Scheduling Analysis in Practice: Early Lessons Learned

Dr. Kai Richter

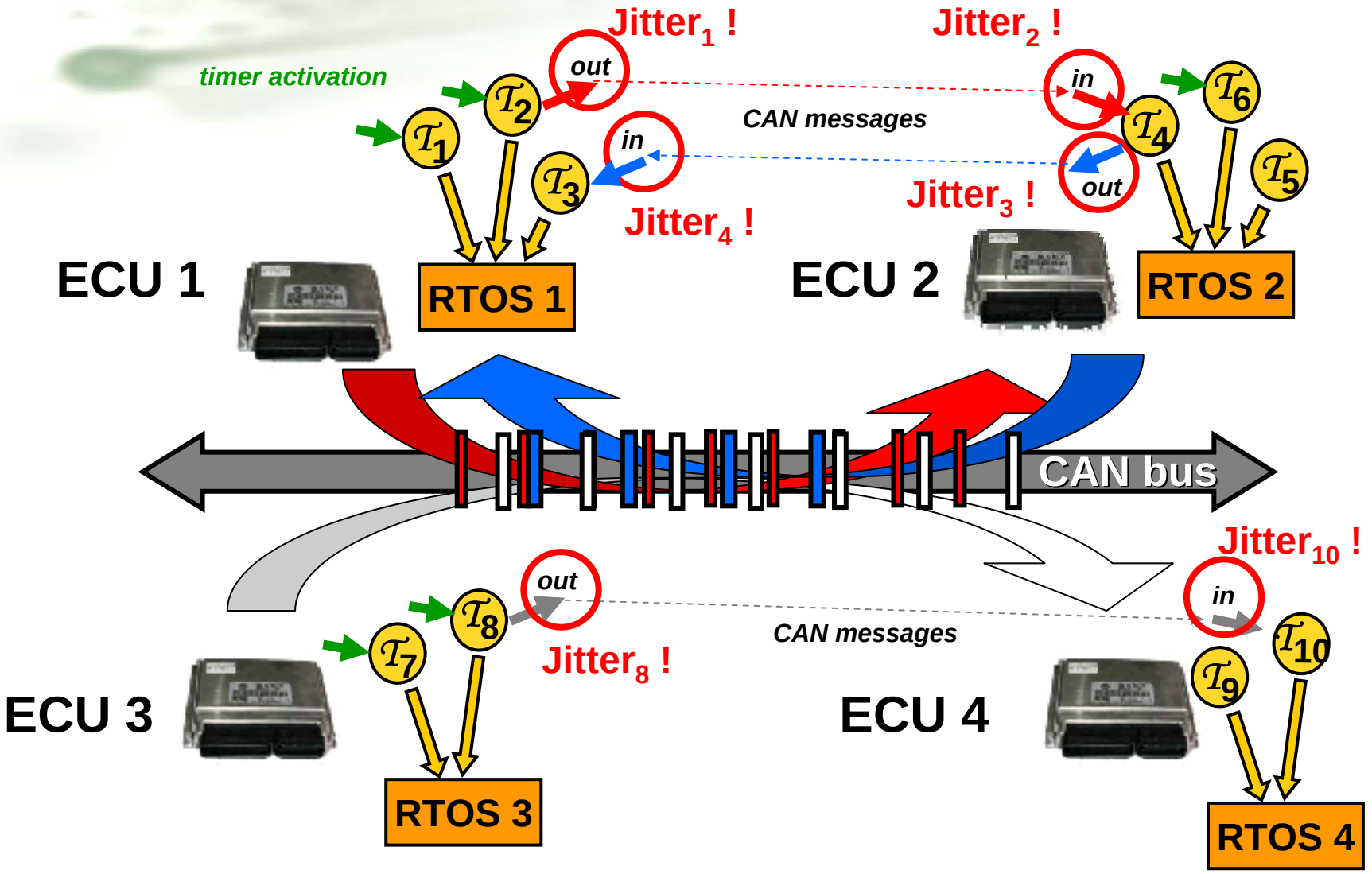
CTO @ Symtavision GmbH,
Braunschweig,
Germany

SymTAVision

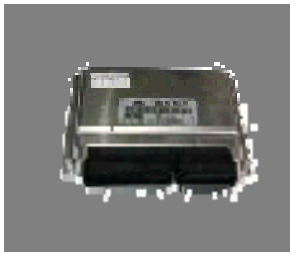
SymTAVision Background

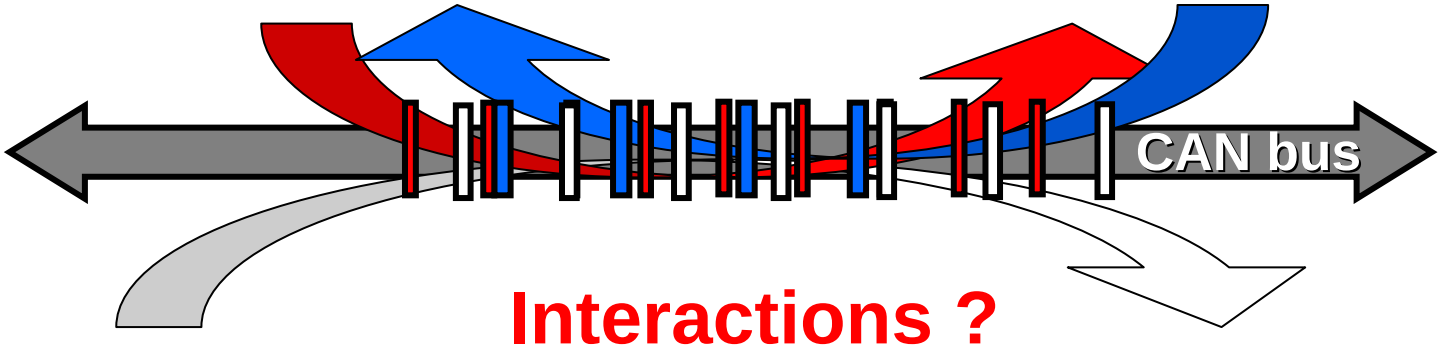
- Institute of Computer and Network Engineering,
  TU Braunschweig, Germany; EE department
- Roots in circuit design (HW), “software-less”
- Prof. Rolf Ernst: Design Automation of Embedded Systems
  - HW/SW Co-Design (COSYMA)
  - System-level applications modeling: SPI (1998-2004)
  - System-level architecture analysis: SymTA/S (2002-current)
  - SymTA/S group currently 5-7 PhD student + 10 in other groups
  - SymTA/S Tool Suite (will participate in comparison Wed/Thu)
- May 2005: spin-off SymTAVision
- Our mission: “Go out and help the people!“

Typical Automotive System



Black-Box System Integration





Interactions ?
Requirements ?
Jitters ? Deadlines ?
Flexibility/Tolerance?

