A New Worst-Case Timing Approach for Automotive

Dr. Nicholas Merriam

Worst is not always best

Contents

- Introduction, motivation
- Basics of (Worst-Case) Timing Analysis
- Why today’s WCRT Analysis is problematic
- Why measurement and modelling are best friends
- Summary

Introduction

Why care about timing?

- No **safe** and **highly available** embedded software without rock-solid timing.
- If you don't *properly* care about timing, it will get you in the dark (= late in the project).
- Corrected timing can save $$$
  (cf. "*Timing analysis saves OEM €12m*" in Peter Gliwa's book)

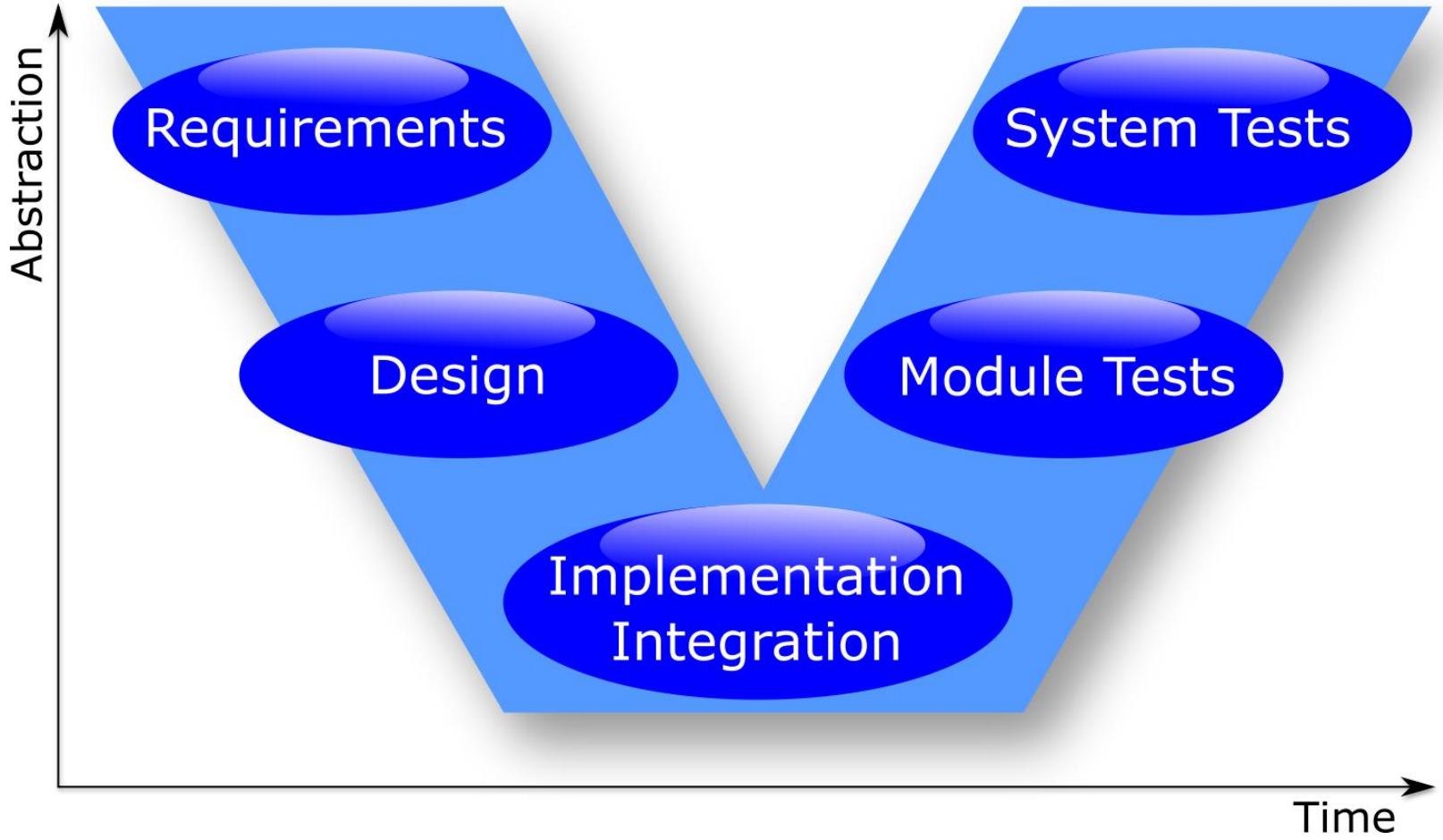
Why care about **worst-case** timing?

- Safety v. Availability
  - Fail-safe
- Timing is highly variable
  - **External** variation
    - Input signals arrive with jitter
  - **Internal** variation
    - Execution time varies, depending on software path
    - Response time varies, depending on pre-emption
- How many cases for ISO 26262 ASIL D/C/B/A?
  - Consider a single, *worst* case
    - Argue that other cases will function at least as well

