Timing Requirements and Timing Verification

A suggestion based on experience

Peter Gliwa

Contents

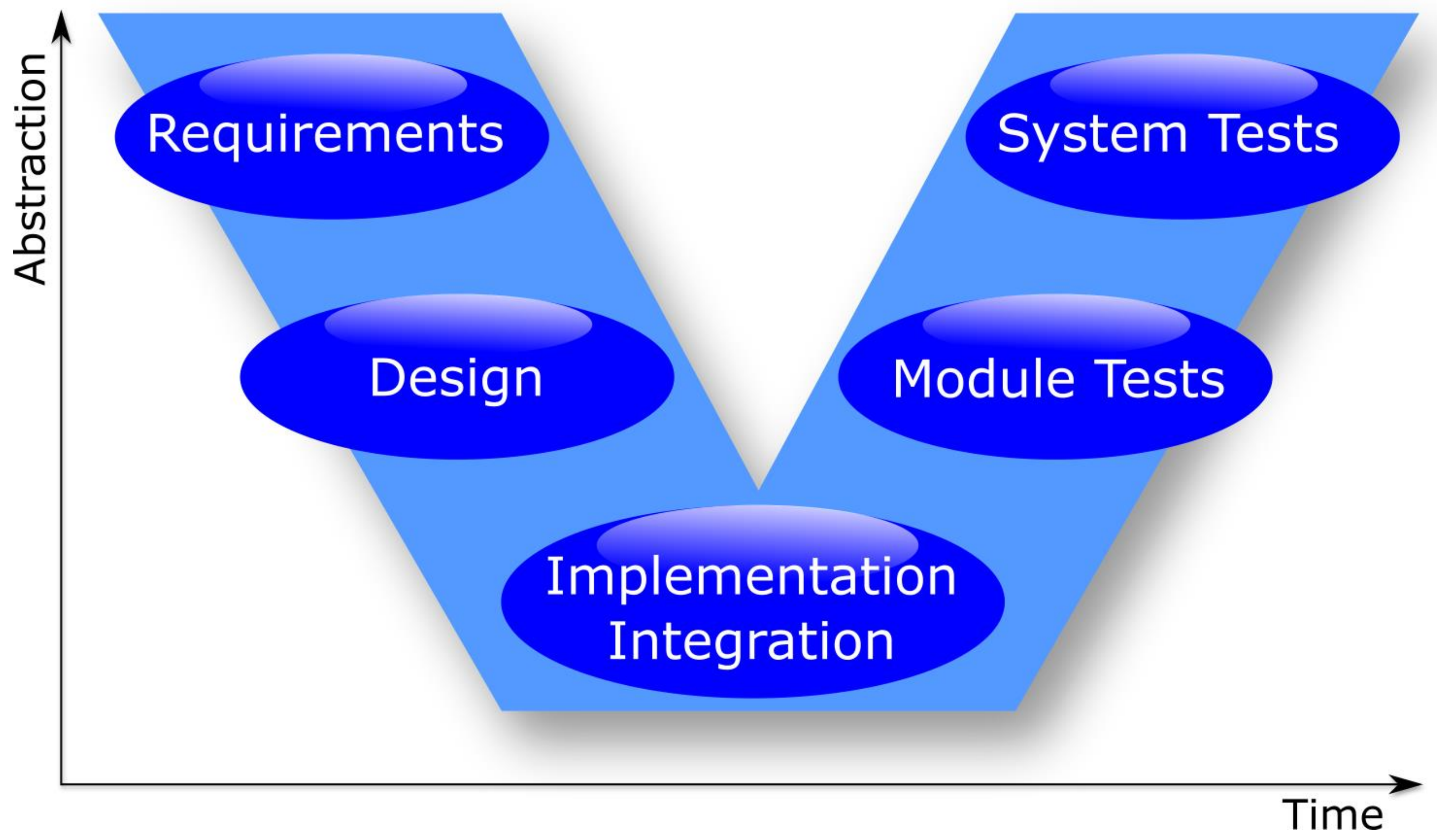
Motivation

Why care about timing?

