Why we need a new Worst Case Timing Approach for Automotive

EMCC2022, 11th Oct 2022, Peter Gliwa

The worst case tale (as told for hundreds of years)

How long does it take the princess to get inside to safety when the dragon comes??

Tricky, because she needs to get her hair up first...

Prince of static code analysis: "Only I can tell thee the **worst case** time to get back inside!"

Before telling you how the story ends,
we need to discuss some background.

Contents

- Introduction, motivation
- Basics of (Worst Case) Timing Analysis
- Why today's WCET Analysis is problematic
- Let's start a new chapter
- 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).
- Optimized timing can save $$$
  (cf. "*Timing analysis saves OEM €12m*" in my book)

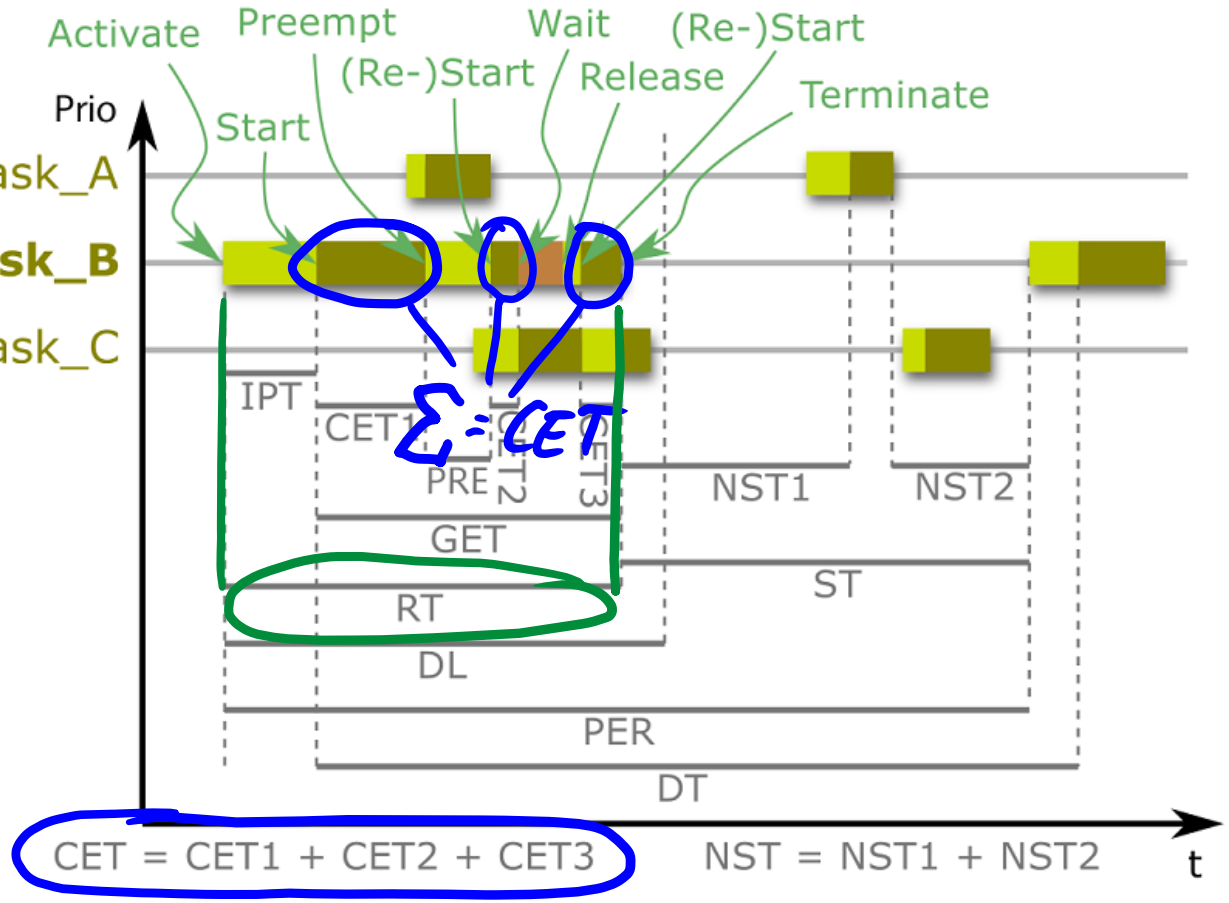
Why care about **worst case** timing?