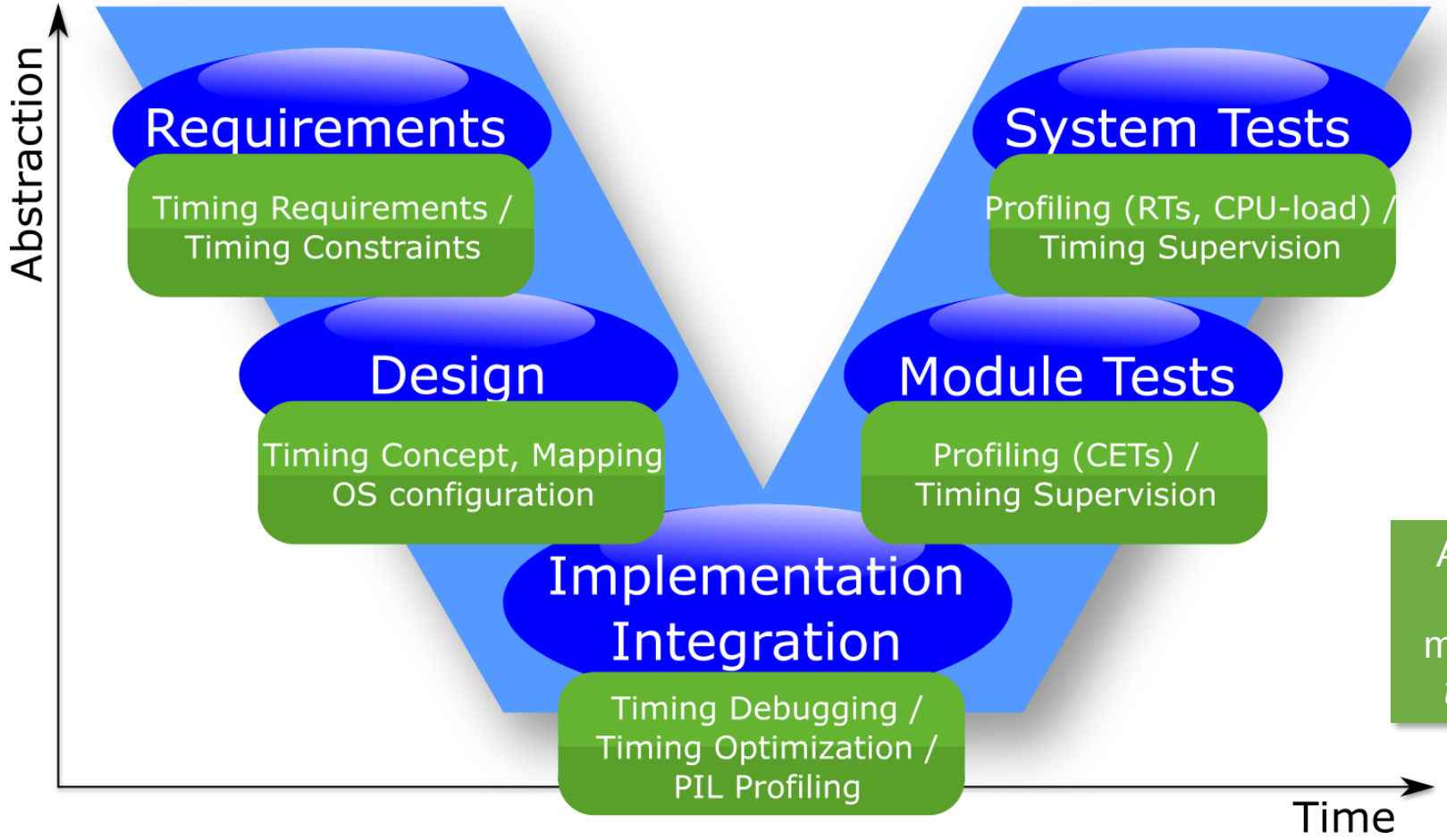
Basics of Timing Analysis

What is this?

The V-model as we know it

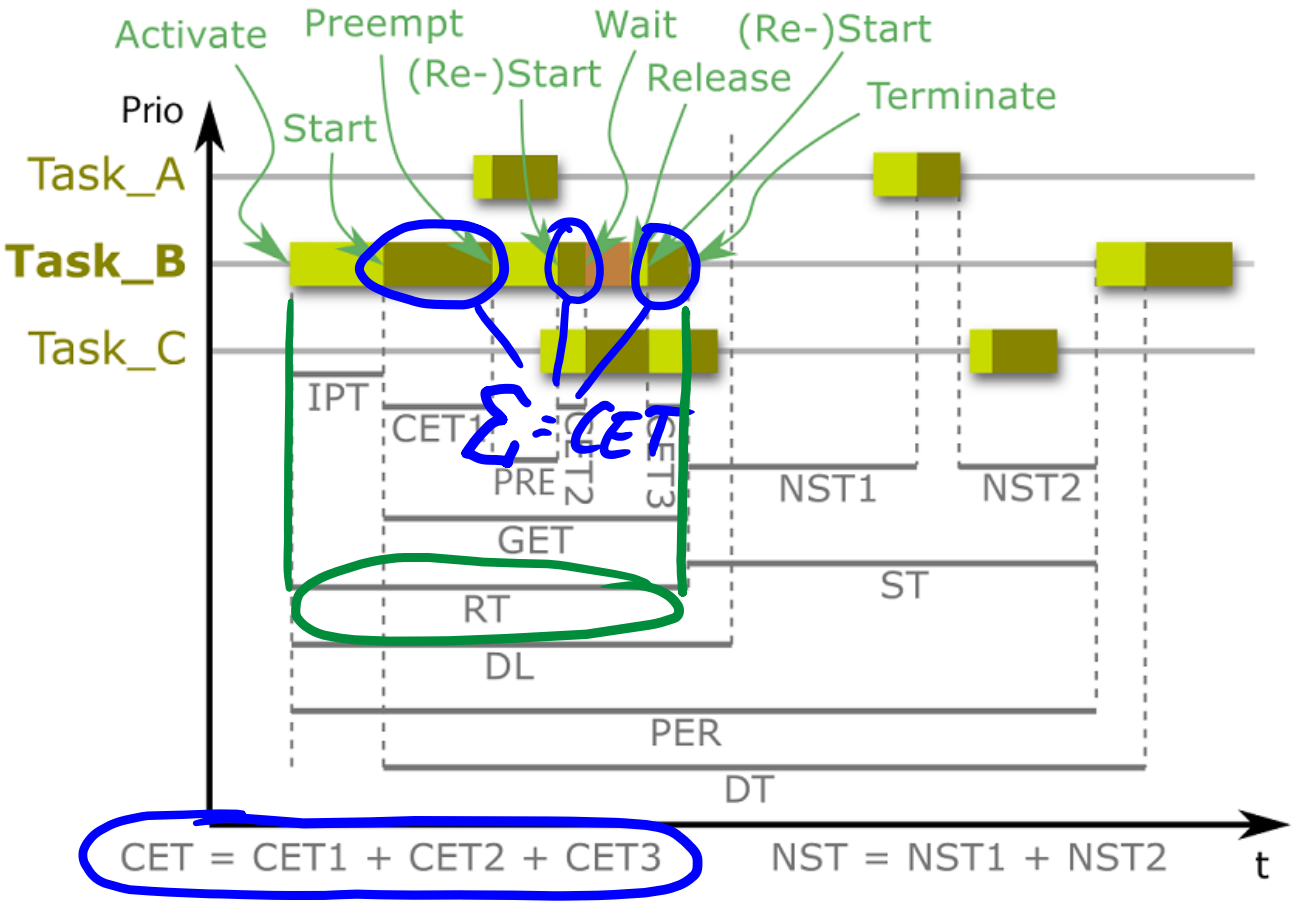


It is applicable to timing as well!