Outline

- Complexity of real-world OS and protocols
- (Mis-)use of APIs in user code
- The fear of the “worst case”
- Data (un-)availability
- Solving the real problems
- Further issues
- Conclusion

Complexity of real-world OS and protocols

- **RMS (rate monotonic scheduling):**
  - **Unique priorities**
  - **Fully preemptive**
  - **Periodic activation**
  - **No recurrence (D=T)**

$$w_i(q) = q\,C_i + \sum_{j \in \mathrm{hp}(i)} C_j \left\lceil \frac{w_i(q)}{T_j} \right\rceil$$

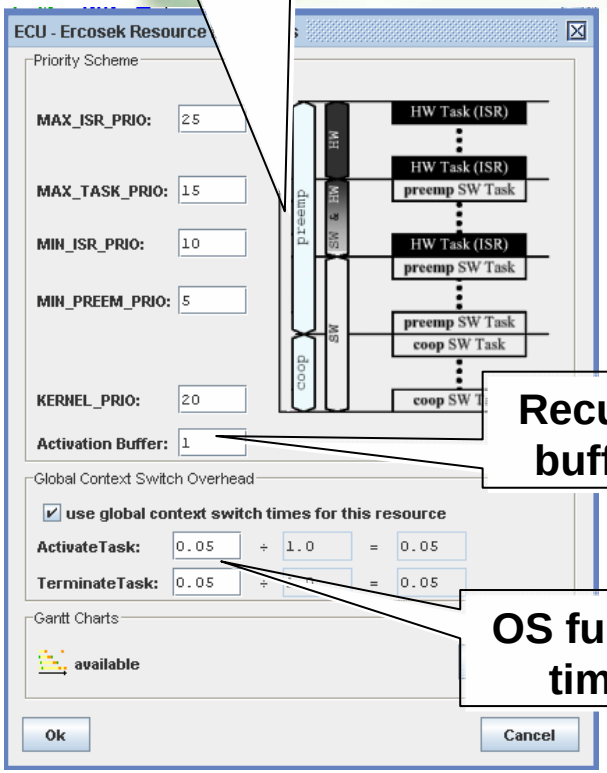
$$R_i(q) = w_i(q) - (q-1)\,T_i$$

  - **extensions support recurrence, burst activation, blocking, etc.**
- **ERCOSek (OS @ automotive supplier):**

  - **Shared priorities**
  - **Mixed preemptive and non-preemptive tasks + shared resources**
  - **HW Tasks (interrupts) above kernel priority**
  - **TimeTable (incl. Offset), static and dynamic Alarms (OSEK)**
  - **Supports recurrence**
  - **OS overhead: Dispatcher and key OS-routines at varying priorities**
  - **“Basic software” with high- and low-priority drivers**