- Safety-relevant projects need to address corner-cases not covered by testing.
  → ISO 26262

- **Increase availability**
  The system should still work even if timing beyond what was tested occurs.
  Switching to some error mode is safe but typically comes with reduced functionality.

Basics of Timing Analysis

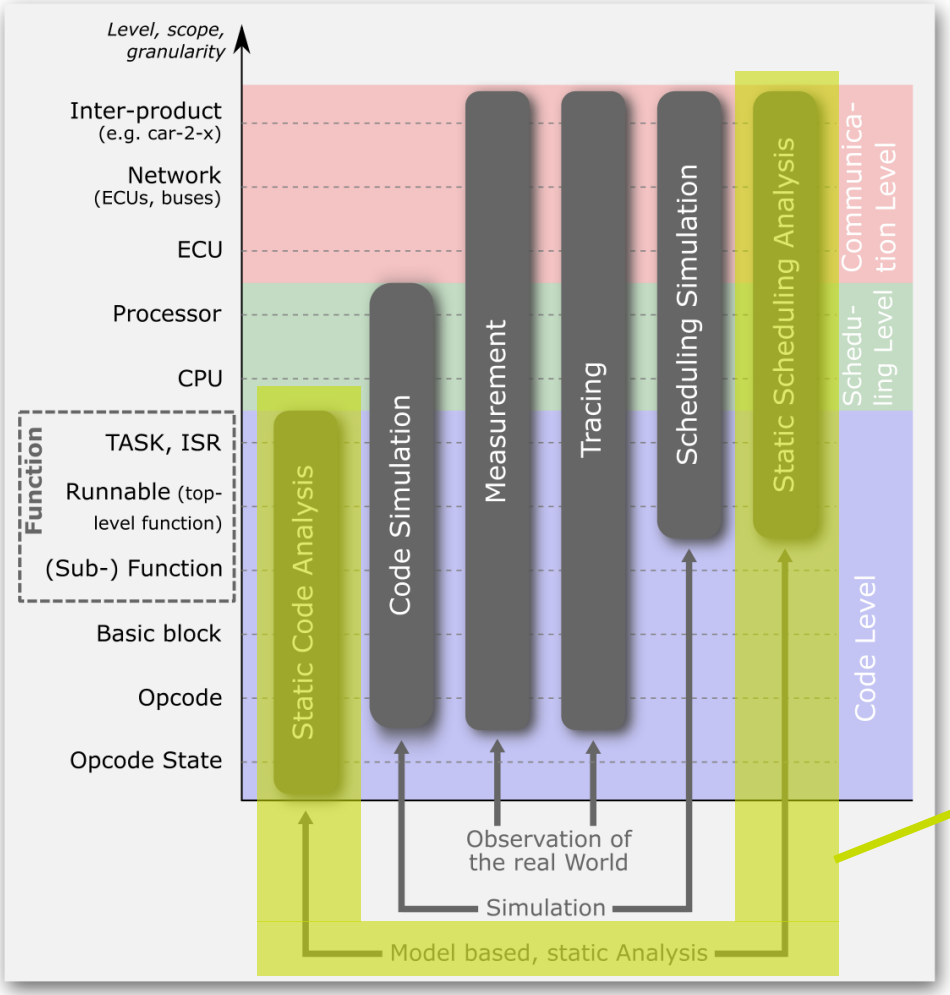
What are WCET and WCRT?