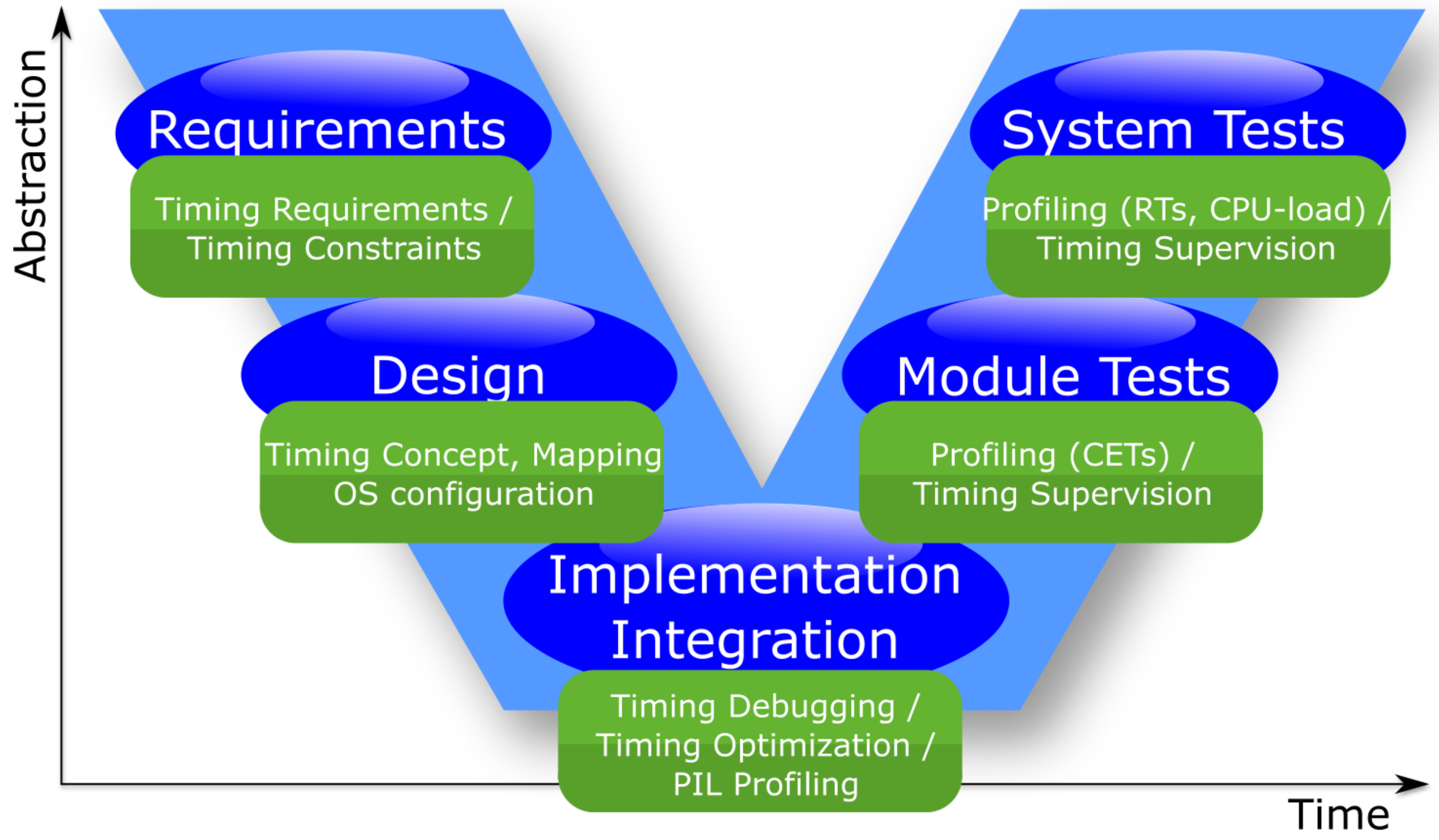
- No **safe** and **highly available** embedded software without rock-solid timing.
- If you don't *properly* care about timing, it will get you in the dark (= late in the project).
- Optimized timing can save $$$
  (cf. "*Timing analysis saves OEM €12m*" in my book)

What is this?