ERCOSek Example: OS Config

Complex priority scheme

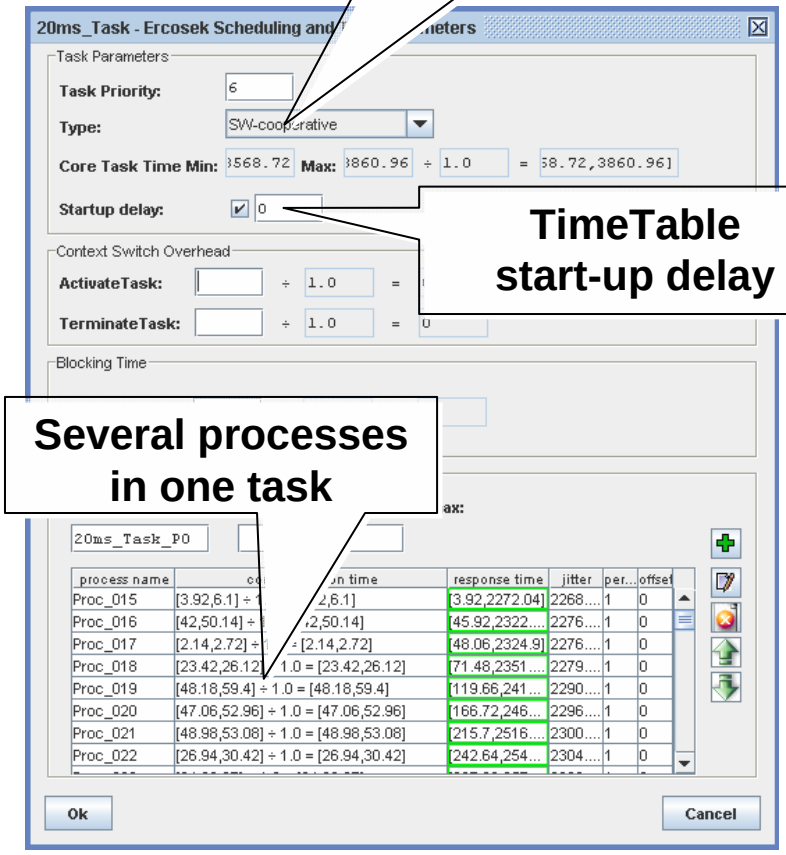


Recurrence buffer size

OS function timing

OS setup

Selection: preemptive / non-preemptive



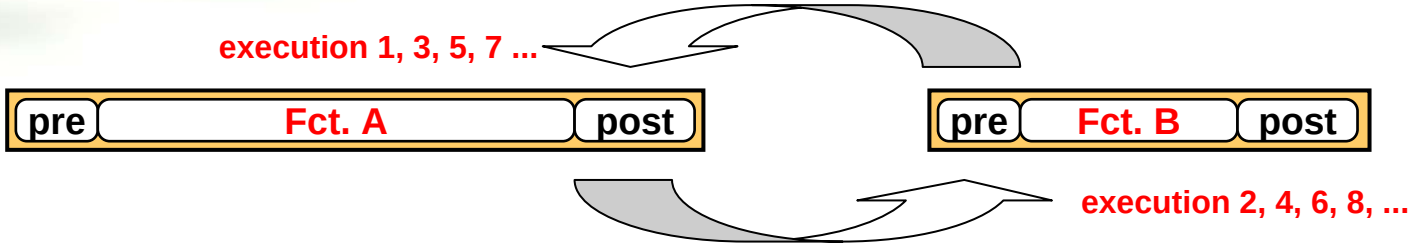
TimeTable start-up delay

Several processes in one task

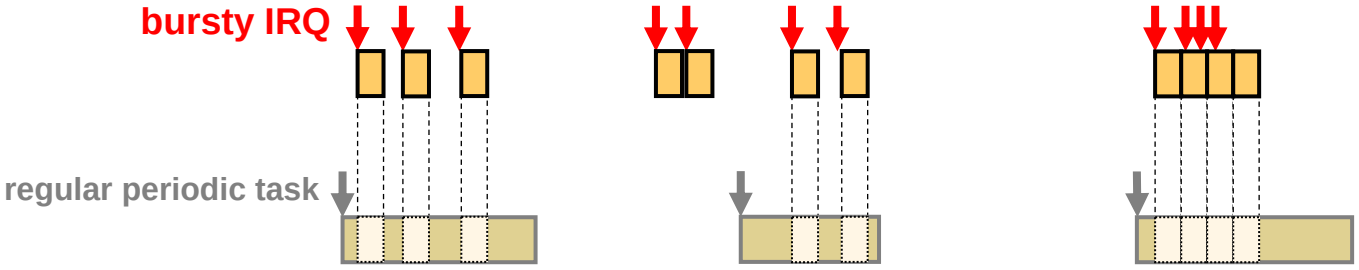
task setup

ERCOSek Example: Timing Behavior

- **Cyclic alternation of functions**



- **Activation in bursts**

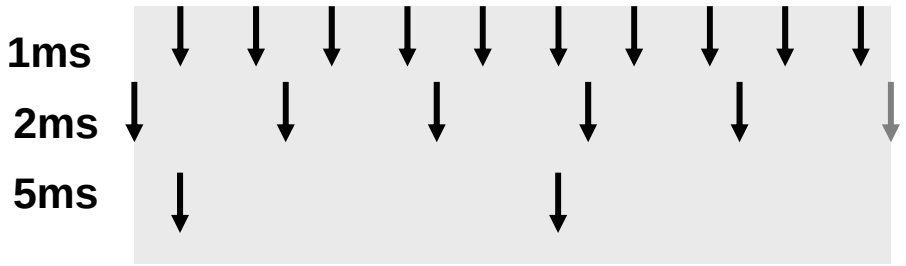


(Mis-)use of APIs in user code

- OS concepts accessible through APIs, e.g.
  - TimeTable
  - Alarms (time-out interrupts)
  - drivers
- The way APIs are used has huge influence on scheduling
  - Needs further adaptation of analysis models
  - Use of APIs often not systematic
  - AND: not easy to find out
- There is no single analytical model for an OS or bus, the individual design must be taken into account!

ERCOSek Example: Task Activation

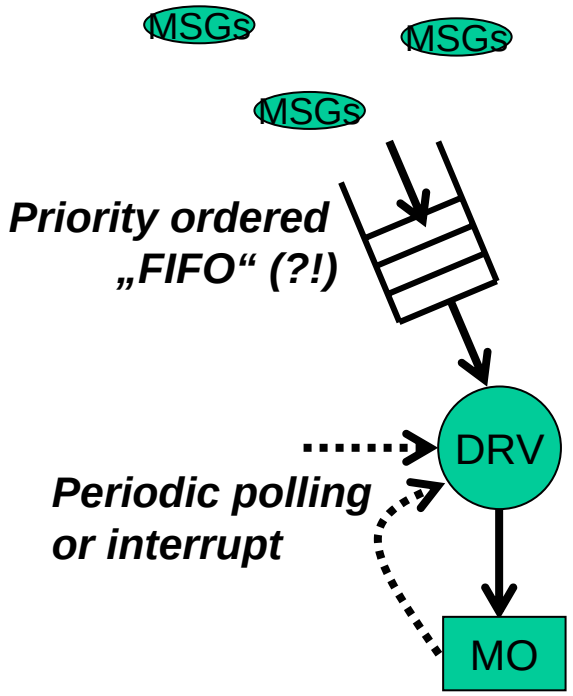
- **Engine ECU:**
  - **1ms, 2ms, 5ms tasks activation by TimeTable**



  - **Chained activation of 10ms,20ms,50ms,100ms… tasks**
    - **“pseudo” TimeTable in user code**
    - **by dynamic Alarm(), depends on RPM value**
  - **Further RPM-dependent external interrupts**
    - **in phase with crank shaft**