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

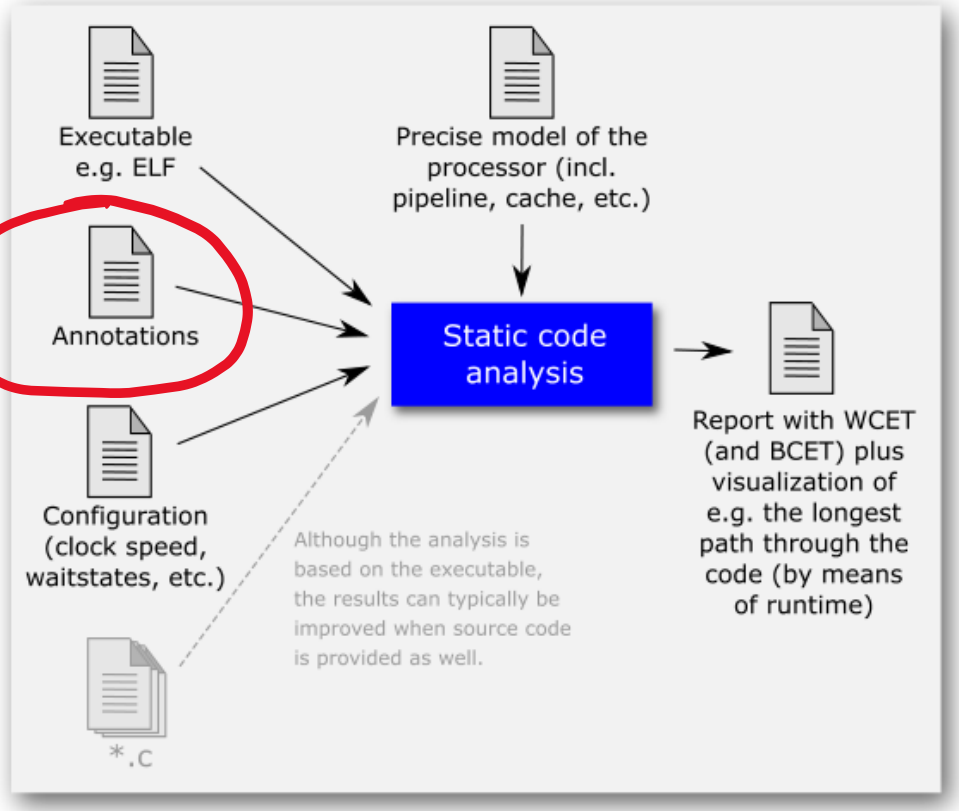
WCRT = **W**orst **C**ase **R**esponse **T**ime
= theoretical maximum RT

Overview Analysis Techniques

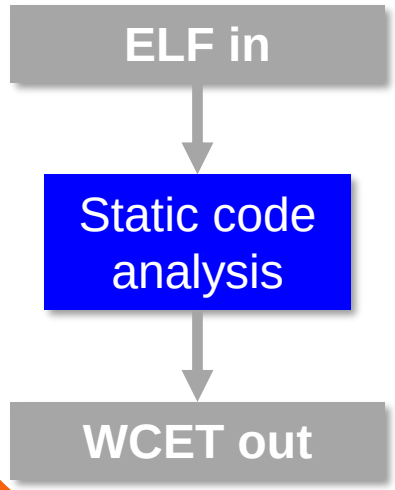


Today's focus

Static Code Analysis (WCET)



Or (more simple):