Apply your development processes and methodologies also to the *timing* of your software!

What are WCET and WCRT?



WCET = **W**orst **C**ase **E**xecution **T**ime
= theoretical maximum CET

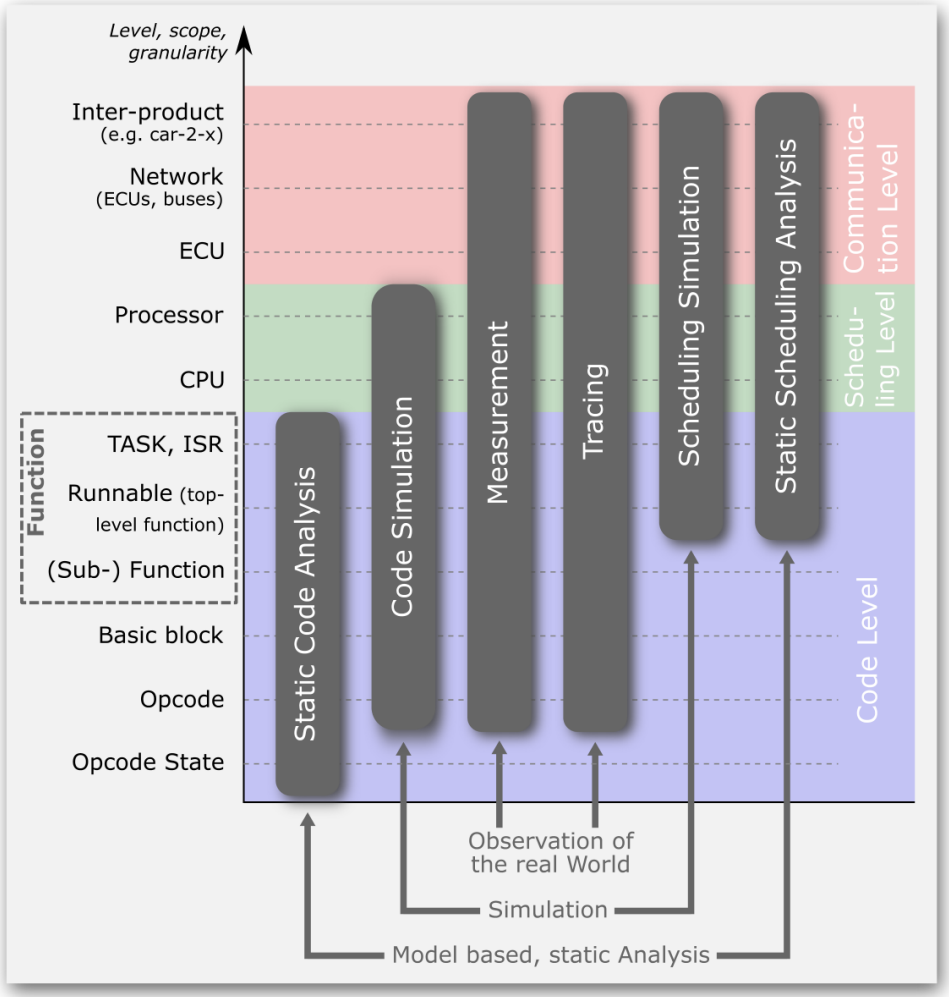
WCRT = **W**orst **C**ase **Response T**ime
= theoretical maximum RT

DL = Deadline (max. allowed RT)
timing constraint, timing requirement

Analysis Techniques: Summary

- Static Code Analysis
  - How? Analyze binary
  - What? Provide WCET
- Code Simulation
  - How? Simulate processor, execute target machine code
  - What? Run target code on x86
- Measurement
  - How? Instrument SW (T1.cont)
  - What? Get timing parameters, supervise SW
- SW-based Tracing
  - How? Instrument SW (T1.scope)
  - What? Get scheduling traces, see 'the real thing'
- Scheduling simulation
  - How? Simulate OS
  - What? Explore scheduling on x86
- Static Scheduling Analysis
  - How? Mathematical approach
  - What? Provide WCRT

