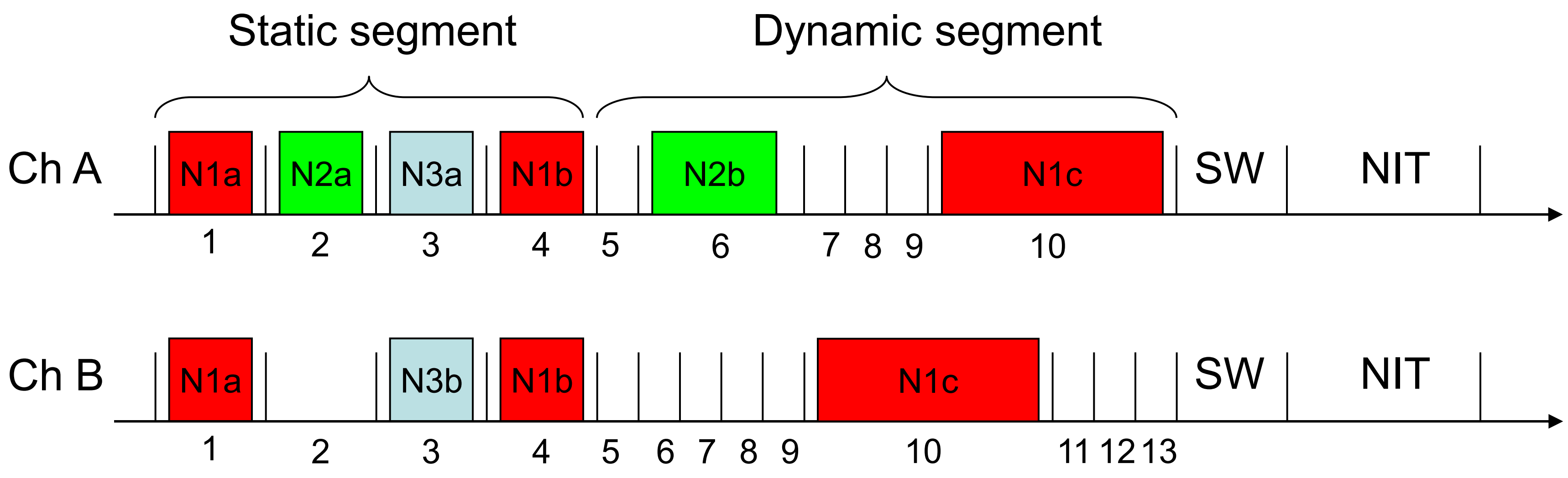
Periodical communication scheme

- Static segment for time-triggered communication
- Dynamic segment for event-triggered communication
- Symbol window for medium test
- Network idle time for resynchronization

