Timing Analysis of Automotive Architectures and Software

Nicolas Navet
University of Luxembourg, founder RealTime-at-Work
Focus is not the formalisms but on what to expect from timing analysis

- Timing analysis in the automotive domain
- Beware of formal models
- Can timing-accurate simulation provide required guarantees?
Software has become the key to innovation

- Software grows exponentially
- Complex new technologies are introduced
- Pace of innovation ↑

How (formal) timing analysis can keep up?

Software is disrupting complete industries

Every company has to learn to become a software company

Model-Driven Development is certainly a powerful enabler but ..

Still lacks
- Timing-augmented design flow
- Timing equivalent execution between model and run-time
- Automation features: “state the what, not the how” + “correct by construct”
Hundreds of timing constraints

Figure from [17]

- Responsiveness
- Freshness of data
- Jitters
- Synchronicity
- ...

Involves hardware, software, networks, gateways, runtime environment (OS, Middleware, hypervisors) Multi-source SW and HW
What makes things hard in automotive

Technologies: numerous, complex and not conceived with verifiability as a requirement

- # of networks, complexity of Autosar (>150 doc) with limited support for timing specification, multi-core ECUs, GPU computing for ADAS, etc

- # of functional domains, buses, gateways, ECUs, size of code, tasks, wiring, number of variants, etc

Development process

- Limited regulatory constraints
- No “culture” of verification
- Traceability of timing constraints!
- Time, costs & resource utilization constraints
- Most developments are not done in-house
- Carry-over / Vehicle Family Management
Verification along the dev. cycle

Simulation

- Functional simulation
- Timing-accurate simulation of ECU, bus, system-level
- Hardware in the loop, software-in-the-loop, processor in-the-loop, etc

Formal verification

- Worst-Case Execution Time analysis
- Worst-Case Response time analysis: ECU, bus, system-level
- Probabilistic analysis (academia)

Testing

- Integration tests
- Execution time measurements
- Off-line trace analysis
- Smart monitoring tools

“Early stage”

Technological & design choices

“Project”

Configuration & optimization

“Real”

Refine and validate models & impact of non-conformance

16-Mar-16
Nicolas Navet / University of Luxembourg
Zoom on Worst-Case Response Times

Simulation

Formal verification

Testing

WCRT: formalisms mature enough to derive usable bounds ... if system complexity is “reasonable”

✓ Worst-Case Response time analysis: ECU, networks, system-level
The actual true worst-case is between the two curves.

Average difference is 4.7% (up to 35%) – WCTT are accurate here because modeled system is simple and easily amenable to analysis.
Zoom on Worst-Case Response Times

Accurate model \(\rightarrow\) verification
Approximate model \(\rightarrow\) debugging, but usually unpredictably unsafe for verification

\[
K_t^*(t) \triangleq \left| \frac{t + \varphi_t(t)}{T} \right| + \left| \frac{t - \varphi_t(t)}{T} \right| + 1
\]

- **Worst-Case Response time analysis:** ECU, networks, system-level

Requires knowledge of
- All activities: tasks, runnables, frames, signals
- Software code to derive execution times
- Complete embedded architecture with all scheduling & configuration parameters for buses and ECUs

Conservative assumptions possible with high resource utilization in automotive ?!

Nicolas Navet / University of Luxembourg
Models cannot replace testing

Ex: CAN communication traces

Priority inversion here because frames are not queued in the order of priority

Check comm. stack implementation, periods, offsets, jitters, model for aperiodic traffic and transmission errors, clock drifts, etc..
Question: How do we know (formal) timing analysis models are trustworthy ?!
What do we have at hand

- Are the models published? **Usually no**
- Is the source code of the tool available? **No**
- Do we have qualification? **No**
- Are there public benchmarks on which validate the results? **No**
- Limited number of end-users and cost-pressure? **Yes**
- Complexity of the models and implementations? **High**
- Can we prove the correctness of the analysis results? **Not yet** – step in that direction [2] for Network-Calculus analyses