The V-model as we know it

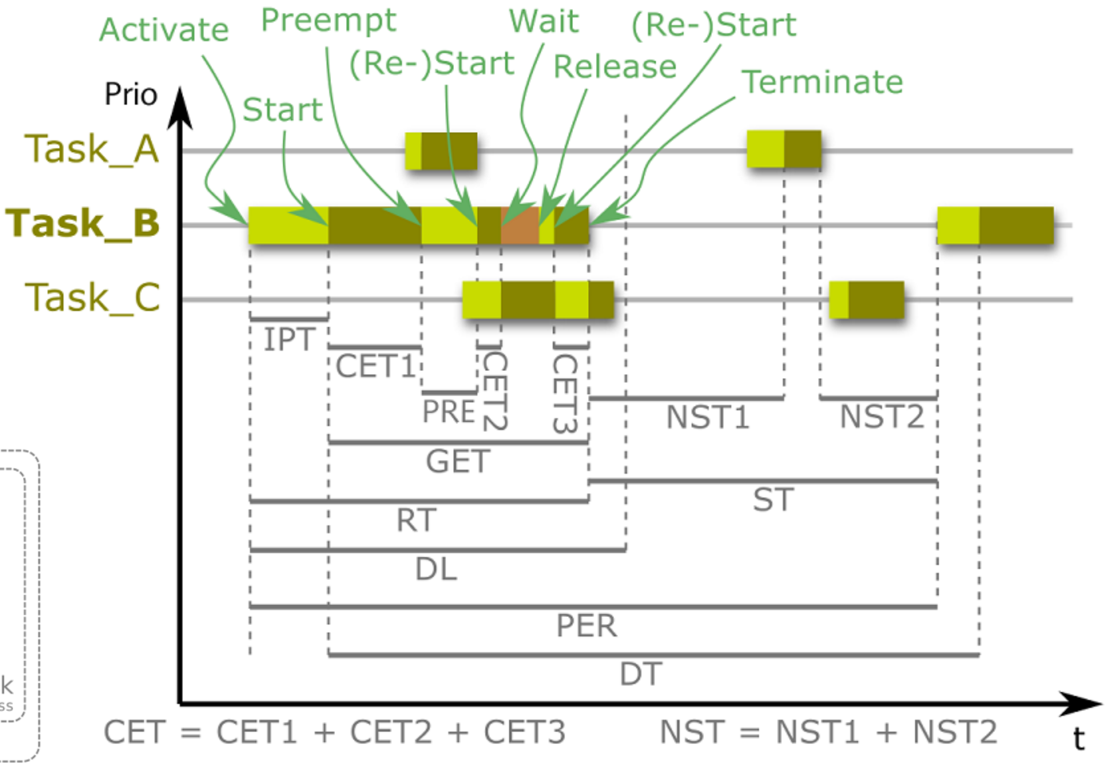


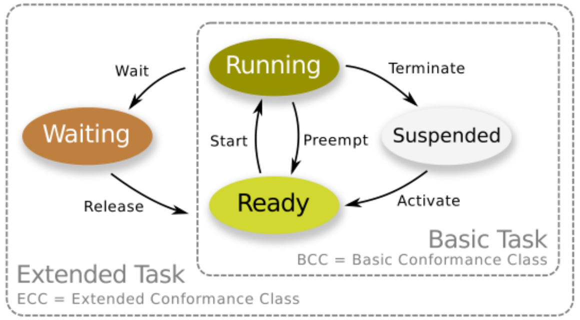
It is applicable to timing as well!


Abstraction
Requirements
Timing Requirements / Timing Constraints
Design
Timing Concept, Mapping OS configuration
Implementation Integration
Timing Debugging / Timing Optimization / PIL Profiling
Module Tests
Profiling (CETs) / Timing Supervision
System Tests
Profiling (RTs, CPU-load) / Timing Supervision
Time