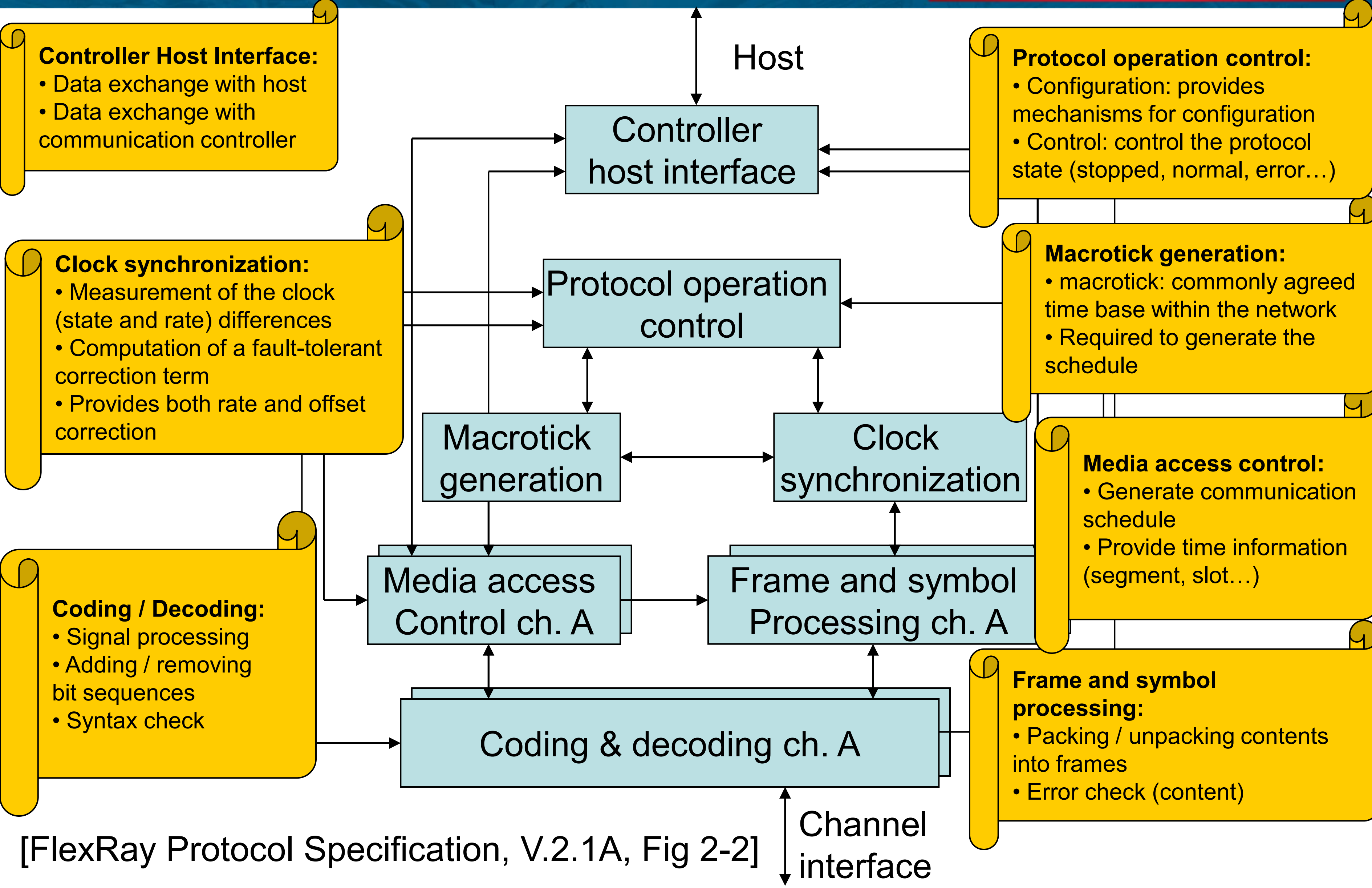


Static segment: static schedule for time-triggered communication

Dynamic segment: prioritized access for event-triggered communication

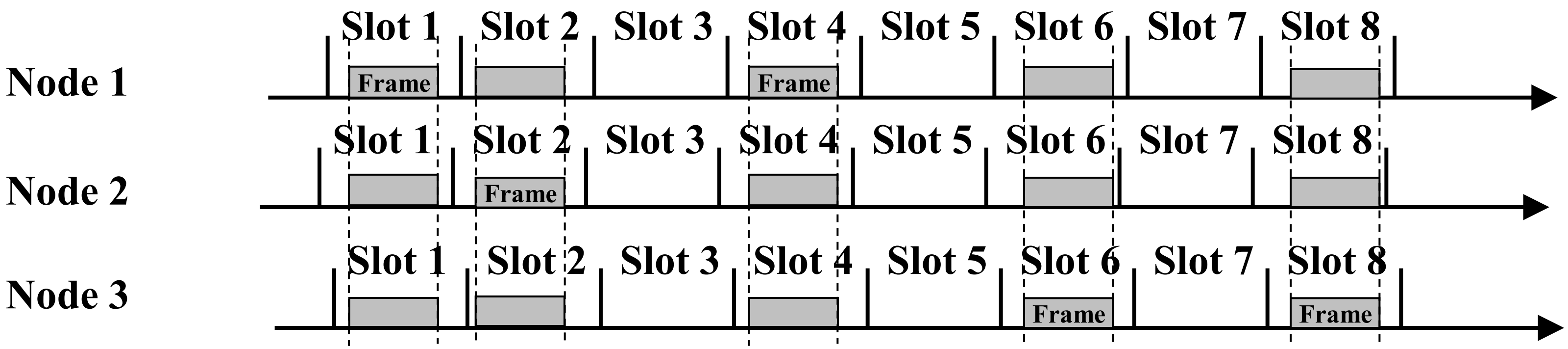


FlexRay controller overview



[FlexRay Protocol Specification, V.2.1A, Fig 2-2]