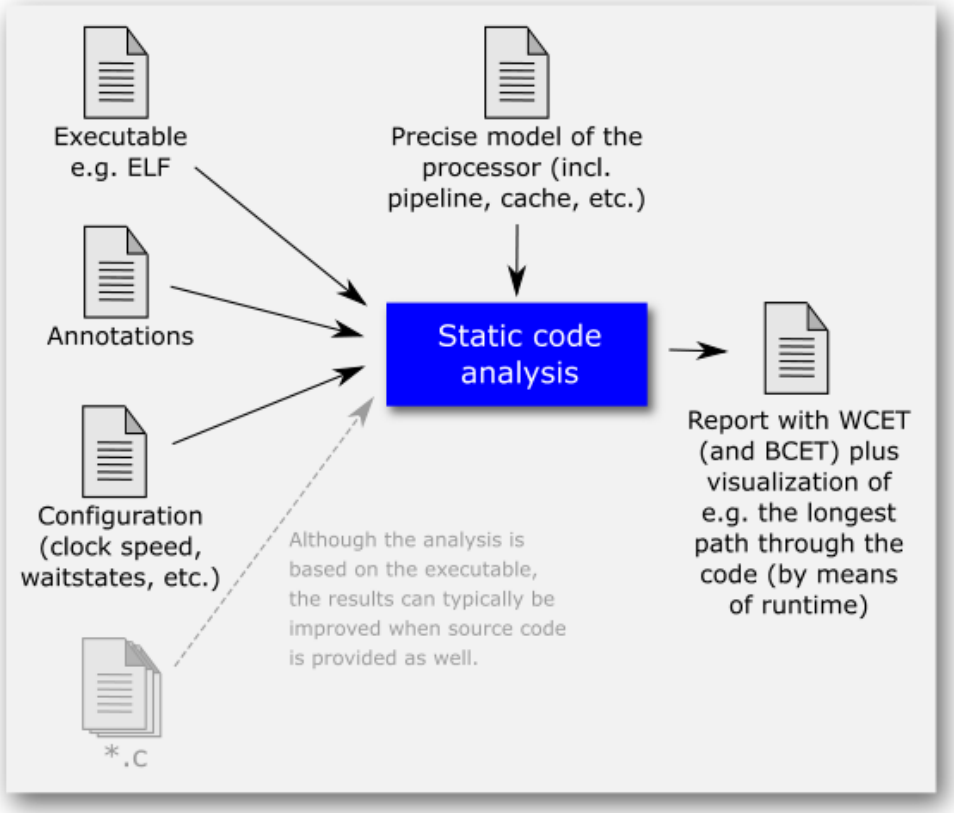
Overview Analysis Techniques



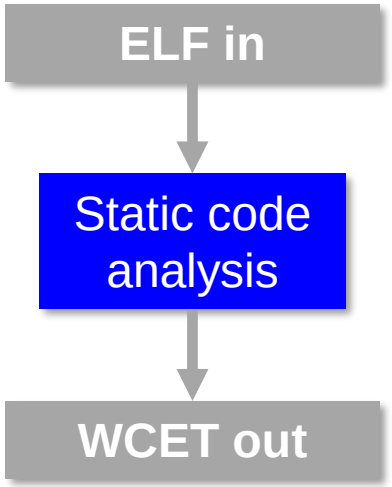
Model-based v. real world

- Model-based
  - Available before real hardware
  - Available before real software
  - Complex model is expensive
  - Requires validation of model
  - No embedded hardware needed
  - Analysis can be very fast
  - Analysis is easy to automate
  - Modelling is recommended
- Real world
  - Real hardware or detailed simulator
  - Limited before real software
  - Accurate measurement is not easy
  - Requires validation of test cases
  - Expensive hardware environment
  - Testing can be time-consuming
  - Hard to automate (*e.g.* test drive)
  - Some test evidence is mandated

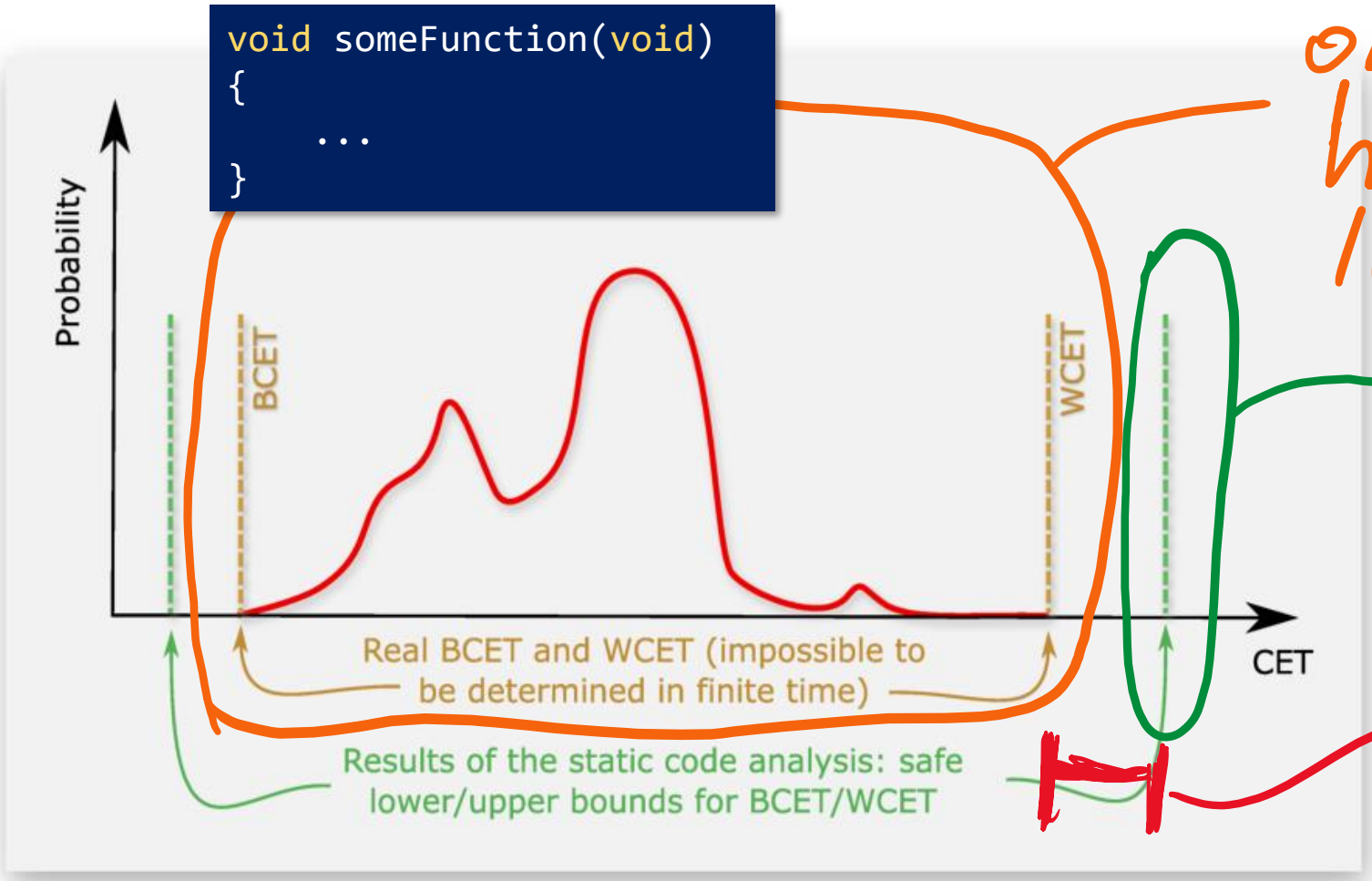
Static Code Analysis (WCET)