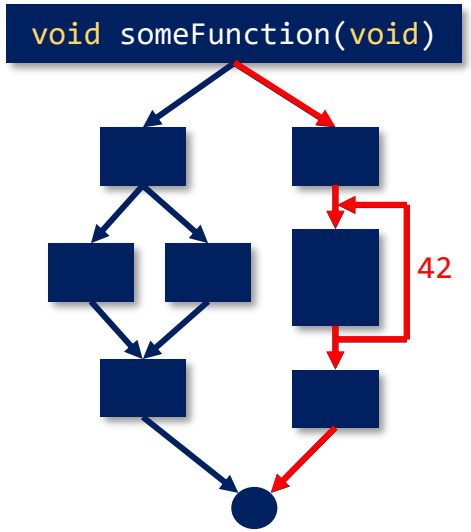


this is what you will need to invest a huge effort into!

it's so nice and simple, it's the only thing many will remember

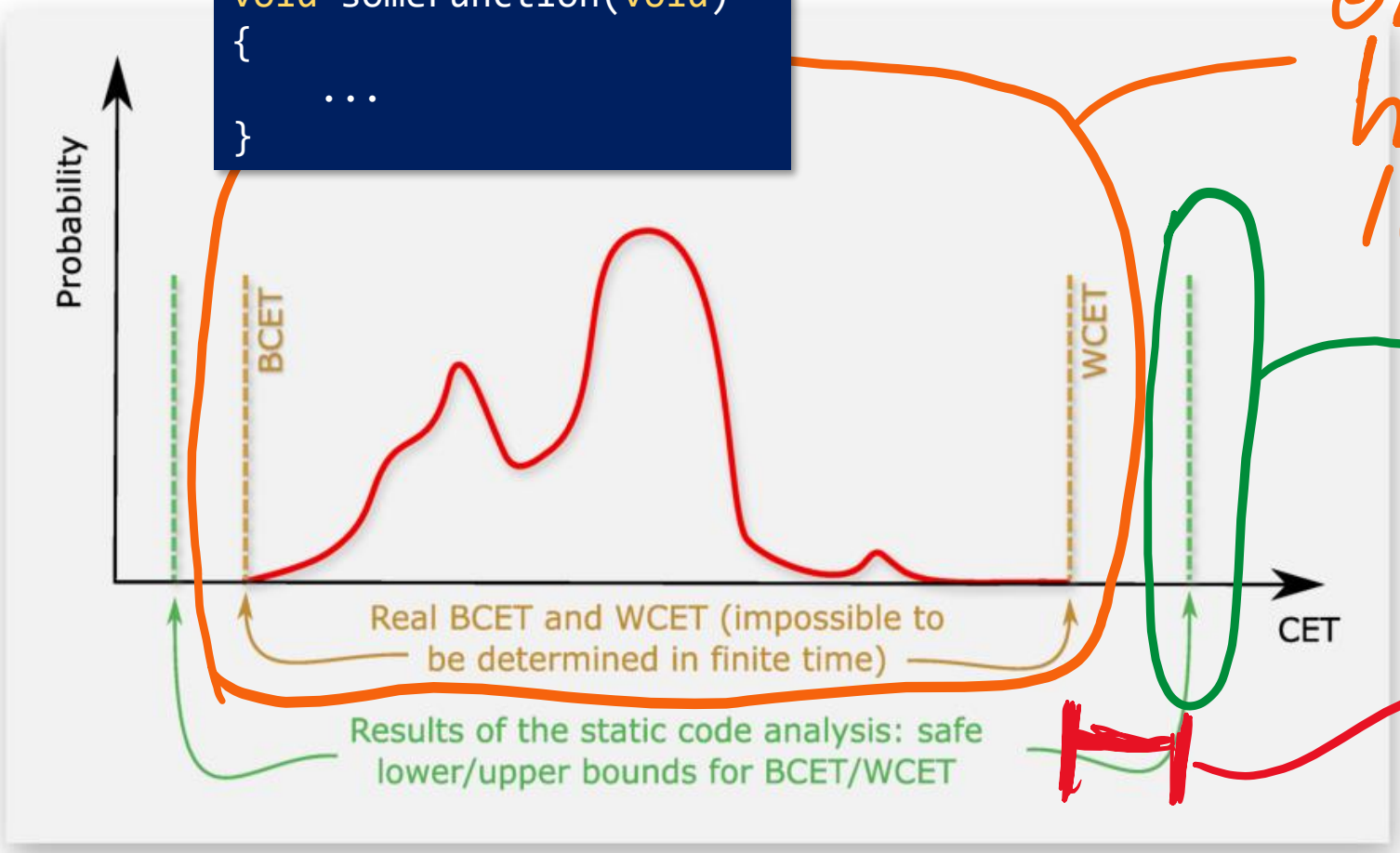
Static Code Analysis (WCET)