Aim: provide a global time base within the network to correct the quartz drift and avoid collision on the bus



Requirement: fault tolerant algorithm

- **No single point of error**
- **Single faults are discarded**

virtual vehicle
Vehicle EE & Software

Goal

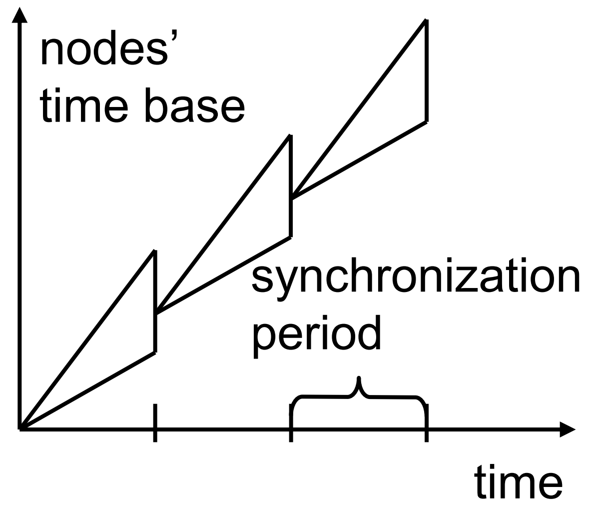
- Synchronize the macroticks between the nodes
- Keep the system precision (maximal time difference between any two nodes) bounded

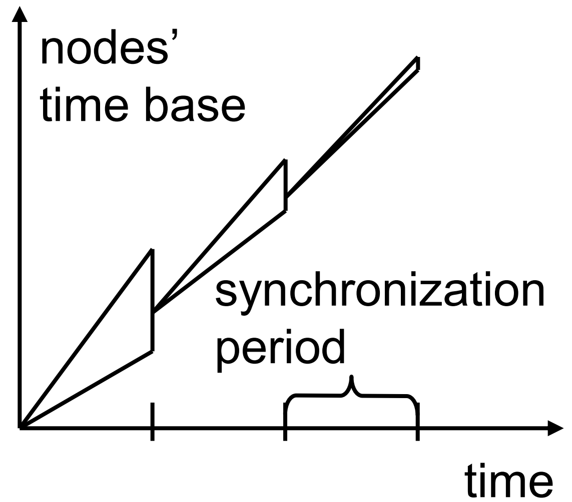
Offset correction

- Goal: minimize the clock state difference at cycle start
- Correct the number of microticks per cycle
- Discrete correction (once per cycle)

Rate correction

- Goal: minimize the clock state difference within the cycle
- Modify the number of microticks per macroticks
- Continuous correction