Or (more simple):



Static Code Analysis (WCET)



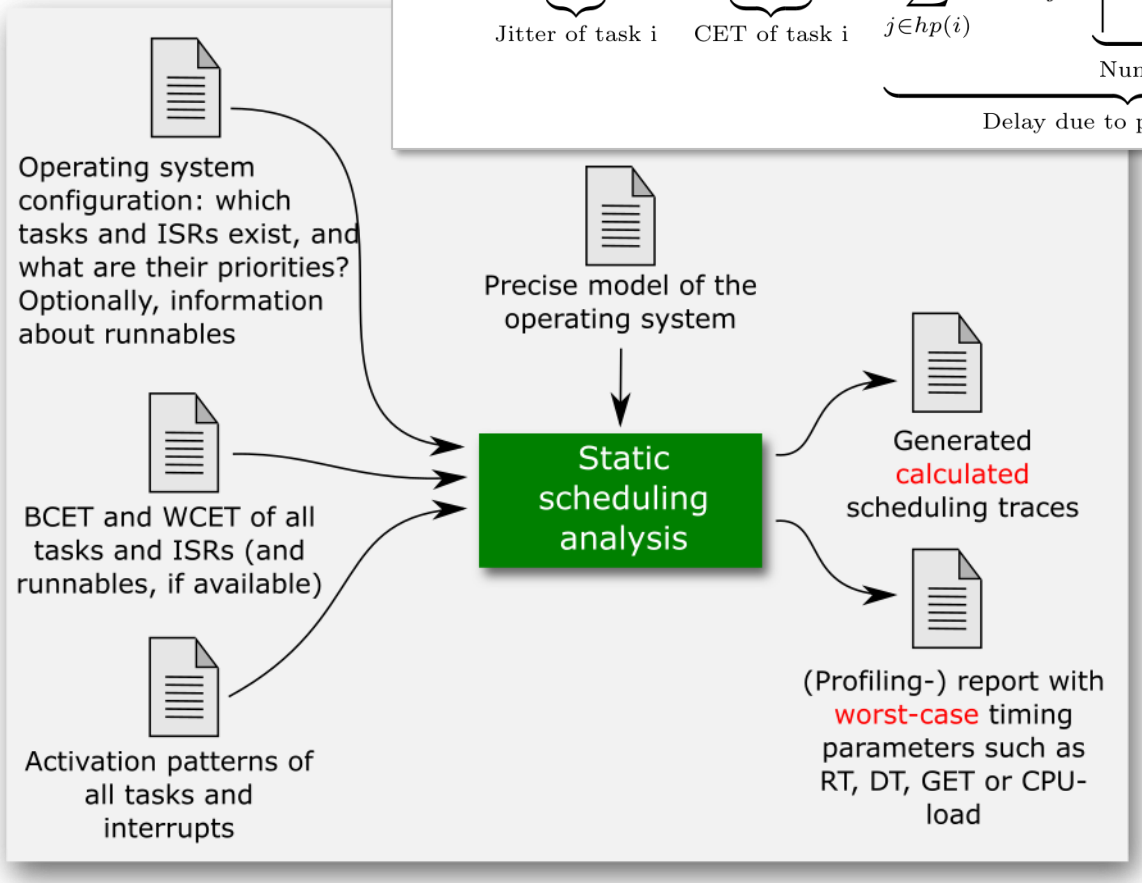
only God knows how this exactly looks like!

this is what we get from the tool

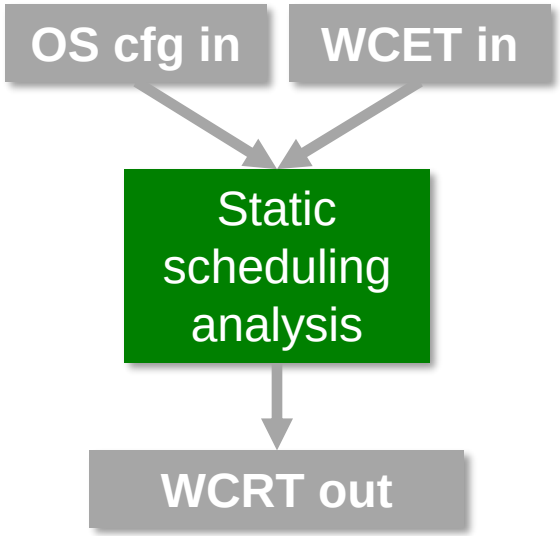
this gap might be very big!!

Static Scheduling Analysis (WCRT)