- Example: determine WCET of function `someFunction`
- How does Static Code Analysis (WCET) work?
  - Based on the binary, `someFunction` gets disassembled
  - All calls/jumps get identified, a call tree gets generated.
  - Using abstract interpretation, the longest path (greatest number of loop iterations etc.) through the code is identified.
  - The analysis makes sure, that the initial states of cache, pipeline, etc. are such that the execution of the longest path shows the maximum possible execution time.
- Does the result depend on the input (test) data?
  - No! This is the great advantage of static code analysis.
  - However, you might need to annotate (see later).



Static Code Analysis (WCET)

```
void someFunction(void)
{
    ...
}
```



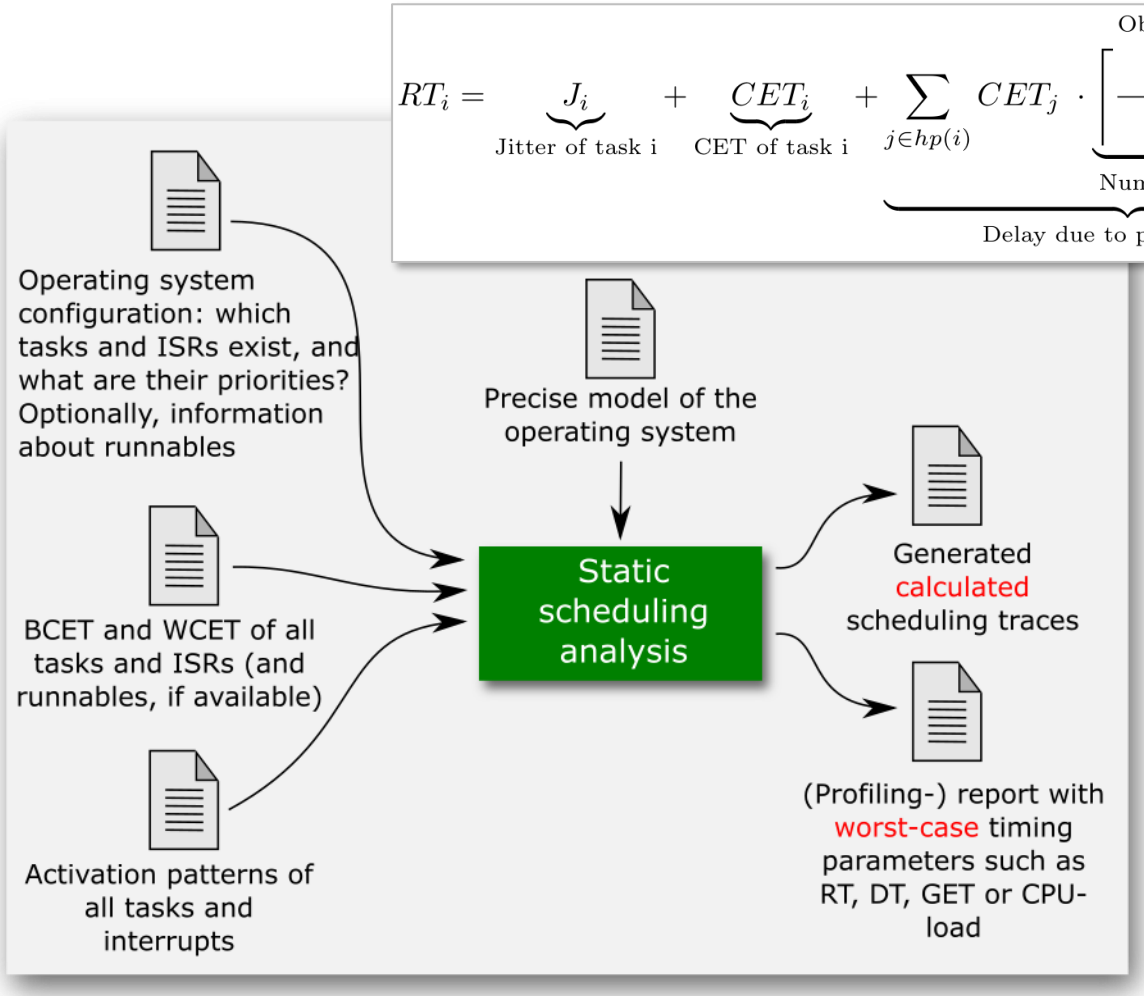
only God knows how this exactly looks like!