Basics of Timing Analysis

Timing parameters



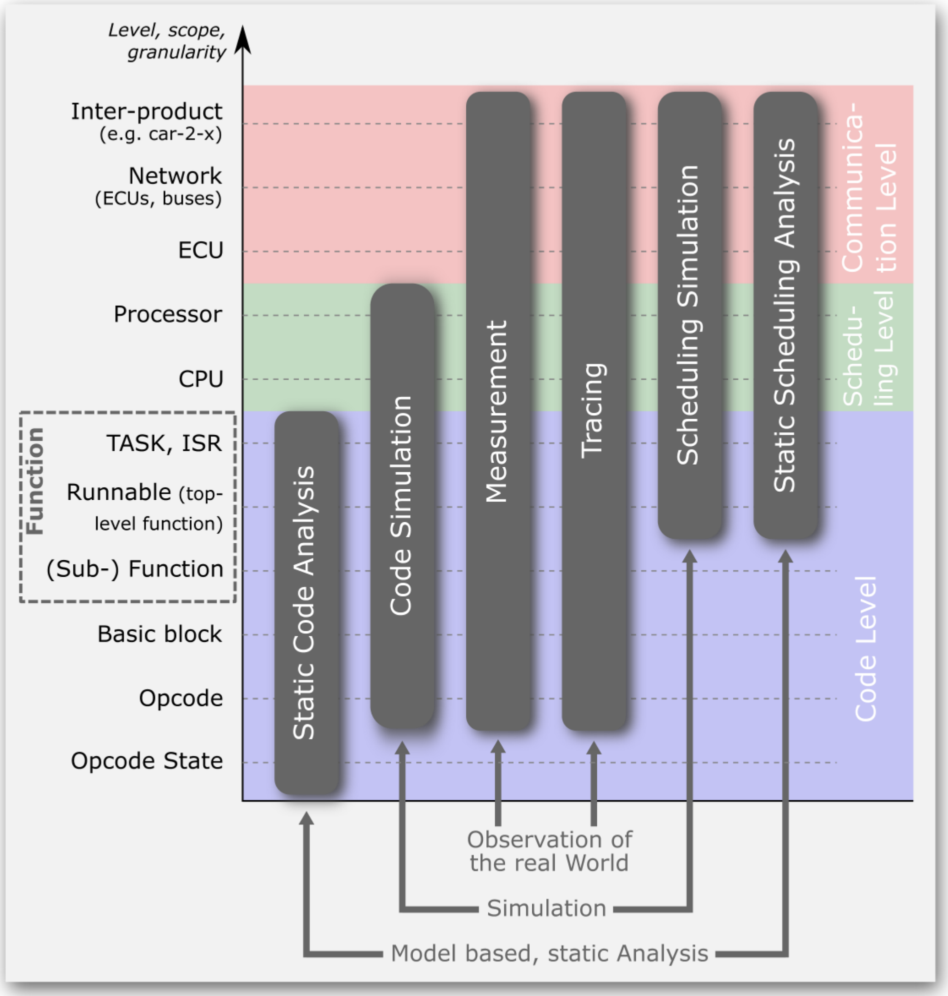


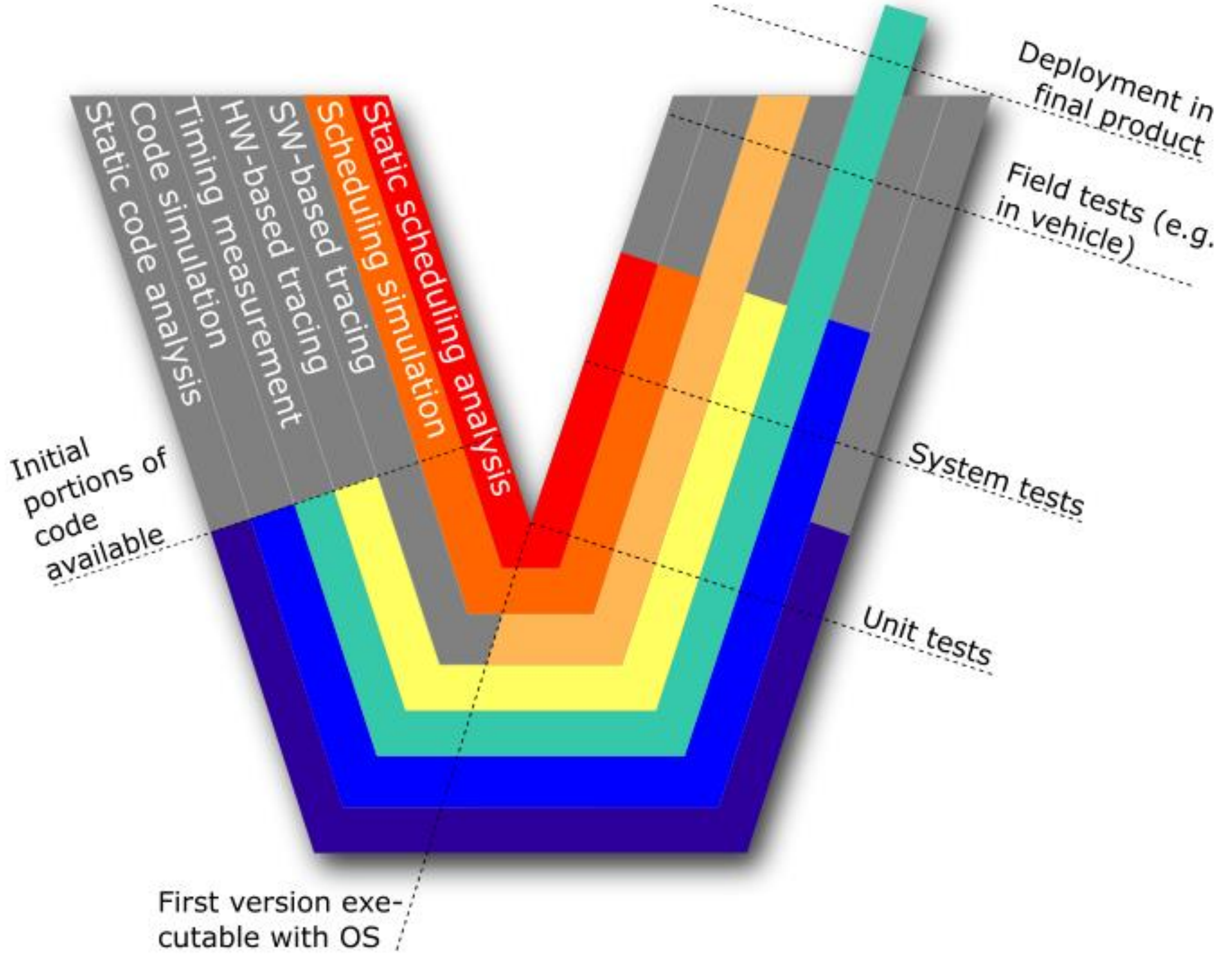
Abr.	Explanation (EN)	Erklärung (DE)
IPT	initial pending time	Initialwartezeit
CET	core execution time	Nettolaufzeit
GET	gross execution time	Bruttolaufzeit
RT	response time	Antwortzeit
DT	delta time	Deltazeit
PER	period	Periode
ST	slack time	Restzeit
PRE	preemption	Unterbrechungszeit
JIT	jitter	Jitter
CPU	cpu load	CPU Auslastung
DL	Deadline	Deadline
NST	Net slack time	Nettorestzeit

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




Static code analysis
Code simulation
Timing measurement
HW-based measurement
SW-based tracing
Scheduling simulation
Static scheduling analysis
Deployment in final product
Field tests (e.g. in vehicle)
System tests
Unit tests
Initial portions of code available
First version executable with OS

Timing requirements
Timing methodology

GLIWA recommendation

- **Step 1**
  Collect timing requirements through interviews
- **Step 2**
  Initial timing design using scheduling simulation, create timing budgets
- **Step 3**
  Monitor budgets through timing measurements (e.g. T1.cont)
  Monitor scheduling through tracing (e.g. T1.scope, T1.steaming)
- **Step 4**
  Optimize timing (see chapter “Timing optimization” of my book)
- **Step 5**
  Verify timing through **automated (!)** timing tests