**Good practice - several techniques and several tools for cross-validation**
Examples of cross-validations

✓ Comparing simulation and analysis results
✓ Validating a simulator using real communication/execution traces: e.g., comparing inter-arrival times distributions
✓ Re-simulating worst-case situation from schedulability analysis
✓ Validating schedulability analysis against lower-bounds: e.g., validating Network-Calculus AFDX analysis with unfavorable scenarios from [3]
✓ Cross-validating schedulability analysis by comparing different formalisms / tools: e.g. network-calculus VS event-streams VS trajectory approach
✓ …

Validating timing accurate simulation models is much easier than schedulability analysis tools
Complex analytic models is a dead-end

Ex: Towards realistic Controller Area Network Analyses

- Non-prioritized waiting queues
- Non-abortable transmission requests
- Not enough transmission buffers
- Delays in refilling the buffers
- Delay data production / transmission request
- Segmented messages
- Autosar mixed-transmission requests
- Aperiodic traffic
- Transmission errors
- Gatewayed traffic
- ...

Subset of the 50+ papers [14]

- Not everything covered, no complete integration
- Many analyses too pessimistic to be usable
- Precise analyses are often intractable and error prone

If formal analysis is needed, systems must be conceived accordingly
Timing-accurate simulation of embedded architectures

- Today: timing accurate simulation / analysis of complete heterogeneous embedded architectures
  - Speedup > 10
  - Suited up to (1-10^{-6}) quantiles
- Tomorrow: system-level simulation with models of the functional behavior

High-level protocol layer

Application software

Ethernet

Gateway

Functional model


[RtAW-Pegase screenshot]
Timing analysis: Some/IP SD [7,8]

SOME/IP SD: service discovery for automotive Ethernet
Objective: find the right tradeoff between subscription latency and SOME/IP SD overhead

- Simulation complementary to worst-case analysis
- 2 steps: coarse grained models, then coupling with timing-accurate network simulator
- Same CPAL models could be used to implement testbeds
Simulation for .. safety-critical systems ?!

IMO: if system can be made robust to rare (quantified) deadline misses, then designing with simulation is more effective in terms of resource usage

Know what to expect from simulation – typically:

- Worst-case behaviors are out of reach but extremely rare events (e.g., \( Pr << 10^{-6} \) - see[1])
- Able to provide guarantees for events up \( Pr < 10^{-6} \) in a few hours
- Coarse-grained lower-bounds analysis to cross-validate

Sound simulation methodology

- **Q1**: is a single run enough ?
- **Q2**: can we run simulation in parallel and aggregate results ?
- **Q3**: simulation length ?
- **Q4**: correlations between “feared events” ?
Simulation for safety-critical systems?!

Know what to expect from simulation – typically:

- Worst-case behaviors are out of reach but typically extremely rare events (e.g., Pr << 10^{-6})
- Able to provide guarantees for events up Pr < 10^{-6} in a few hours
- Coarse-grained lower bounds analysis to cross-validate simulation methodology

Q1: is a single run enough?

Q2: can we run simulation in parallel and aggregate results?

Q3: simulation length?

Q4: correlations between "feared events"?

Tool support should help here:

Right: numbers in gray should not be trusted
Left: derive simulation time wrt target quantile
Timing-Augmented Model Driven Development

✓ Functional integration fails if control engineering assumptions not met at run-time: sampling jitters, varying response times, etc

Solution: injecting delays in the simulation - but how to do that early stage without knowledge of complete configuration?

Ongoing work [6,18]:
1. Designer defines timing-acceptable solution in terms of significant events: order & quantified relationships btw them
2. Derive QoS needed from the runtime systems: CPU, comm. latencies
3. Resource reservation & QoS ensured at run-time
Body of efficient formalisms & tools but

• models and their assumptions should be questioned by end-users
• cross-validation is a must

Ahead of us:

• lower-bounds with search intensive techniques
• better practices: validation benchmarks & proofs of result correctness
• Mixed-criticality (MC) timing analyses for MC constraints

Formal timing models cannot be safely used in systems that have not been conceived for timing analyzability → input for upcoming standards
✓ Timing-accurate simulation is well suited to automotive systems that can tolerate deadline misses with a *controlled* risk

✓ Today: timing accurate simulation of complete heterogeneous automotive communication architectures

✓ Tomorrow: system-level simulation with models of the functional behavior

✓ Formal methods most useful if 1) automated 2) integrated with standard development environments

→ need for timing-augmented MDD with correct by construct system synthesis