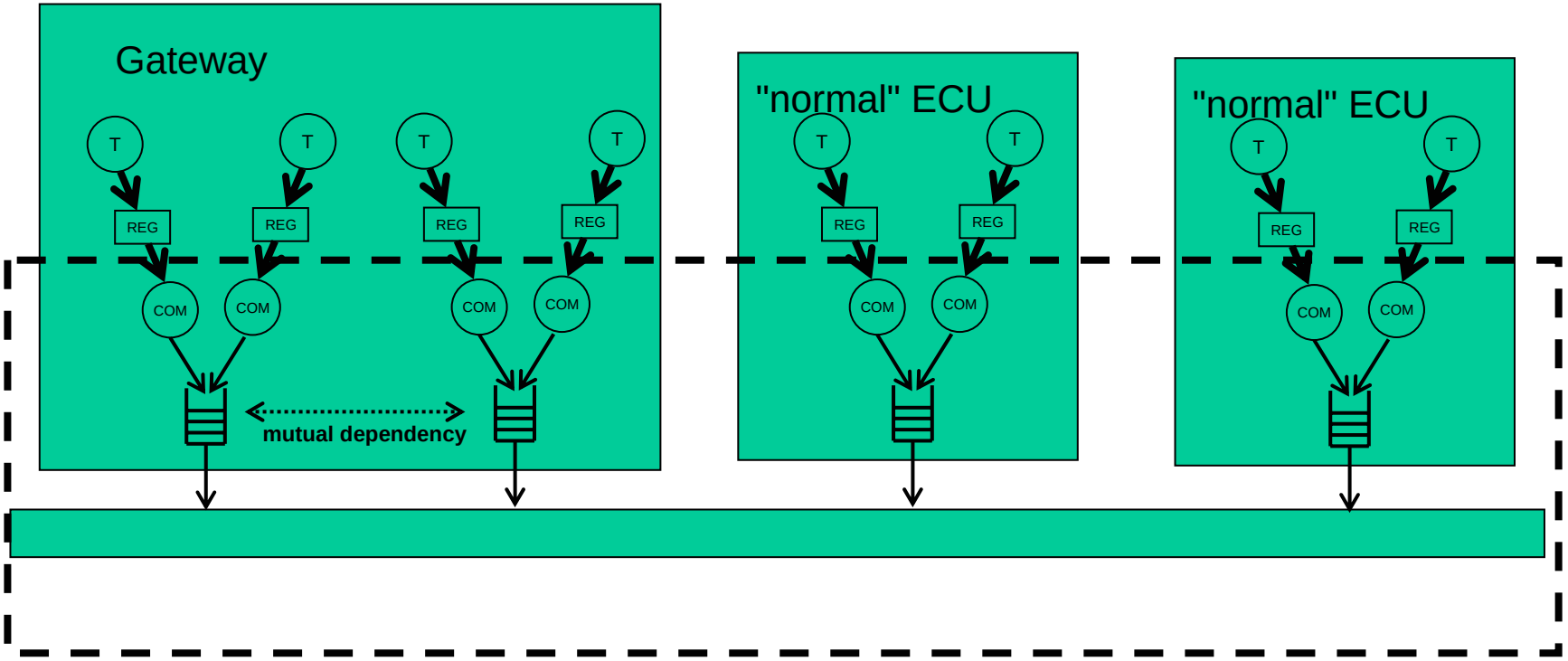
CAN Example: Driver Structure

- Messages are „said“ to be periodic because they are generated by periodically activated tasks
- Scheduling and driver structure distorts this periodicity
- Interface implementation leads to complex blocking
- mutual dependencies between local blocking and global schedule



CAN Example: Analysis Complexity

- **Individual send buffers → mutual dependencies between local blocking and global schedule**



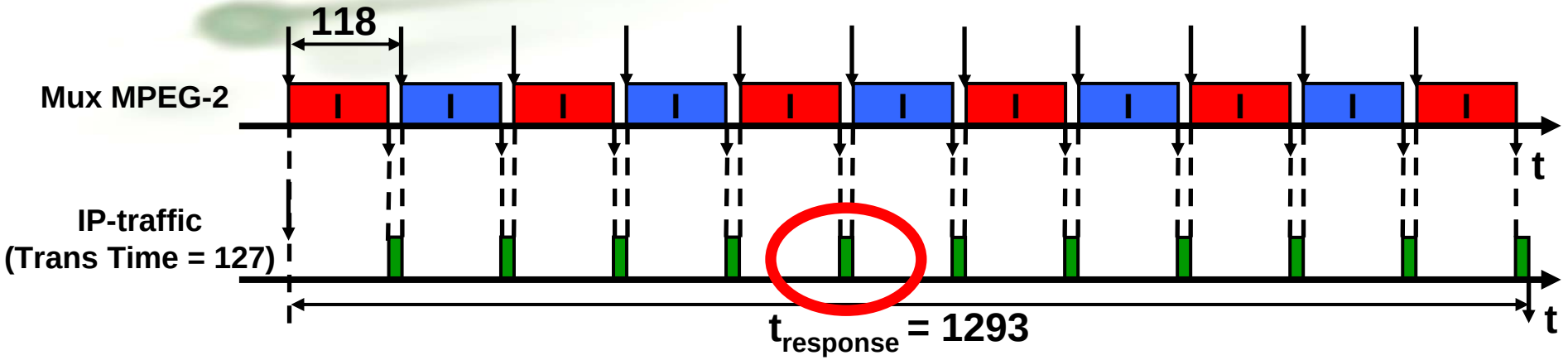
SCOPE of CAN Analysis ?

The fear of the “worst case”

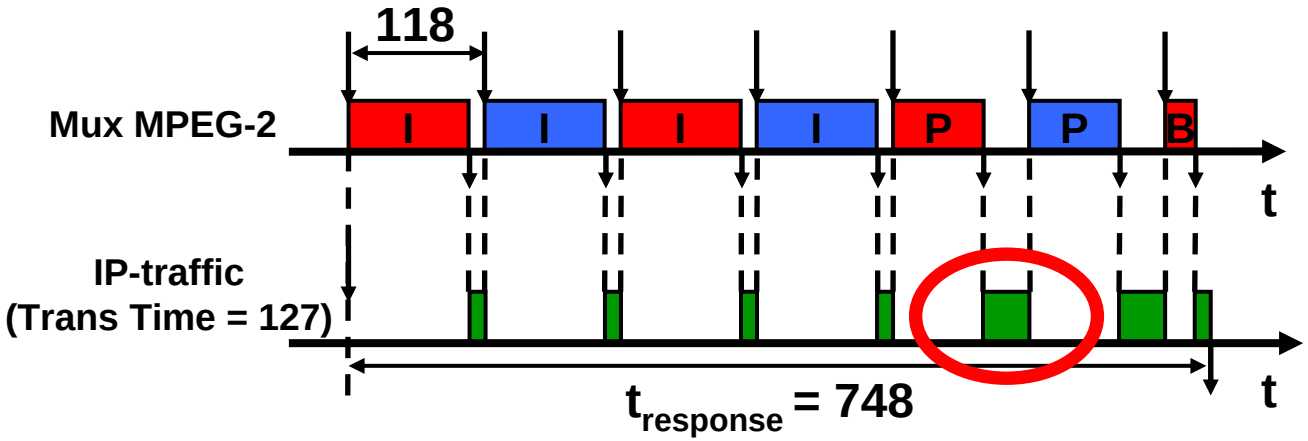
- Designers do not like that term because
  - They associate inaccuracy, conservativeness with it
  - They argue that the “worst case” will never occur …
  - … and they refer to “the product be up and running already without any problems”
  - this can be a “killer”
- Analysis must often compete with simulation/test accuracy
- KEY: Information about mutually exclusive effects
  - Blocking
  - Context dependent behavior
  - Scenarios (especially in changing environments)