this is what we get from the tool

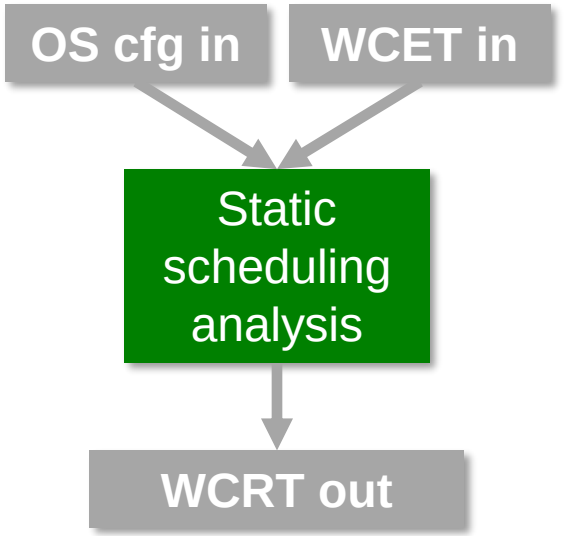
this gap might be very big!!

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):



Why today's WCET Analysis is problematic

How the worst case tale ends (NO happy end)

Princess: "Okay, what is the WCET?"

Prince: "2 hours"

Princess: "What??"

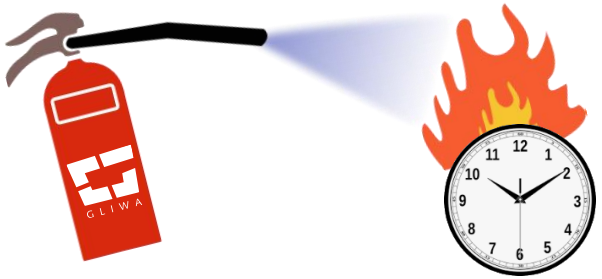
Prince: "Well, a meteorite might fall onto the end of your bunch blocking it while a cow passes by getting trapped in your hair while....."

Princess: "?????"

And before dashing off, he – instead of killing the beast – suggests she should cut her hair...

Okay, that was the tale. How about the real world?

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



- One recent example: automotive ASIL-D project

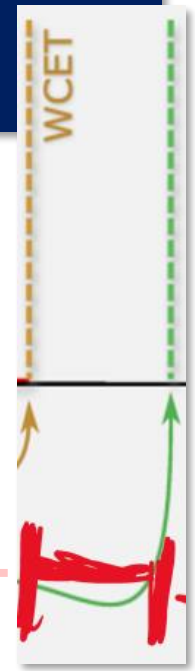
- OEM requirement: “The WCET has to be provided for all tasks and interrupts.”

WCET!