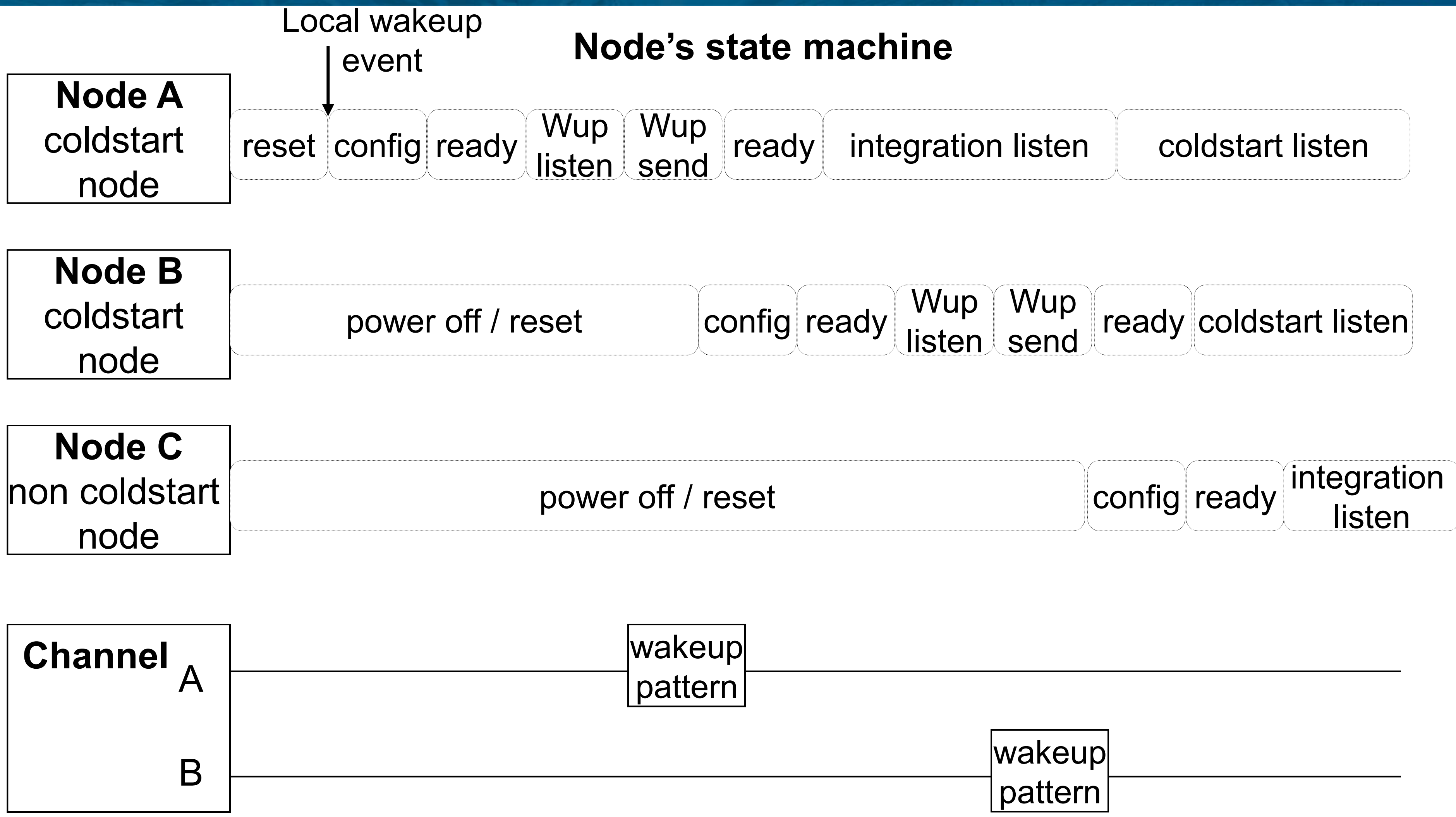


Motivation

- Wake-up the network and provide initial synchronization
- Fault tolerant (network operation relies on start-up)
- Fast operation (fault recovery)