Example: Intra Context

- **Bus scheduling analysis *without* Intra context**

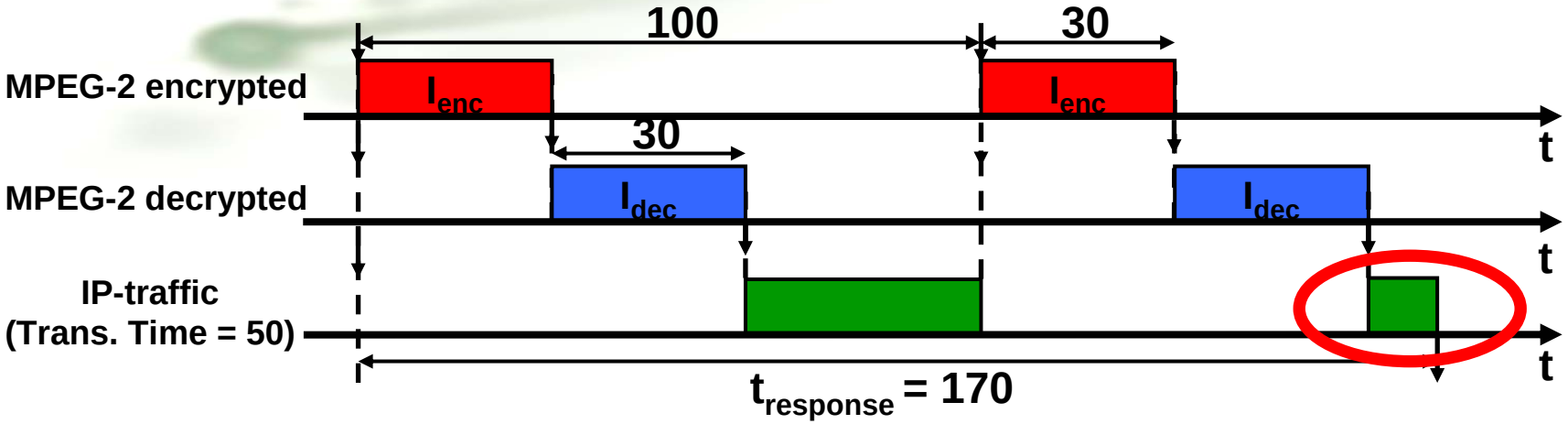


- **Bus scheduling analysis *with* Intra context**

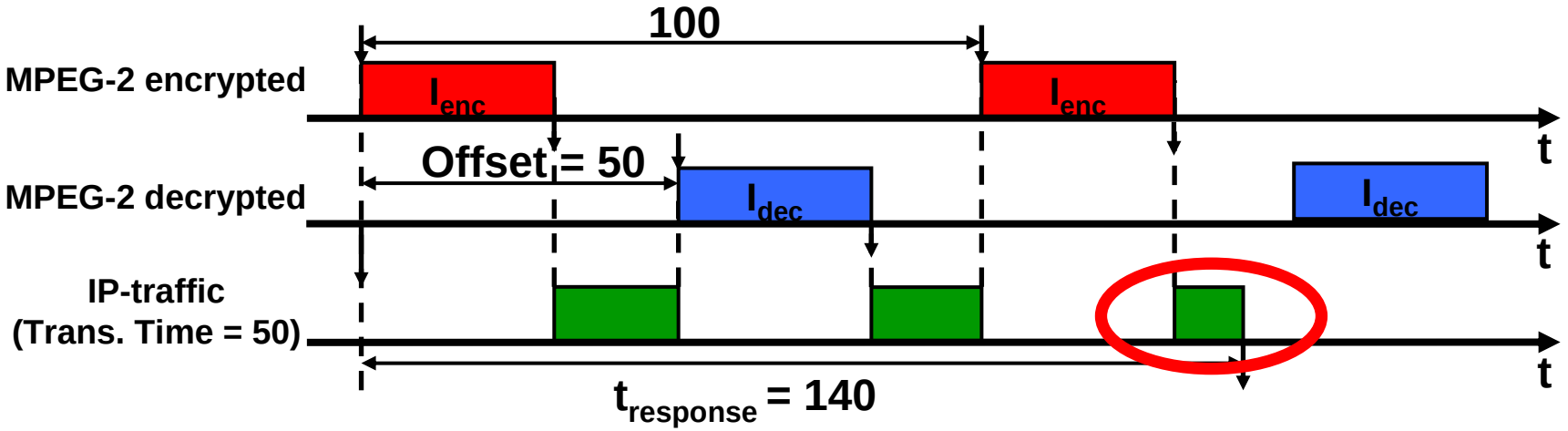


Example: Inter Context

- Bus analysis *without* Inter context



- Bus analysis *with* Inter context



Data (Un-)availability

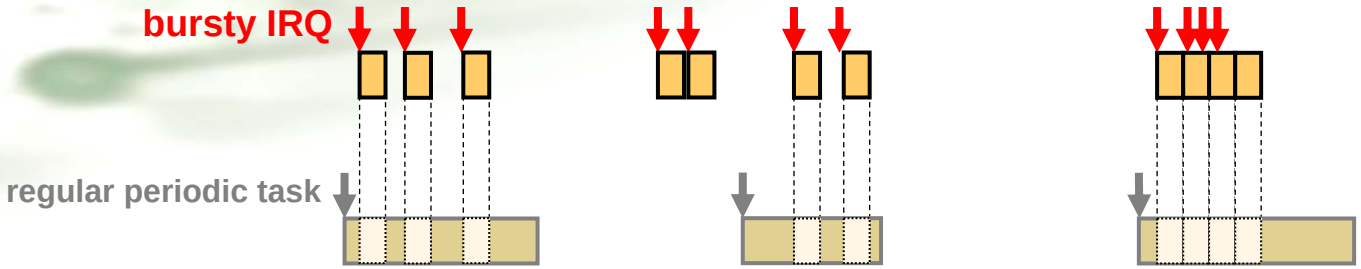
- Analysis needs data
- It can be hard to gather information
  - Find the right people
  - Use their language
  - Do not assume they know much about analysis
  - Transform the information into „something useful“ for analysis
- Examples:
  - Overall structure
  - Use of concepts
  - Dynamic behavior of interrupts, messages, mutexes, etc…
  - Execution times

Example: Core Execution Time Analysis

- Critical for scheduling analysis
- Host of designers are not familiar with
  - Formal execution time analysis
  - Simulation
  - Instrumentation
  - Tracing tools

- KEY: Help them to produce that data!