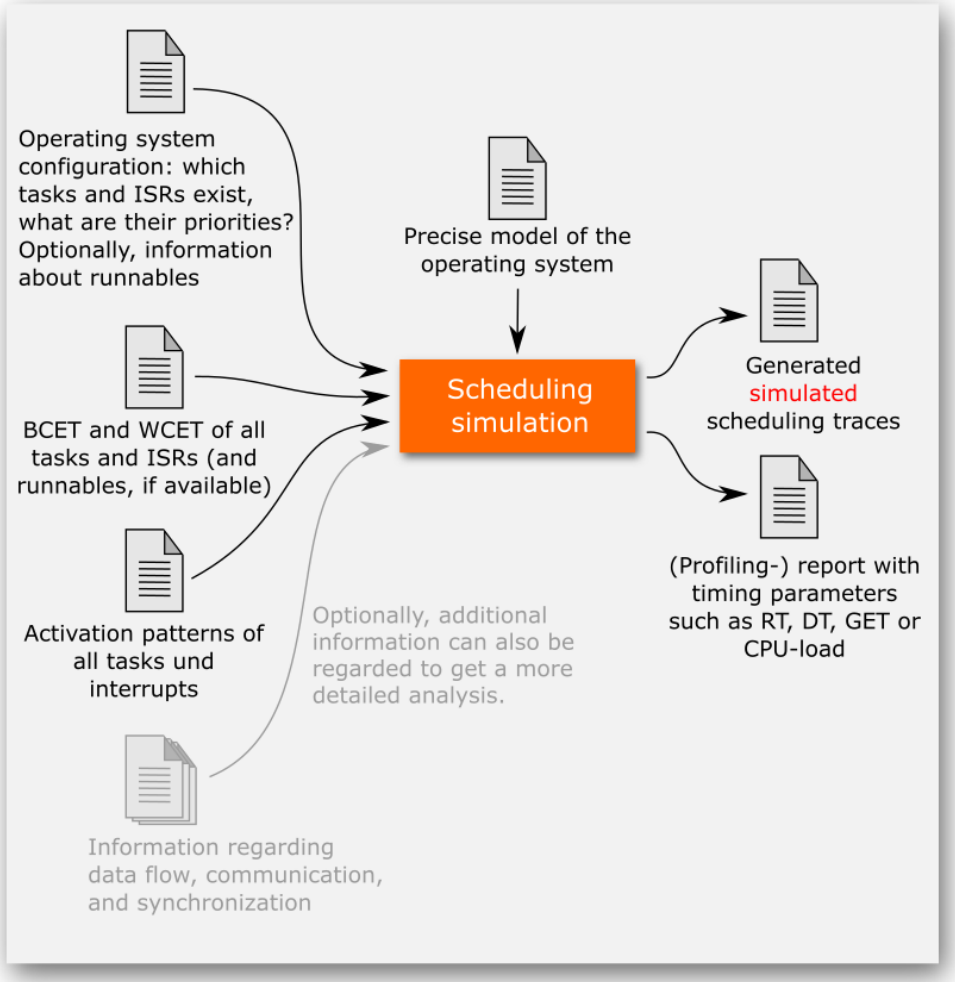
$$RT_i = \underbrace{J_i}_{\text{Jitter of task i}} + \underbrace{CET_i}_{\text{CET of task i}} + \underbrace{\sum_{j \in hp(i)} CET_j \cdot \underbrace{\left\lceil \frac{\overbrace{J_j + RT_i}^{\text{Observation interval}}}{PER_{0,j}} \right\rceil}_{\text{Number of preemptions}}}_{\text{Delay due to preemptions}} \leq \underbrace{DL_i}_{Deadline}$$



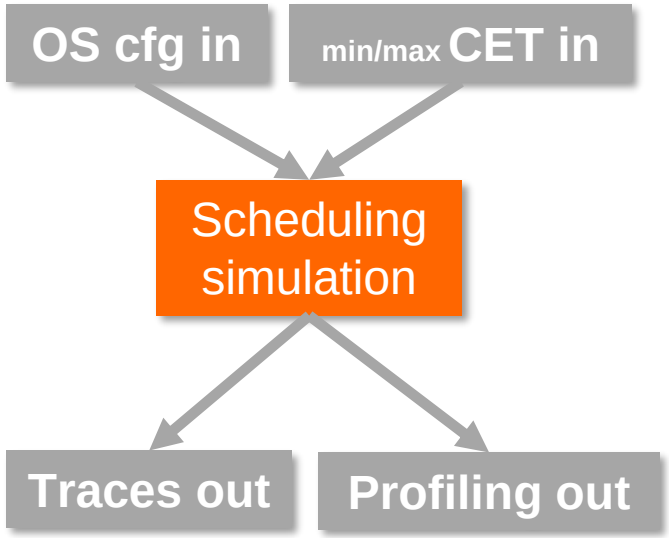
Or (more simple):



Scheduling Simulation



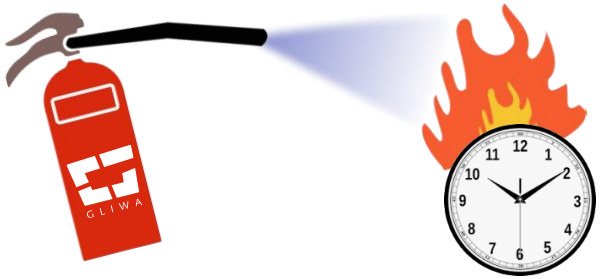
Or (more simple):



Why today's WCRT Analysis is problematic

What happens in real projects?

- GLIWA does a lot of ‘fire-fighting’: projects with timing issues ask for help.



- OEMs require more and more pessimism (more is not always better!)

- Result: **loss of focus;** some really important timing aspects get neglected.

Source: BSE-Galerie

Additional constraints: time consuming

- Dramatic **over-estimation** without additional information
  - Aperiodic tasks
  - Mutual exclusion
- Dangerous **under-estimation** without additional information
  - Jitter
  - Clock-drift

How deadlines are applied

- Today’s approach
  - Timing requirement is defined, e.g. $DL_{TaskB} = 1ms$
  - This translates to $RT_{TaskB} < 1ms$
  - For safety-relevant projects, this is interpreted as $WCRT_{TaskB} < 1ms$
  - Since the WCRT is not available, it is implemented as $upper\_bound < 1ms$