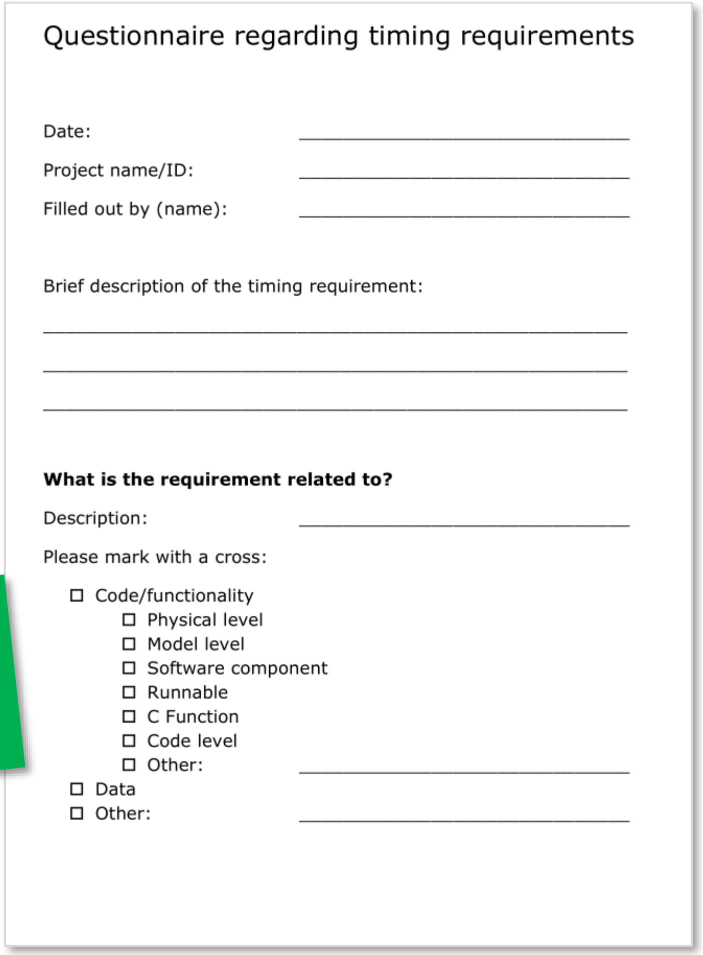
This is our suggestion, a good fit for many (but not all) projects.

Step 1: Collecting timing requirements

- Interviews with
  - Functional developers
  - Integrators
  - BSW experts
  - Experts from previous generation of ECU (if any)

Free!

Download https://gliwa.com Book → Online material



Questionnaire regarding timing requirements

Date: ____________________

Project name/ID: ____________________

Filled out by (name): ____________________

Brief description of the timing requirement:

____

____

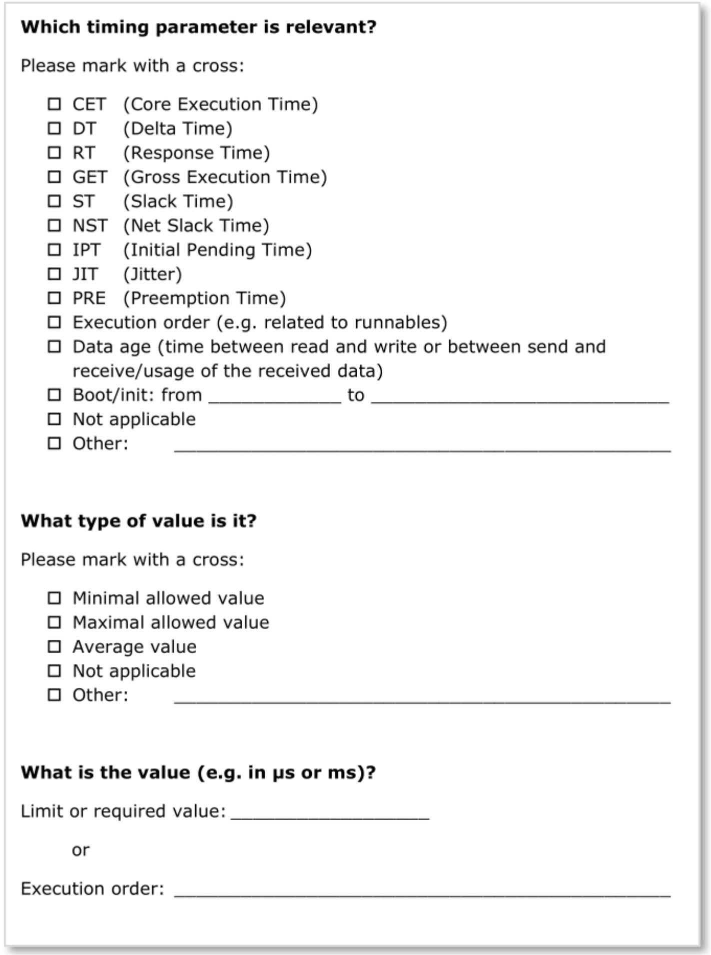
____

What is the requirement related to?