Example: Timing of External Interrupts



- External interrupts are often “bursty”
- Measured traces can be transformed into “models”
  - Requires reasonable interpretation
  - Tailored to the specific problem at hand

- KEY: Help them to produce that data!

Solving The Real Problems

- Analysis produces „numbers“
  - Response times, load, buffer sizes, …
  - Can be compared against constraints
- BUT: Plain verification is the rare case !!!
- More often, designers have „vague“ goals
  - Increase productivity
  - Detect hot spots / bottlenecks
  - Forecast future extensibility

KEY: Find out what they want and what they need and discuss!

Approaching Problems

- Examples
  - ECU people want to avoid overloaded CPU but have no information about external interrupt frequencies
  - Bus people want to know “stability” and criticality of messages but have no information about dynamic behavior (jitter)
  - Network people want to increase the throughput of their gateways but do not know how much data is going through
- Approach
  - Identify the results needed to make people happy
  - Identify the data you have and you can obtain
  - Discuss assumptions on data that’s missing
  - Discuss potential analysis and expected results
  - Consider “What-if” experiments as an advantage of analysis

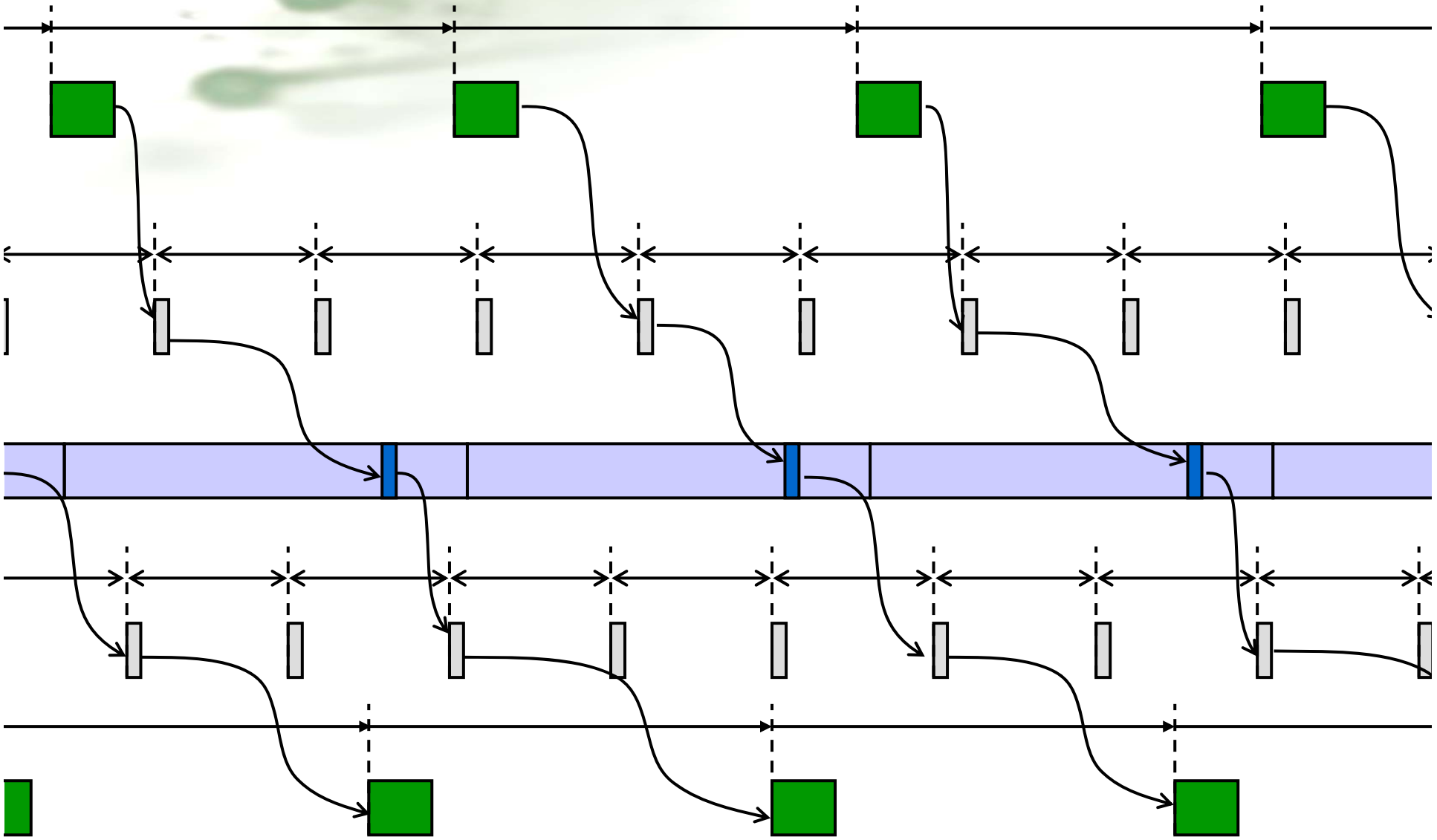
Further Issues

- Comprehensibility and usability of methods/models
  - Designers often reject to “learn” new models nor tools, unless reasonably easy to use
  - At best, the analysis fits fully into the existing tools
- Integration and Supply chain considerations
  - Business processes complicate matters further
  - Black boxes increase the data (un-)availability problem
- Real Benefit
  - Scheduling analysis competes with simulation/test
  - Establishing new technology essentially requires HUGE benefits for the design process