Three phases

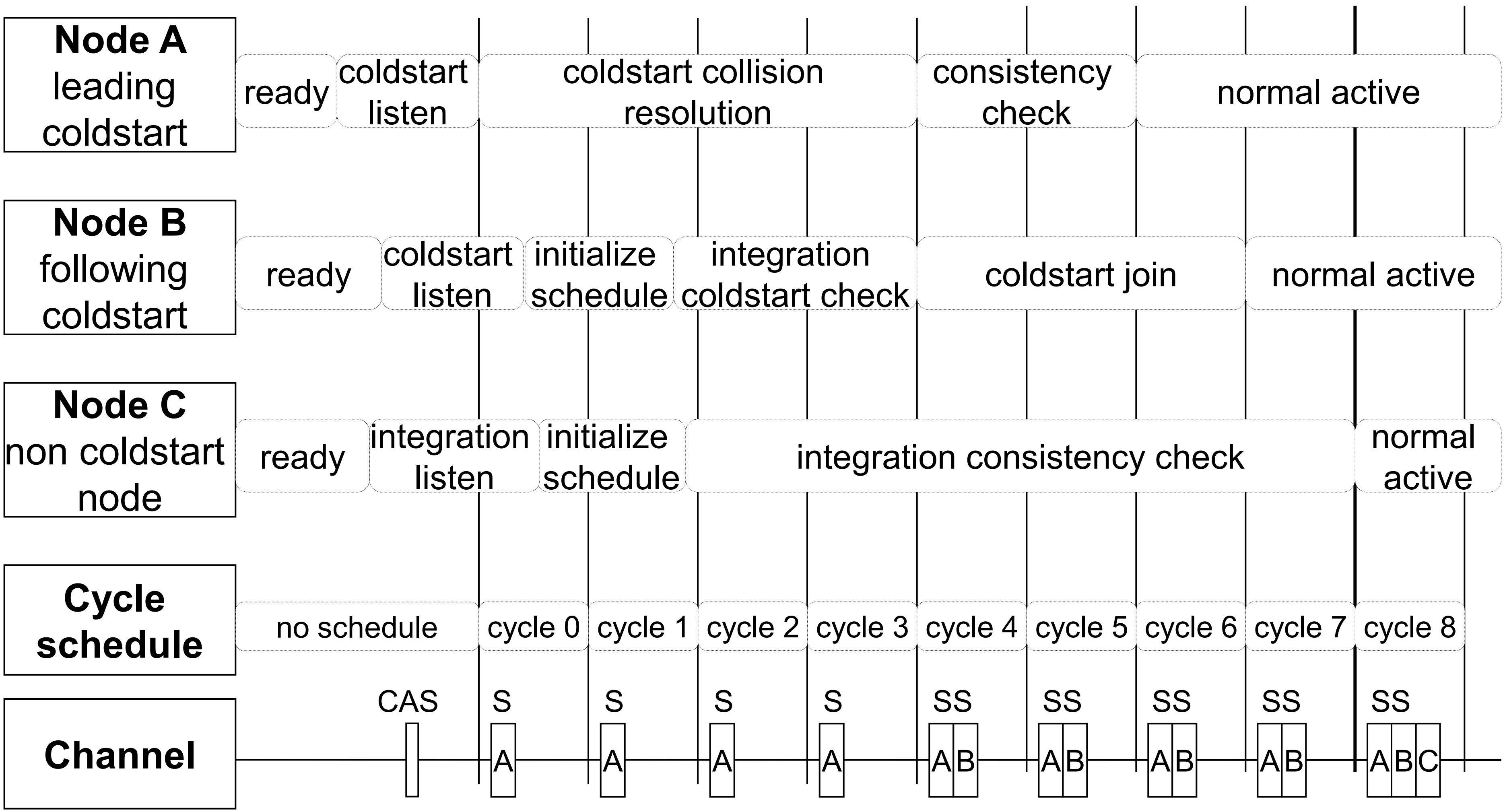
- Wakeup: to wake-up the network (active stars, nodes) if it is still asleep
- Startup: to begin communication (initialize schedule) when the nodes are awake
- Reintegration: to integrate single nodes within a running cluster



[FlexRay Protocol Specification, V.2.1A, Fig 7-6]