Annotations: time consuming and dangerous

- Abstract interpretation is not always able to identify the upper loop bound of loops.
  → max. value is used. Here, e.g. a = 2^{32}-1
- If the real max. value is, let's say, 42, the WCET overestimation in enormous!
- Add an annotation to tell the tool a = 42
- Real projects often have hundreds of annotations → **very time consuming!**
  → more important problems get neglected
- Many annotations relate to third party object code → **error-prone, dangerous!**

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



Some thoughts about probability

- Today's approach
  - Timing requirement is defined, e.g. $CET_{TaskB} < 1ms$
  - For safety-relevant projects, this is interpreted as $WCET_{TaskB} < 1ms$
  - Since the WCET is not available, it is implemented as upper_bound < 1ms
- *P(CET=WCET)* is likely to be a **very** small number. Think of
  - Winning the 6/49 lottery a thousand times in a row
  - Blasting a pot of paint and seeing the full text of the bible after the paint settled
  - Having all humans wiped out in a second, each by its individual shooting star
- Do such events play any role in our real world's life?
  NO!! So why should they when it comes to timing?

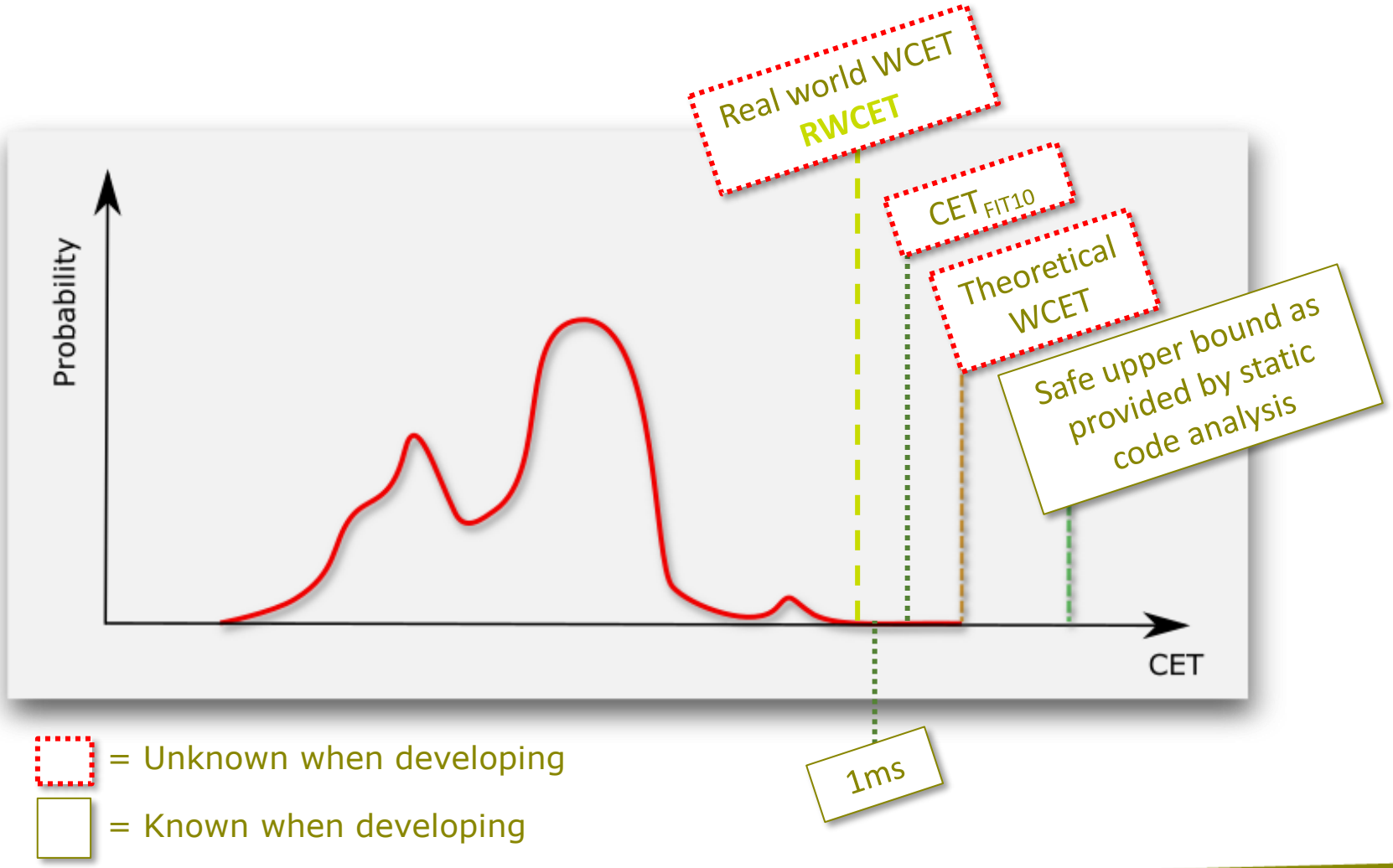
What is it that we need?