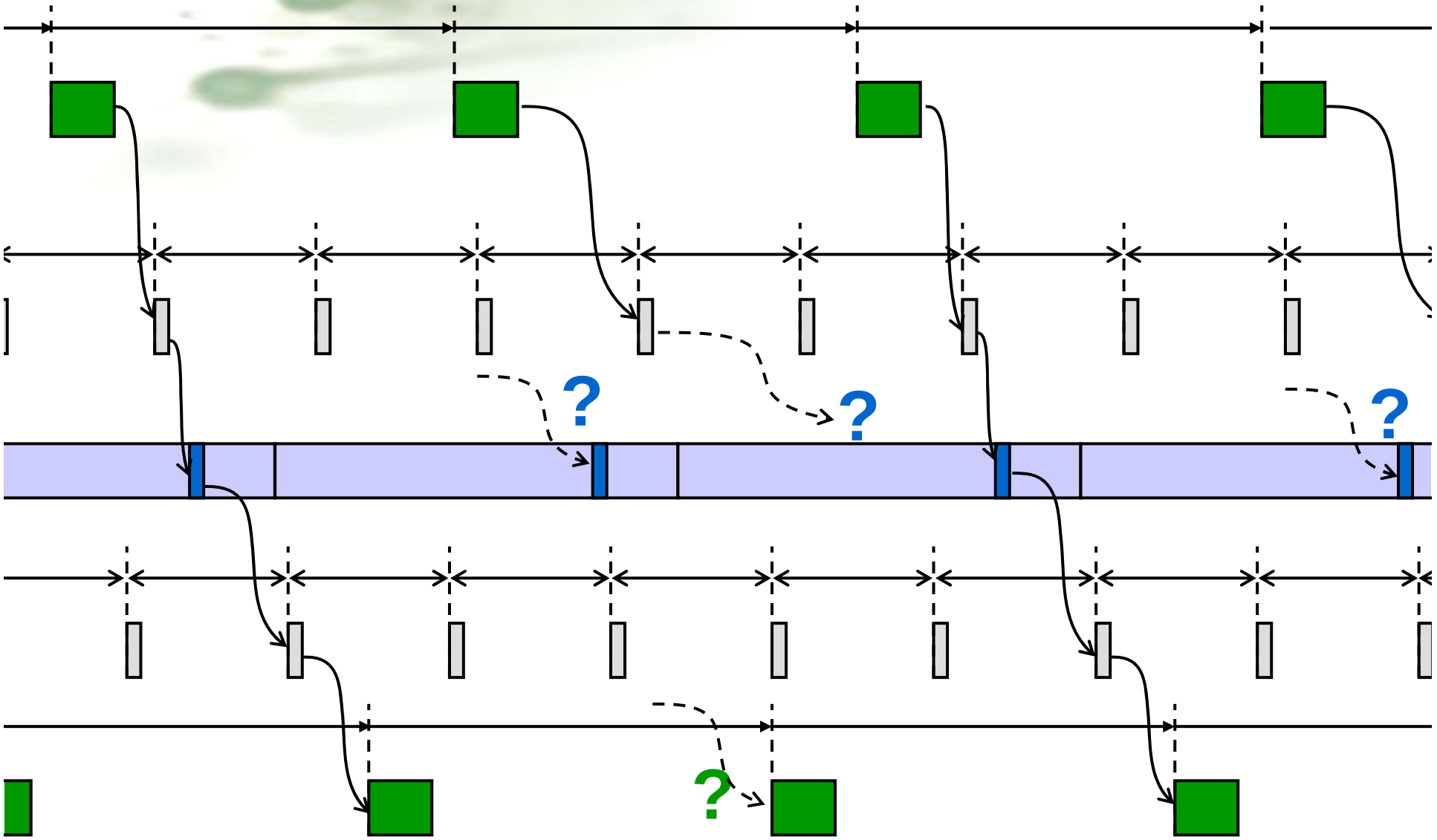
System-Level in The Future

- CAN and Priority-driven ECU scheduling
  - Loosely coupled subsystems
  - Supports dynamic environments
- FlexRay and global synchronization
  - Tighly coupled
  - Short end-to-end delays
- Reality: heterogeneous mixture
  - CAN segments for less „critical“ applications
  - Fully synchronous for safety critical
  - Mixed for

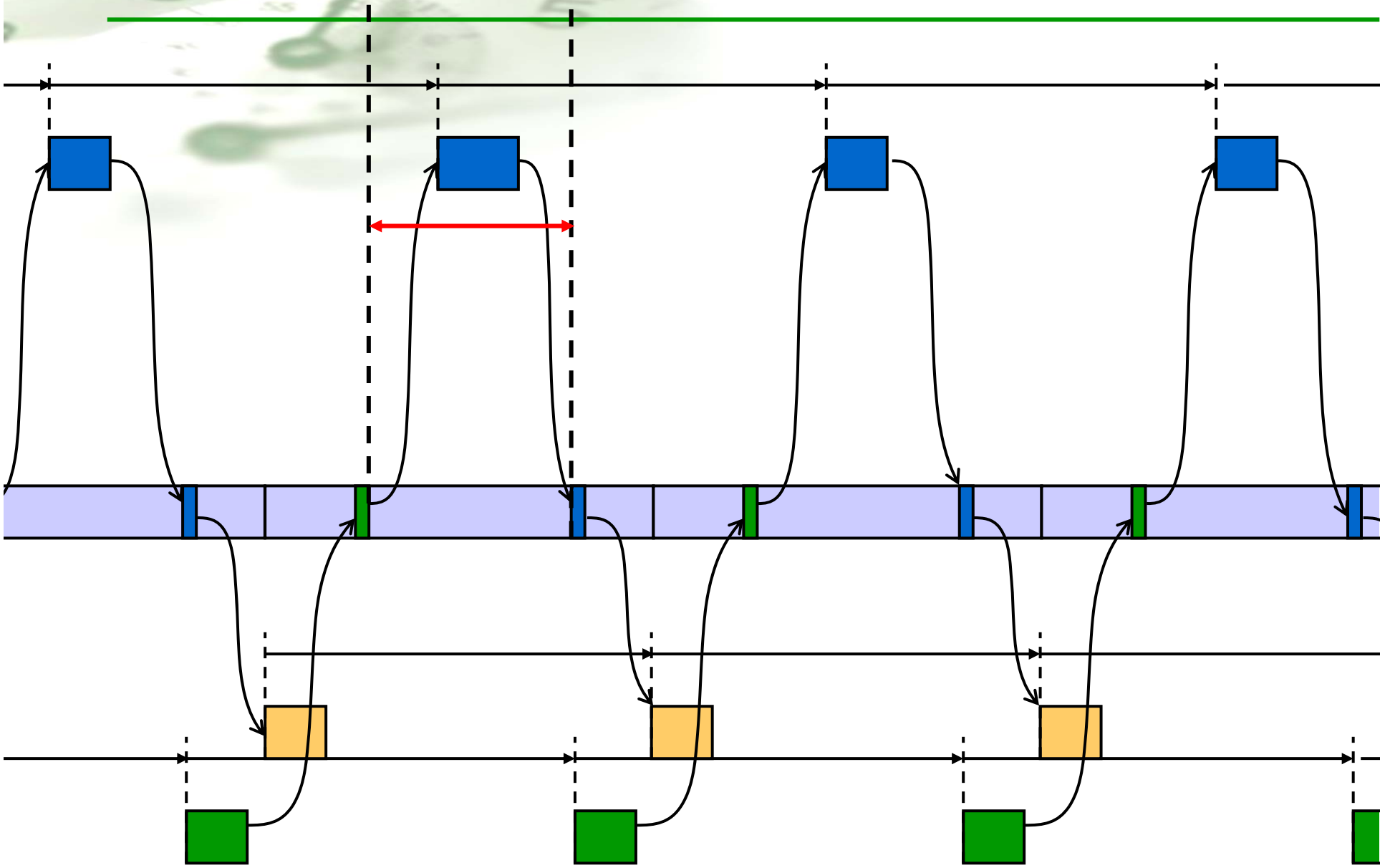
Case 1: FlexRay (cyclic send/rec.), OK!