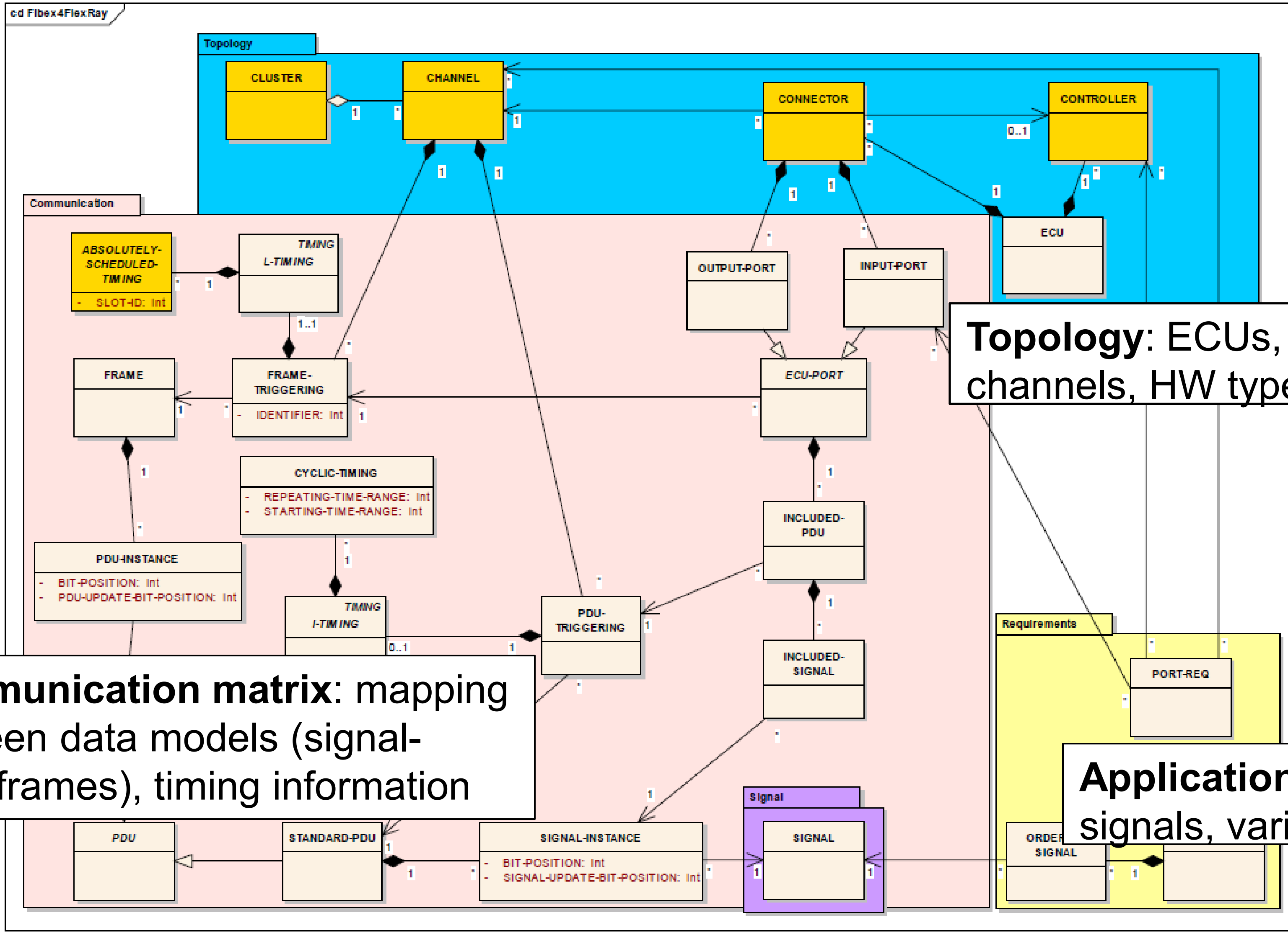
Startup – illustration

Node's state machine



[FlexRay Protocol Specification, V.2.1A, Fig 7-10]

Integration issues



Topology: ECUs, comm. channels, HW types

Communication matrix: mapping between data models (signal-PDU-frames), timing information

Application: signals, variable

[FIBEX - Field Bus Exchange Format, Version 3.0 ASAM AE, 2008, Fig 10-1]

Control flow – integration within the SW architecture

- **Event-triggered** (action triggered with rxd / txd interrupt)
  flexibility but control flow difficult to handle (interrupts)
- **Time-triggered** (comm. task synchronous to bus schedule)
  a-priori known behavior (time domain) but complex dependencies between operating system and communication system

Data flow – transmission scheme

- **Buffer**: frame filtering (ID, cycle) performed in hardware,
  Time-triggered comm.: **latest data stored** (old version discarded)
- **FIFO**: frames stored sequentially, further processing in software
  Event-triggered comm.: **all frames are available**

System configuration – amount of data

- **Protocol configuration**: FlexRay schedule / syntax (>70 param.)
- **Data access and interpretation**: buffer configuration, mapping between frame and signals (>50 parameters)