Description: ____________________

Please mark with a cross:

- ☐ Code/functionality
  - ☐ Physical level
  - ☐ Model level
  - ☐ Software component
  - ☐ Runnable
  - ☐ C Function
  - ☐ Code level
  - ☐ Other: ____________________
- ☐ Data
- ☐ Other: ____________________



Which timing parameter is relevant?

Please mark with a cross:

- ☐ CET (Core Execution Time)
- ☐ DT (Delta Time)
- ☐ RT (Response Time)
- ☐ GET (Gross Execution Time)
- ☐ ST (Slack Time)
- ☐ NST (Net Slack Time)
- ☐ IPT (Initial Pending Time)
- ☐ JIT (Jitter)
- ☐ PRE (Preemption Time)
- ☐ Execution order (e.g. related to runnables)
- ☐ Data age (time between read and write or between send and receive/usage of the received data)
- ☐ Boot/init: from ____________ to ____________________
- ☐ Not applicable
- ☐ Other: ____________________

What type of value is it?

Please mark with a cross:

- ☐ Minimal allowed value
- ☐ Maximal allowed value
- ☐ Average value
- ☐ Not applicable
- ☐ Other: ____________________

What is the value (e.g. in µs or ms)?

Limit or required value: ____________________

or

Execution order: ____________________

Step 2a: Initial design using scheduling simulation

- Configure operating systems on all cores
  - Create TASKs
  - Create ISRs
  - Optionally create and assign runnables
- Configure activation patterns
  - When/how are ISRs triggered and TASKs activated?
- Provide BCETs and WCETs to be used in simulation
  - Budgets
  - Measurements from previous generations of the SW

Step 2a: configure scheduling simulation

Step 2b: Initial design using scheduling simulation

- Explore the scheduling
  - Get to know the timing of your ECU early!
  - Optimize
  - Compare different concepts

Step 2b: run and use scheduling simulation

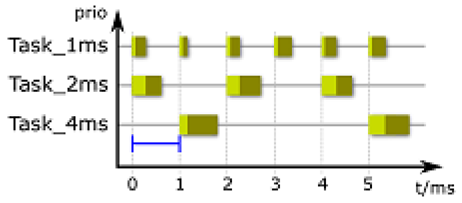
- Find a solid scheduling
  - The WCETs used in the simulation for TASKs, ISRs and runnables can then function as timing requirements. In other words: if these are not exceeded, the scheduling, the timing is safe.

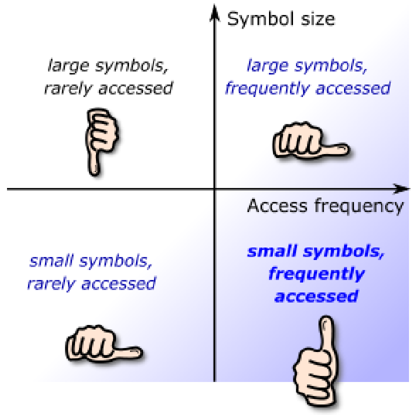
Step 3: Look at the **real** world

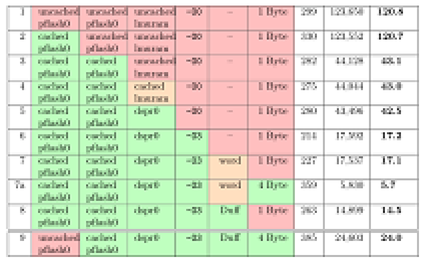
- Do not rely on model based timing analysis or scheduling simulation only!
- Models and simulations are **NOT** the reality! In the end you build real products, not virtual products.

Step 4: Optimize timing (top down)

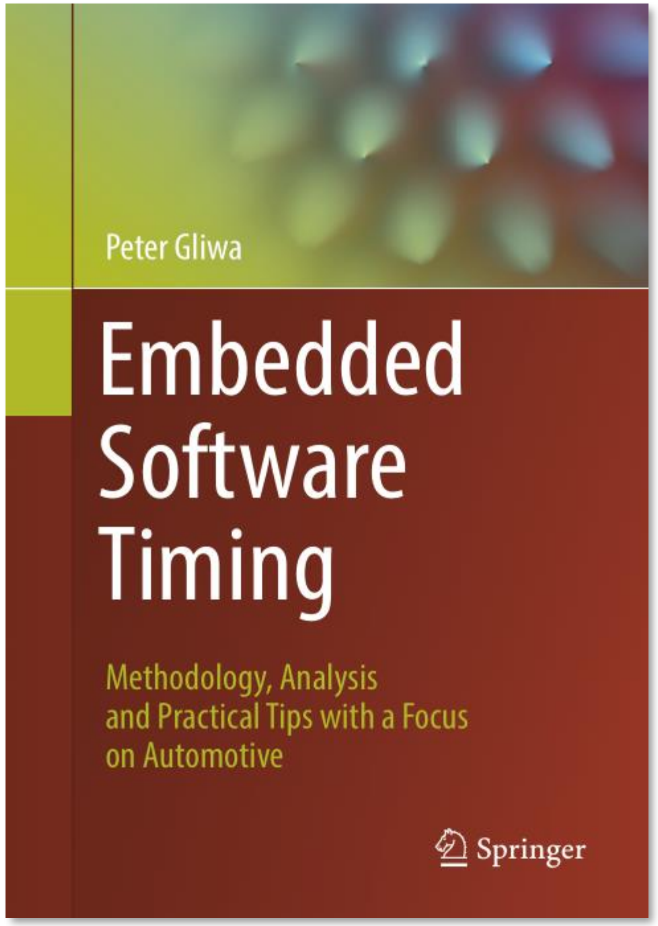
- Scheduling level
- Memory usage
- Code level