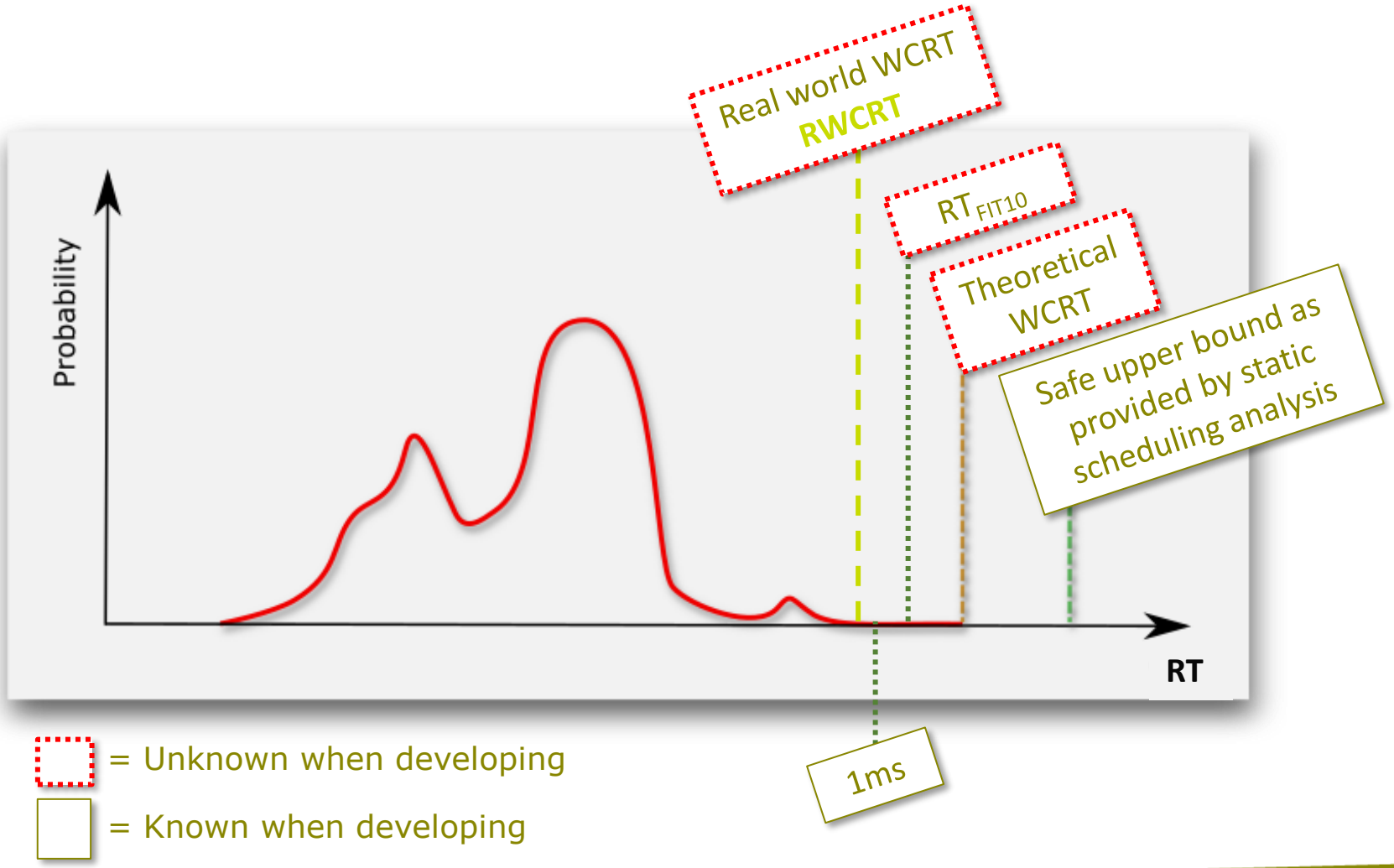
What is it that we need?

What does ISO26262 require?
For ASIL-D, less than 10 FIT meaning less than 10 faults in 10^9 hours of operation

→ Impossible to translate to a timing constraint

Definition 'Real world WCRT'
Looking back at the end of the life-time of all units: greatest RT value which ever occurred.
Let's call it **RWCRT**.

Our constraint is actually
DL = 1ms → RWCRT < 1ms



For WCET, see Peter Gliwa's talk

- **Slides**
  https://gliwa.com/downloads/EMCC2022_WCET_Peter_Gliwa.pdf

- **Video**
  Check out GLIWA's YouTube channel!


Why measurement and
modelling are best friends

Model-based  real world

- Model-based
  - Available before real hardware
  - Available before real software
  - **Validates testing**
  - No embedded hardware needed
    - Maximize hardware availability
  - Analysis can be very fast
  - Analysis is easy to automate
- Real world
  - Real hardware or detailed simulator
  - Limited before real software
  - **Validates model**
  - Expensive hardware environment
    - But no additional cost
  - Testing can be time-consuming
  - Hard to automate (*e.g.* test drive)

Validation

- Models contain unsafe errors
  - Not always trivial errors
  - Measured results can point to an error in the model
- Models contain unnecessary pessimism
  - Measured results can point to a safe improvement in the model
  - Mutual exclusion
    - Start engine in test mode
- Measurements omit test cases
  - Modelled results can point to a missing test case
- Measurement granularity is hard to guess (tasks/runnables?)
  - Modelled results can better focus measurement

A new approach to embedded timing

1. Use measurement **and** model-based methods

2. Use measurements to refine models and
   models to refine measurements

3. Make timing consideration a first-class part of embedded software
   - …rather than hoping to get the seal of approval at the end of development.

I have a dream…

- In this dream, we get together
  - OEMs
  - Tier-1s
  - Timing tool vendors
  - Timing pioneers (for example academics)

Freephotos

- We discuss
  - The facts
  - The needs
  - The requirements
  - Possible solutions

Defensive code with respect to timing

Defensive code

Check inputs even when they are expected to be correct / in range.

- On code-level

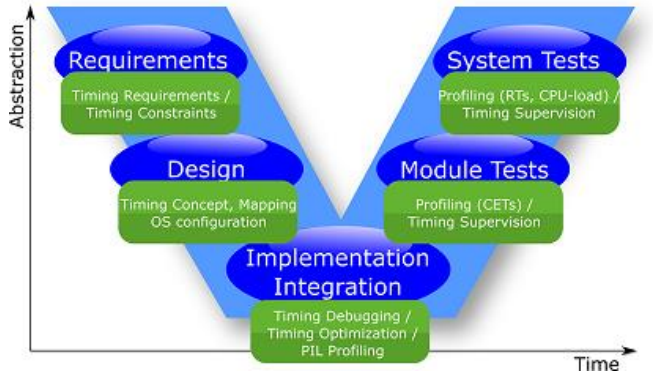
```
void someFunction(void)
{
    unsigned int i;
    WCET_ASSERT( a <= 42 );
    for (i=0; i<a; i++) {
        ...
    }
}
```