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 WCET'
Looking back at the end of the life-time of all units: greatest CET value which ever occurred.
Let's call it **RWCET**.

Our constraint is actually
RWCET < 1ms



Worst Case rarely is Worst Case

- WCET static code analysis typically
  - does not consider interrupts (assumption: there are no interrupts)
  - does not consider multicore effects such as conflicts at the memory interface or cross-bar arbitration (assumption: there are no other cores)
  - does not consider DMA (assumption: DMA not used)
  - does not consider data cache write-backs (assumption: data caches disabled)

- In the past 20 years, I have not come across a single *real* project which fulfils these assumptions.

- **The worst case people use in safety relevant projects is not the worst case!**

Worst Case rarely is Worst Case

- WCET static code analysis t
  - does
  - does
  - does
  - does

Recommendation from static analysis experts:
Measure / estimate the impact of these effects and configure/annotate static analysis accordingly.

terface cross-bar arbit

assumption: data

What??

Have they not told us for years that measurements are the straight path to hell?? And that the theoretical WCET is the one and only truth???

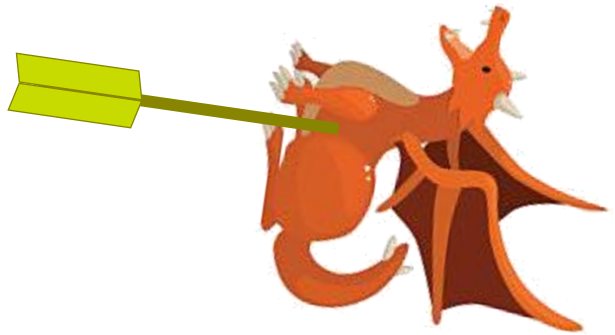
- In the past 20 years, I have n fils these assumptions.
- **The worst case people use in safety relev ase!**


Let’s start a
new chapter

I have a dream…

- In this dream, we get together
  - OEMs
  - Tier-1s
  - Timing tool vendors
  - Timing enthusiasts (from universities e.g.)

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



Summary

Summary

- Embedded Software Timing does matter!
- OEM's WCET requirements can lead to poorer quality. Example: ASIL-D project adds a degradation concept (and thus more complexity) just to fulfill WCET requirements.
- Addressing a purely theoretical WCET binds resources and moves the focus away from real timing issues.
- Let's get together and think about a more sensible future worst case timing approach.


GLIWA
Peter Gliwa
Dipl.-Ing. (BA)
Geschäftsführer (CEO)
GLIWA GmbH embedded systems
Pollinger Str. 1
82362 Weilheim i.OB.
Germany
fon +49 - 881 - 13 85 22 - 10
fax +49 - 881 - 13 85 22 - 99
mobile +49 - 177 - 2 57 86 72
peter.gliwa@gliwa.com
www.gliwa.com

Thank you