Symbol size
large symbols, rarely accessed
large symbols, frequently accessed
Access frequency
small symbols, rarely accessed
small symbols, frequently accessed



1	uncached pflash0	uncached pflash0	uncached lmuram	-O0	–	1 Byte	299	121.650	120.8
2	cached pflash0	uncached pflash0	uncached lmuram	-O0	–	1 Byte	330	121.552	120.7
3	cached pflash0	cached pflash0	uncached lmuram	-O0	–	1 Byte	282	44.129	43.1
4	cached pflash0	cached pflash0	cached lmuram	-O0	–	1 Byte	275	44.044	43.0
5	cached pflash0	cached pflash0	dspr0	-O0	–	1 Byte	290	43.496	42.5
6	cached pflash0	cached pflash0	dspr0	-O3	–	1 Byte	214	17.592	17.2
7	cached pflash0	cached pflash0	dspr0	-O3	word	1 Byte	227	17.537	17.1
7a	cached pflash0	cached pflash0	dspr0	-O3	word	4 Byte	359	5.830	5.7
8	cached pflash0	cached pflash0	dspr0	-O3	Duff	1 Byte	263	14.899	14.5
9	uncached pflash0	cached pflash0	dspr0	-O3	Duff	4 Byte	285	24.603	24.0



Peter Gliwa

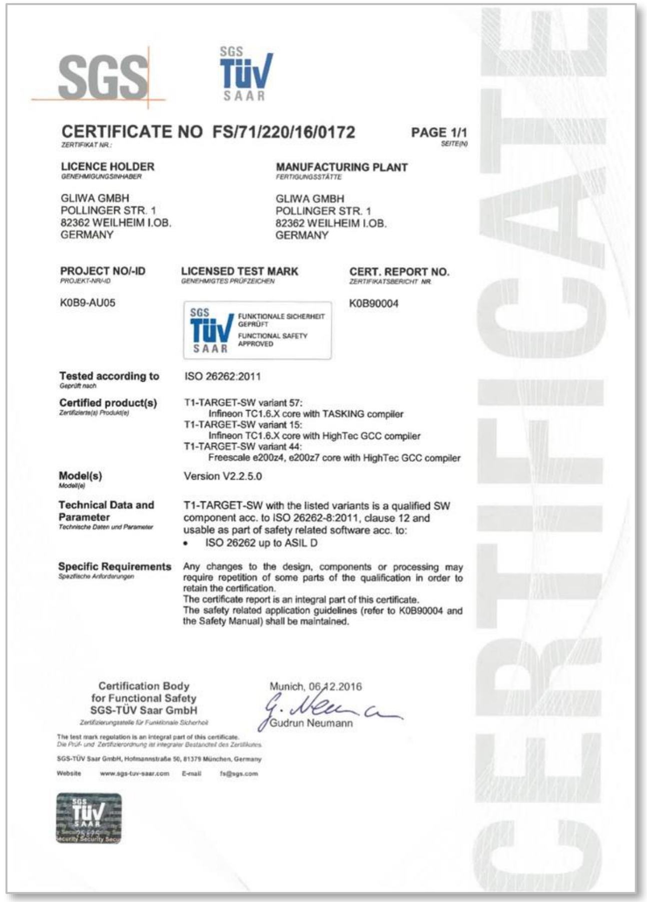
Embedded Software Timing

Methodology, Analysis and Practical Tips with a Focus on Automotive

Springer

Step 5: Verify timing

- Verify timing against requirements
  - E.g. timing parameters CET, RT, DT, CPU-load using measurement
  - scheduling using tracing
- Use **automated (!)** timing tests for all of this!
- Verify
  - In the lab
  - On the HIL
  - In the final environment (the machine, the car, etc.)
- Optionally supervise timing in final product (T1 is **ISO 26262 ASIL D** certified)



SGS

SGS TÜV SAAR

CERTIFICATE NO FS/71/220/16/0172 — **PAGE 1/1**
ZERTIFIKAT NR.: — *SEITE(N)*

LICENCE HOLDER *GENEHMIGUNGSINHABER*	**MANUFACTURING PLANT** *FERTIGUNGSSTÄTTE*
GLIWA GMBH POLLINGER STR. 1 82362 WEILHEIM I.OB. GERMANY	GLIWA GMBH POLLINGER STR. 1 82362 WEILHEIM I.OB. GERMANY