- On scheduling-level

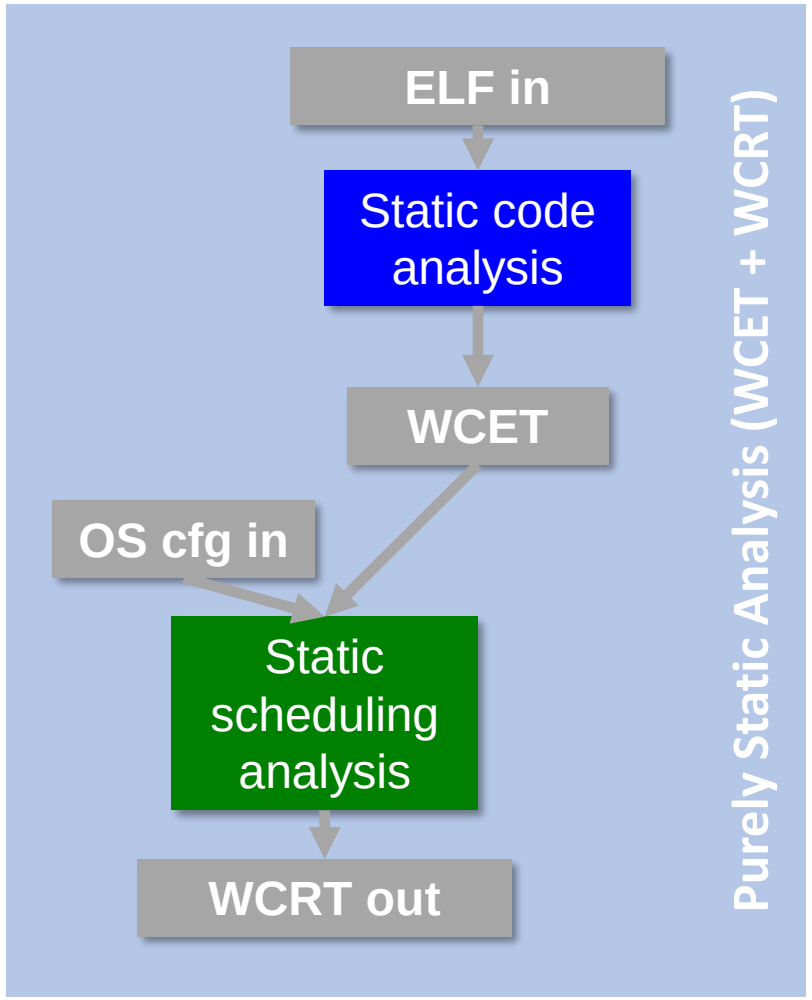
  → timing protection (e.g. through AUTOSAR or T1.cont)

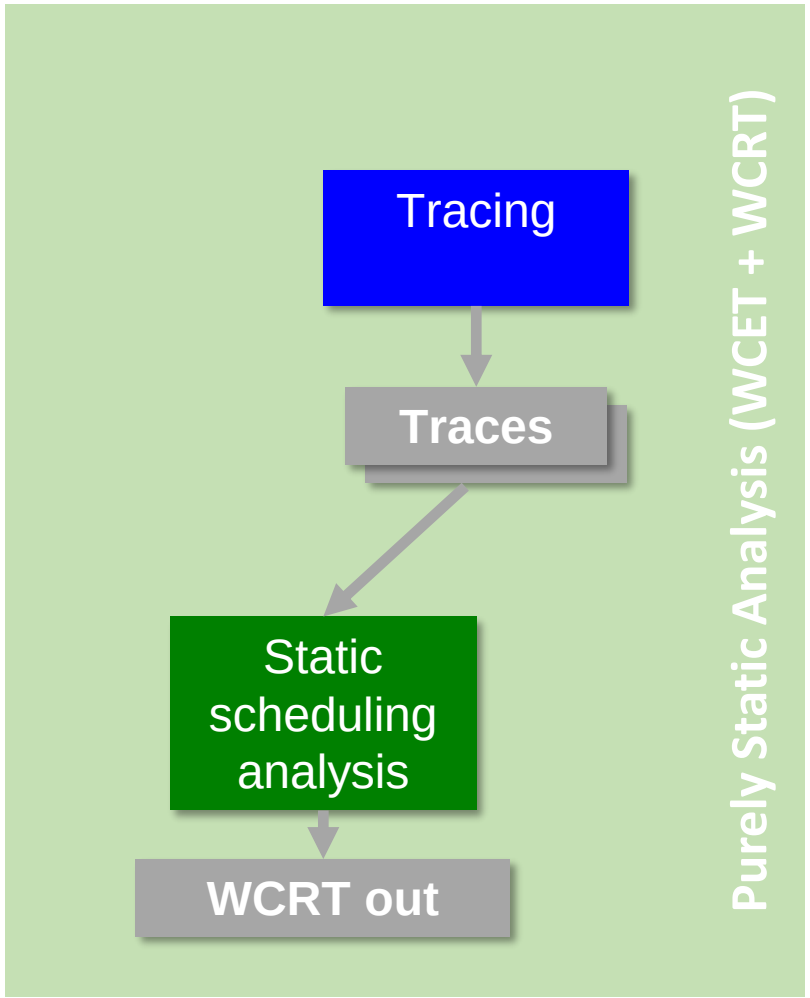
Timing integration in development

- Unit/module tests
  - What is the timing with no pre-emption, no memory conflicts, no cache misses? Is it *already* close to the limit?
  - Beware of premature optimization



Possible inputs for static scheduling analysis





Summary

Summary

- Embedded Software Timing does matter!
- Addressing a purely theoretical WCRT binds resources and moves the focus away from real timing issues.
- Use the best of each: combine model-based techniques with measurement/tracing (not just for verification!)
- Let’s get together and think about a more sensible future worst case timing approach.


GLIWA
Nicholas Merriam
Director
fon +49 - 881 - 13 85 22 - 34
fax +49 - 881 - 13 85 22 - 99
nicholas.merriam@gliwa.com
Manager Target Development
GLIWA Ltd.
c/o ACOLLECTIVE
22 Pavement
York, North Yorkshire
YO1 9UP
United Kingdom
ANALYSIS
gliwa.com
embedded systems

Thank you