Interaction between Operating and Communication system

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="2">Operating system</th></tr>
<tr><th>Event-triggered
(interrupts driven)</th><th>Time-triggered
(schedule)</th></tr>
<tr><td rowspan="3">Communication system</td><td>Event-triggered
(e.g. CAN)</td><td>➔ Priority based communication
+ Flexibility, average response time
- Complex timing analysis,
- No constructive integration</td><td>➔ Static communication scheme supported by the application
+ Easy timing analysis
- Application overhead (e.g. for synchronization)</td></tr>
<tr><td rowspan="2">Time-triggered
(e.g. FlexRay)</td><td rowspan="2">➔ static comm. scheme with interrupt based data interface
+ constructive integration (comm. point of view)
- Complex end-to-end timing analysis
- No constructive integration (node’s point of view)</td><td>➔ Asynchronous systems
+ Easy timing analysis
- Non optimal end-to-end delays (synchronization)
- Frames might be missed</td></tr>
<tr><td>➔ Synchronous systems
+ Easy timing analysis
+ deterministic and optimal end-to-end delays
- Flexibility</td></tr>
</table>

Cars are forming complex distributed systems, evolving in harsh environments; in parallel their reliability requirements increase

Automotive embedded systems from two perspectives

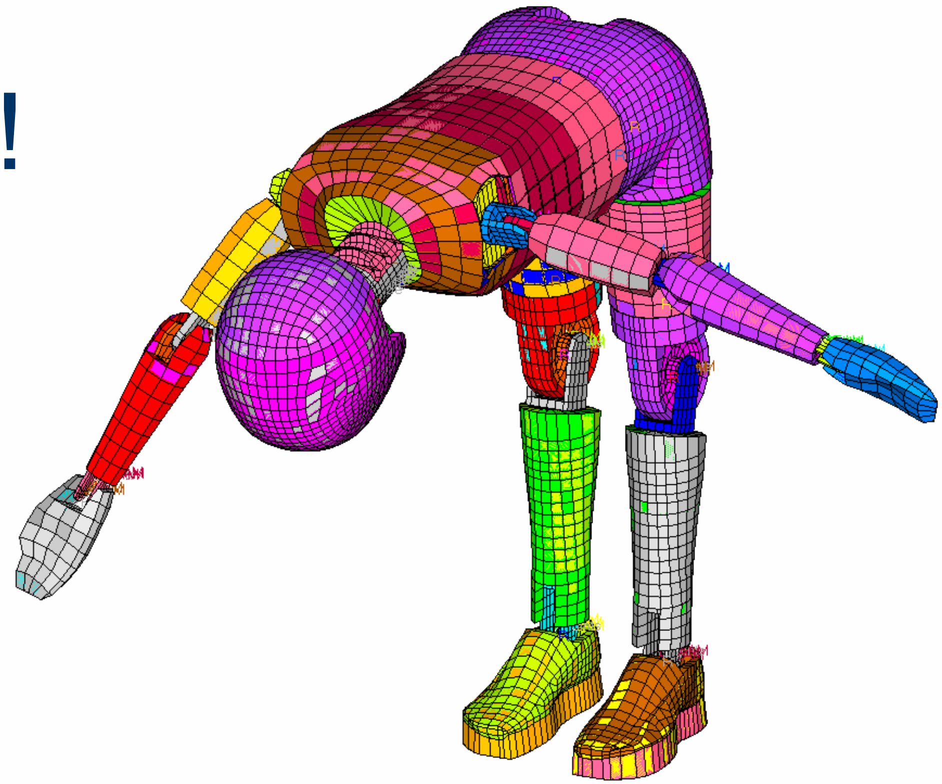
- Software engineering (requirement engineering, model-based development and functional safety)
- System architecture (event- vs. time-triggered)

The question is not anymore **“how can I develop a given function”** but **“how can I make my system more dependable for lower costs”**

- Meta-information for the description of the product are important
- Traceability between the system views required
- Traceability of the development process required
- (model-based) tool-chain as central elements to achieve these goals

Do not forget Verification and Validation activities!

Thank you
for your attention!



www.v2c2.at

K2 / K plus Competence Center - Initiated by the Federal Ministry of Transport, Innovation and Technology (BMVIT). Funded by FFG, Land Steiermark and Steirische Wirtschaftsförderung (SFG)