PROJECT NO/-ID *PROJEKT-NR/-ID*	**LICENSED TEST MARK** *GENEHMIGTES PRÜFZEICHEN*	**CERT. REPORT NO.** *ZERTIFIKATSBERICHT NR.*
K0B9-AU05	SGS TÜV SAAR FUNKTIONALE SICHERHEIT GEPRÜFT FUNCTIONAL SAFETY APPROVED	K0B90004

Tested according to *Geprüft nach*	ISO 26262:2011
Certified product(s) *Zertifizierte(s) Produkt(e)*	T1-TARGET-SW variant 57: Infineon TC1.6.X core with TASKING compiler T1-TARGET-SW variant 15: Infineon TC1.6.X core with HighTec GCC compiler T1-TARGET-SW variant 44: Freescale e200z4, e200z7 core with HighTec GCC compiler
Model(s) *Modell(e)*	Version V2.2.5.0
Technical Data and Parameter *Technische Daten und Parameter*	T1-TARGET-SW with the listed variants is a qualified SW component acc. to ISO 26262-8:2011, clause 12 and usable as part of safety related software acc. to: • ISO 26262 up to ASIL D
Specific Requirements *Spezifische Anforderungen*	Any changes to the design, components or processing may require repetition of some parts of the qualification in order to retain the certification. The certificate report is an integral part of this certificate. The safety related application guidelines (refer to K0B90004 and the Safety Manual) shall be maintained.

Certification Body
for Functional Safety
SGS-TÜV Saar GmbH
Zertifizierungsstelle für Funktionale Sicherheit

Munich, 06.12.2016
Gudrun Neumann

The test mark regulation is an integral part of this certificate.
Die Prüf- und Zertifizierordnung ist integraler Bestandteil des Zertifikates.

SGS-TÜV Saar GmbH, Hofmannstraße 50, 81379 München, Germany

Website www.sgs-tuv-saar.com E-mail fs@sgs.com

SGS TÜV SAAR

CERTIFICATE

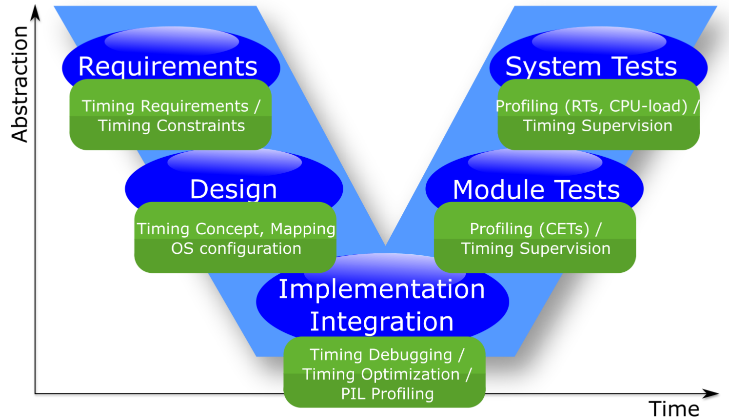
Summary

Summary

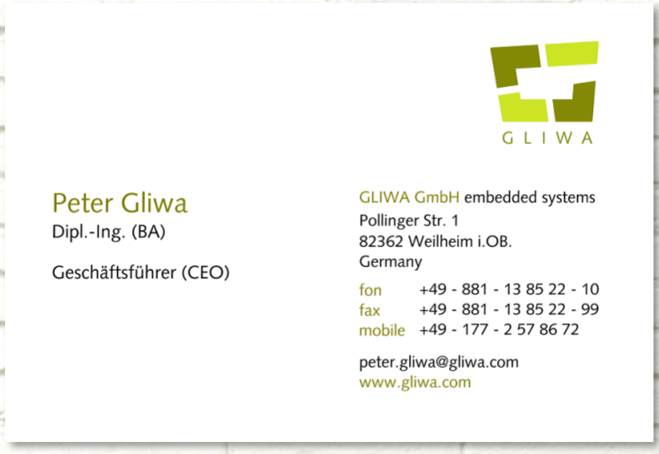
- A suitable timing methodology is important.

- Define sensible requirements
  More is not naturally *better*. Example: OEM specified WCET requirements which lead to poorer quality
  - ASIL-D project adds a degradation concept (and thus more complexity) just to fulfill WCET requirements.
  - Addressing too many timing aspects binds resources and moves the focus away from real timing issues.

- Do not forget the *real* world – in the end you are building *real* embedded systems.


Abstraction
Requirements
Timing Requirements / Timing Constraints
Design
Timing Concept, Mapping OS configuration
Implementation Integration
Timing Debugging / Timing Optimization / PIL Profiling
Module Tests
Profiling (CETs) / Timing Supervision
System Tests
Profiling (RTs, CPU-load) / Timing Supervision
Time

Source: BSE-Galerie



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