Case 1: FlexRay (cyclic send/rec.), irregular!



FlexRay synchronization requires tight and inflexible constraints.

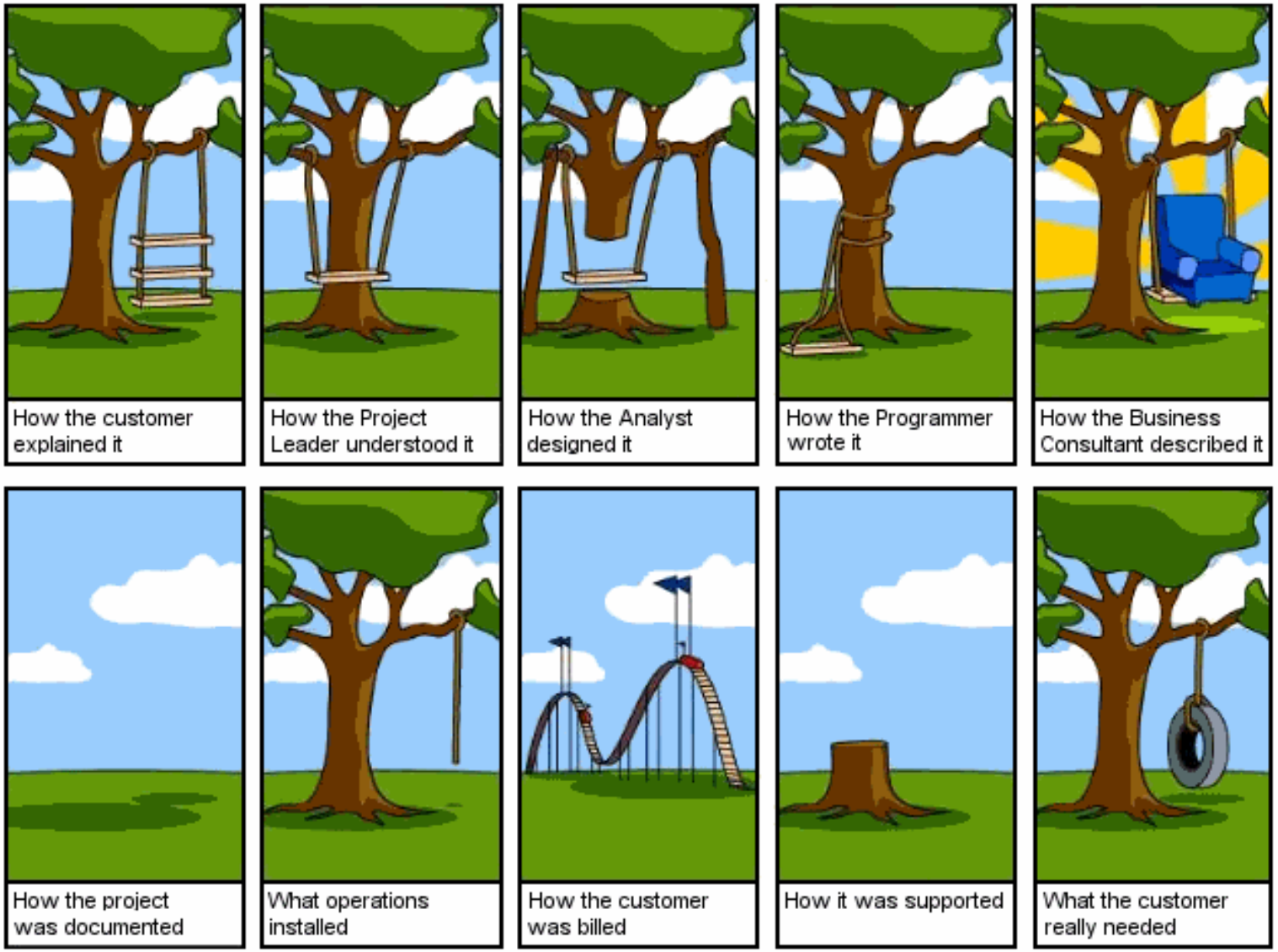


Conclusion

- **Scheduling analysis is a vehicle for “solving critical design problems”, but must**
  - **React to the problems at hand (not vice versa)**
  - **Accept the an imperfect world**
  - **Recognize established processes**
  - **Solve problems and provide new benefits**
  - **NOT pretend to know it all better**

KEY LESSON LEARNED:
Bringing scheduling analysis (and other analysis) to the people requires knowing well their problems and “desires”, and finding the right solution for them.

It is very unlikely that they change their practice.


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed