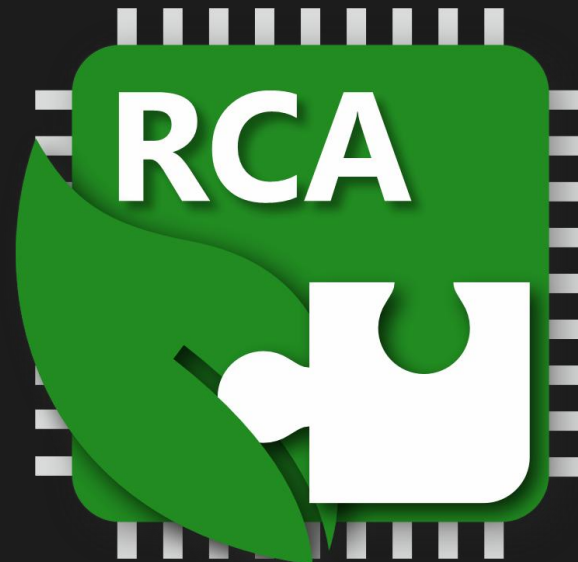


# Automatic Assessment and Real-Time Feedback in Processor Design Education using HADES-V

**Tobias Scheipel**  
tobias.scheipel@tugraz.at

Reconfigurable Computer Architectures  
Institute of Technical Informatics  
Graz University of Technology



This slide set is licensed under:  
CC BY-SA 4.0 International  
<https://creativecommons.org/licenses/by-sa/4.0/>  
Tobias Scheipel, TU Graz 2025  
<https://www.scheipel.com>



# Outline

**Intro of  
HADES-V  
&  
Methodology**

**Motivation  
of an  
Auto  
Assessment**

**Test  
System  
Overview &  
in Action**

**Conclusion  
&  
Outlook**

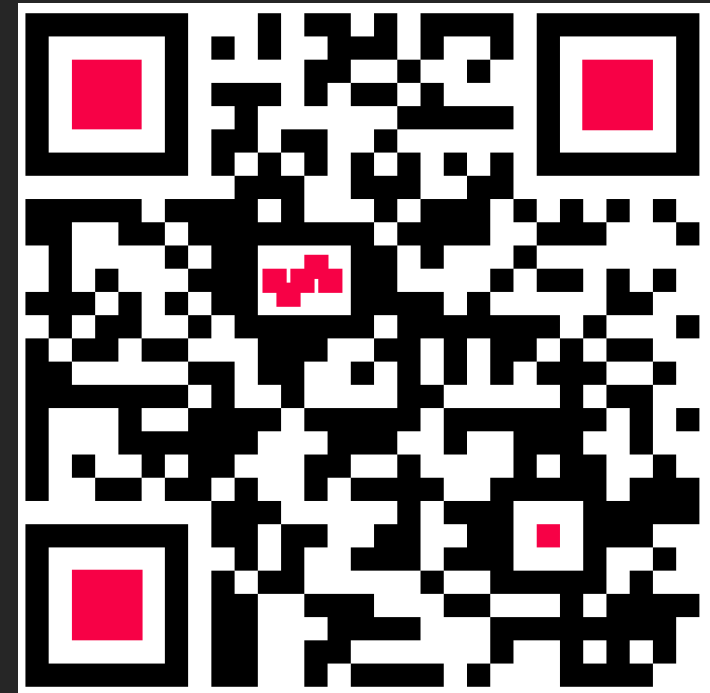
# Why Teach Processor Design?

- › processor design underpins modern electrical engineering
  - traditional courses often very abstract
- › bridge gap between theory and real hardware implementations
- › modular, practical approaches help learners
  - › grasping complex concepts
  - › and build confidence along the way



## What is HADES-V?

- HADES-V is an open educational resource (OER) for teaching microcontroller/processor design
- students build a pipelined 32-bit RISC-V microcontroller using SystemVerilog and FPGA tools
- materials:
  - instruction guide [1]
  - open-source code template [2]
  - scientific paper published @ RISC-V Summit [3]



## What is RISC-V?

- › free & open standard instruction set architecture (ISA)
- › modular & extensible design → combination of base instruction set with ratified and custom extensions
- › created at UC Berkeley in 2010 → now maintained by non-profit RISC-V International
- › driving innovation throughout computing sectors by removing licence barriers and encouraging customization
- › already started being adopted by major silicon vendors

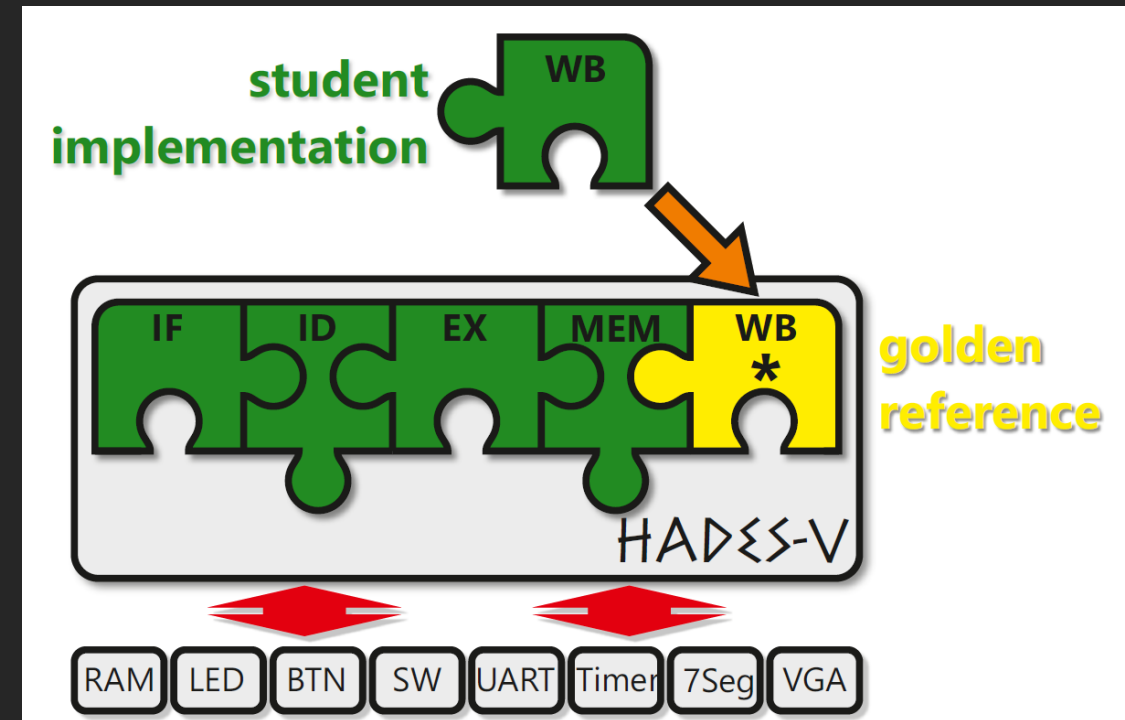


## Tools & Resources

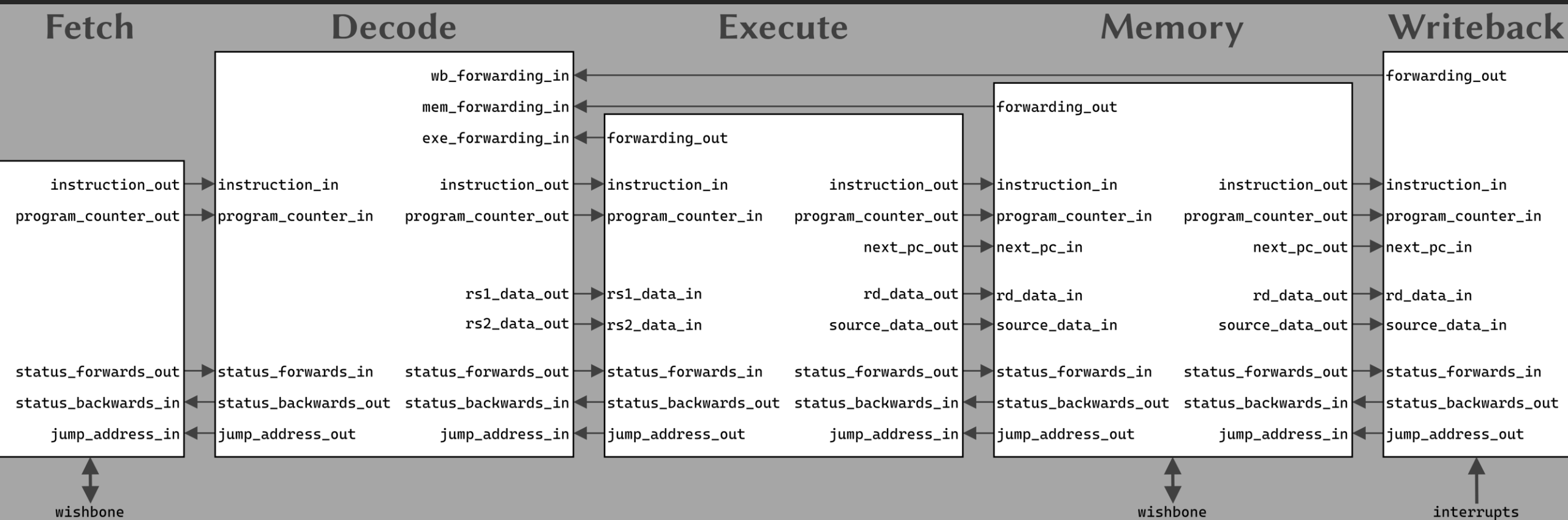
- › **hardware description & simulation:** SystemVerilog, Verilator, GTKWave
- › **synthesis & deployment:** AMD Vivado for FPGA synthesis → Basys3 board
- › **software support:** RISC-V standard toolchain for programming
- › **open licences:** everything except Vivado is open:
  - › **material:** Instruction Guide: CC BY 4.0; Code Template: MIT, RISC-V ISA: CC BY 4.0,
  - › **tools:** Verilator: LGPL 3.0; GTKWave: GPL 2, RISC-V GCC: GPL 2
  - › future: open replacement for Vivado envisioned

# HADES-V Puzzle Methodology

- didactic methodology: jigsaw puzzle approach → every pipeline stage implemented separately by the students (from scratch)
- pre-compiled golden references can be used to validate each module → providing immediate feedback without revealing the solution
- step-by-step process breaks down complexity and encourages incremental learning



# 7 HADES-V Puzzle Methodology



## HADSS-V Learning Objectives

after completing HADSS-V, students will be able to

- design a complete, modular RISC-V based microcontroller unit using SystemVerilog and development environments.
- analyze the functionality and efficiency of microcontroller designs using verification, synthesis and debugging tools.
- implement different pipeline stages of a microcontroller using processor architecture knowledge and hardware description languages.
- independently design innovative extensions for microcontrollers, combining hardware and software.
- explain the execution of software in processors and evaluate their interaction with the hardware.
- communicate technical design decisions in presentations and subsequently respond to feedback from colleagues.

# HADES-V Assessment Methodology

- › course type: laboratory practical, 6 ECTS, ~30 pax
- › individual work / collaboration in the lab ok
- › attendance voluntarily (in principle)
- › three pillars of assessment:
  - › submissions
  - › presentation
  - › mini-project (last submission)
- › 8 submissions in total → 75 points
- › 2-3 submission presentations → 25 points
- › 100 points total



## Why Automate Assessment?

- manually grading hardware designs is **cumbersome, error prone** and **time consuming** → fairness cannot be guaranteed easily
  - students need quick, detailed feedback to move on
  - a scalable, automatic test system ensures fairness, consistency, and real-time progress tracking
  - enables remote learning all over the world → intercultural aspect
- introducing **PERSEPHONE**, the **HADES-V** test system

# The PERSAPHON Test System

- Python-based framework that automatically runs testcases
- student code submission via Git
- extensive set of SystemVerilog, C and Assembly test cases (over 100 per module)
- provides pass/fail outputs and detailed performance insights
- tests aligned with learning objectives
- all gathered in a Markdown-formatted browser-based overview

## The ΠΕΡΣΕΦΟΝΕΣ Test System – for Students

- › helps finding errors & identifying edge cases in the submission
- › transparent communication of current points
- › gamification: increases motivation throughout the course
- › alignment with learning objectives → understand *why*?
- › encourages self-learning and iterative improvement
- › increases feeling of submission ownership → “this is mine!”

## The ΠΕΡΣΕΦΩΝΕΣ Test System – for Educators

- › get an overview of the current state and progress of the students
  - › streamline grading w.r.t. fairness, consistency, reproducibility, etc.
  - › find outliers and irregularities fast (cheating avoidance)
  - › compare different solutions easily
  - › retain control over closed test framework → no solutions leaked!
- ultimately, speed up the grading process!

# Fine-tuning Assessment for Fairness and Transparency

- to mimic manual assessment, we categorized all test cases into *minimal*, *regular*, *maximal*, and *optional* point groups
- points are fine-tuned with genetic algorithms to ensure fairness and feasibility of the auto assessment → tight fit between manual and automatic
- test balloon in 2025: no more manual grading! → worked due to transparency
- cheating is discouraged by plagiarism checks and mandatory presentation sessions where students must explain submissions

	MIN	REG	MAX	OPT	SUM
errors	13	1	9	0	23
calls	242	20	104	0	366
fails	12	1	8	0	21
max	6	6	4	0	16
points	5.70	5.70	3.69	0.00	15.09

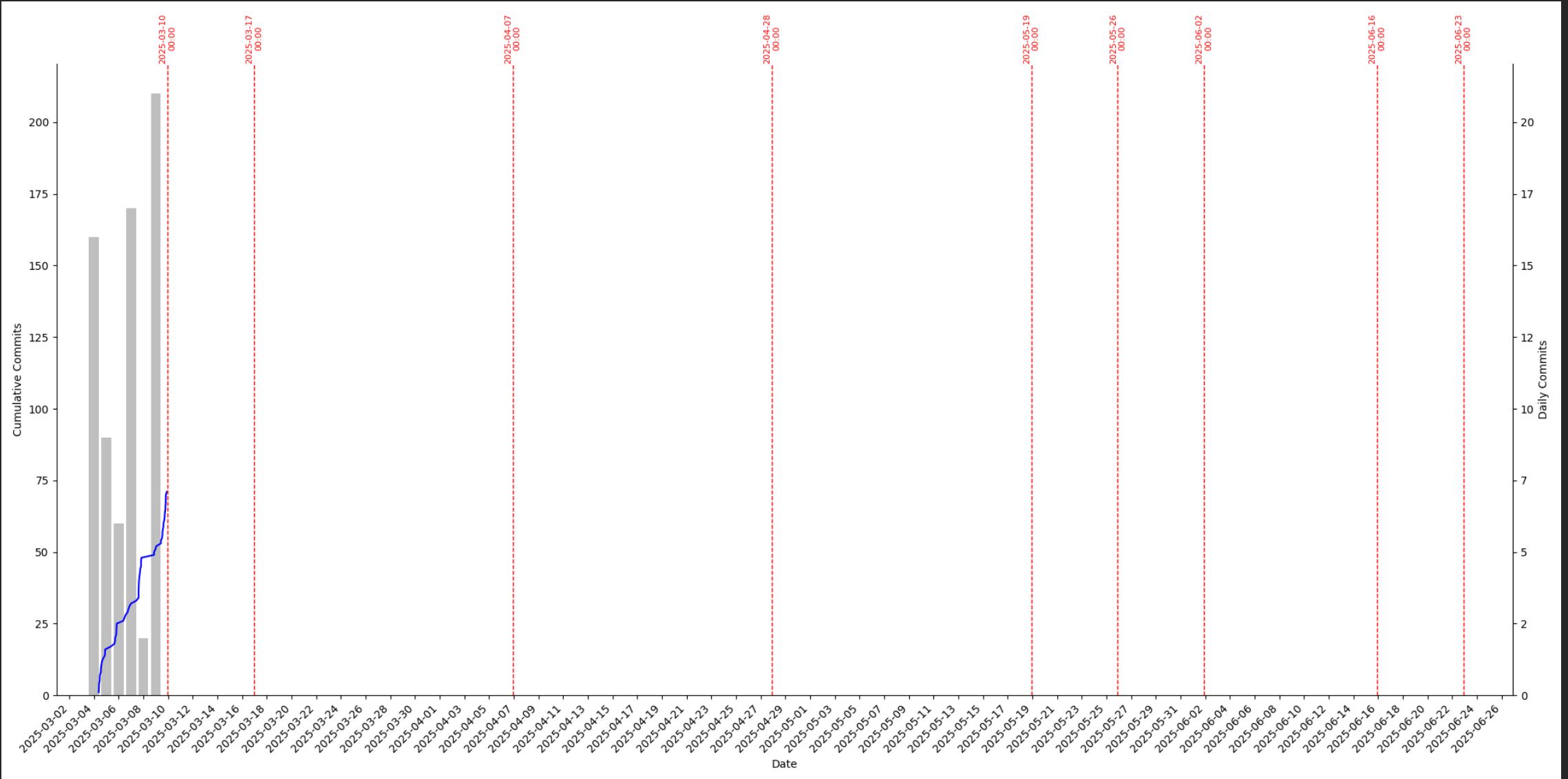
# Outtakes & Screenshots from last Iteration

- › git commit progress over the semester
- › student detailed list of test cases
- › student current grading table
- › mapping to our moodle instance
- › progress of a student over time
- › differences of what a student sees vs what we see

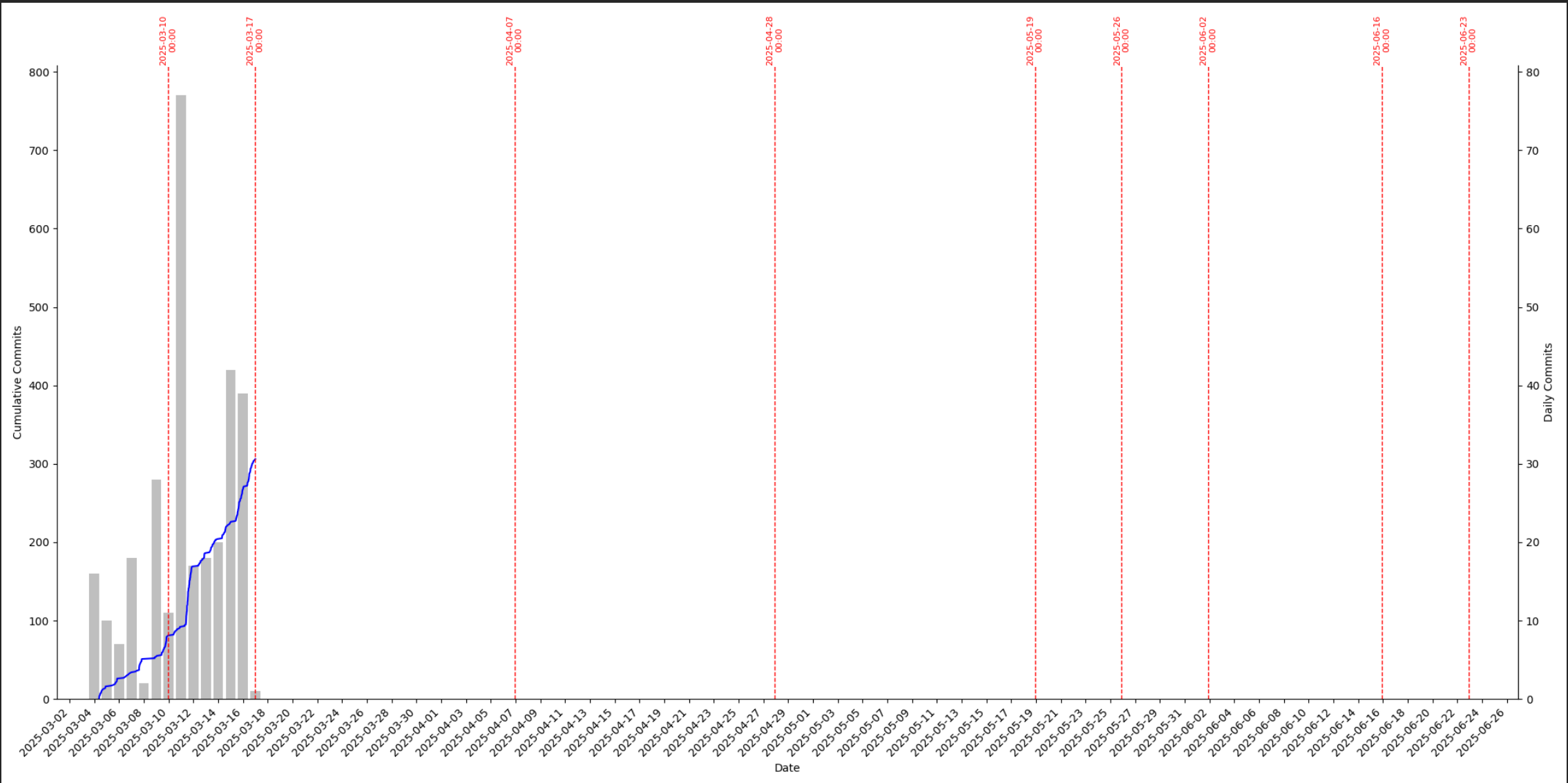


→ **Live Demo**

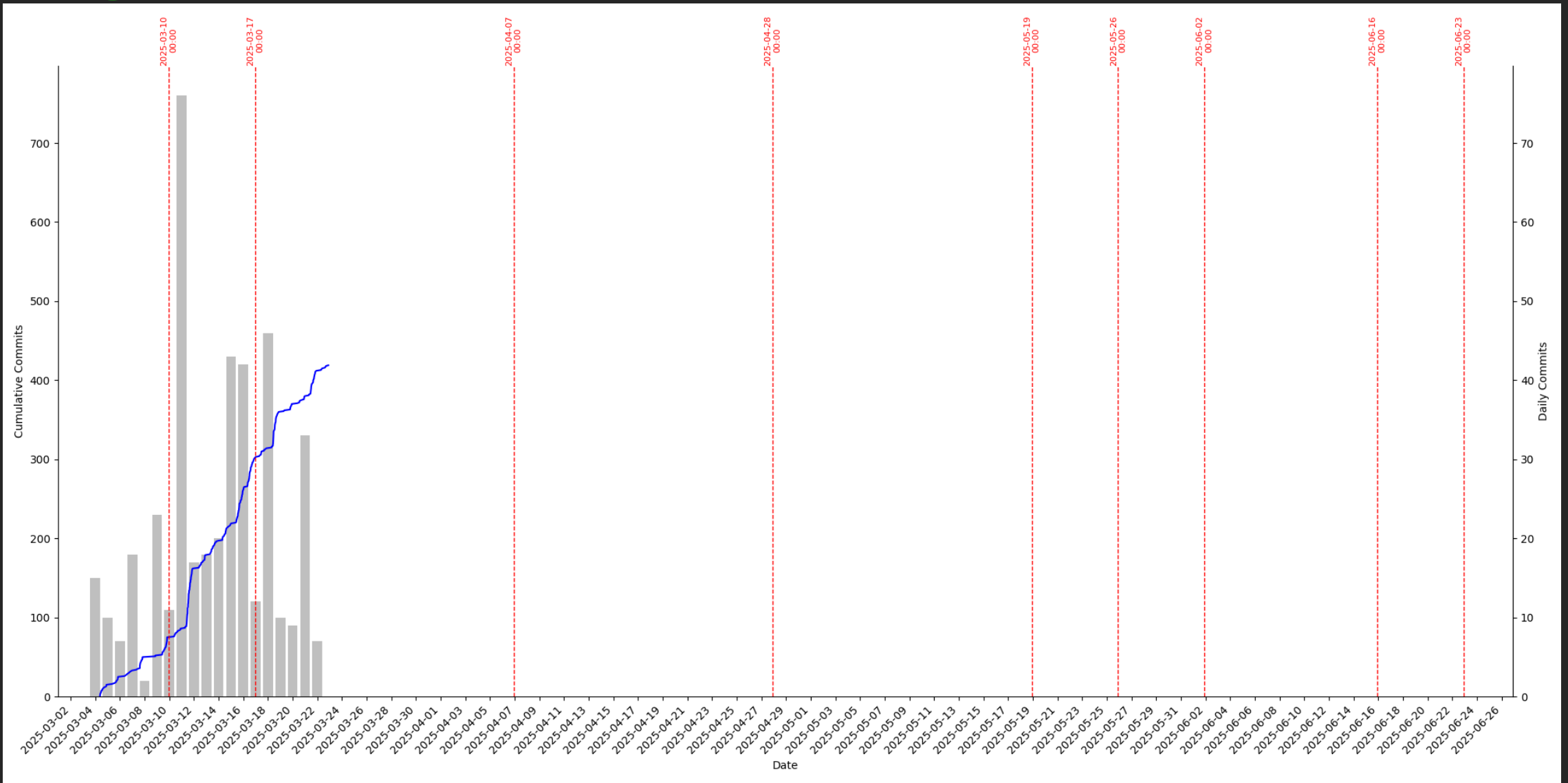
# Progress of Commits



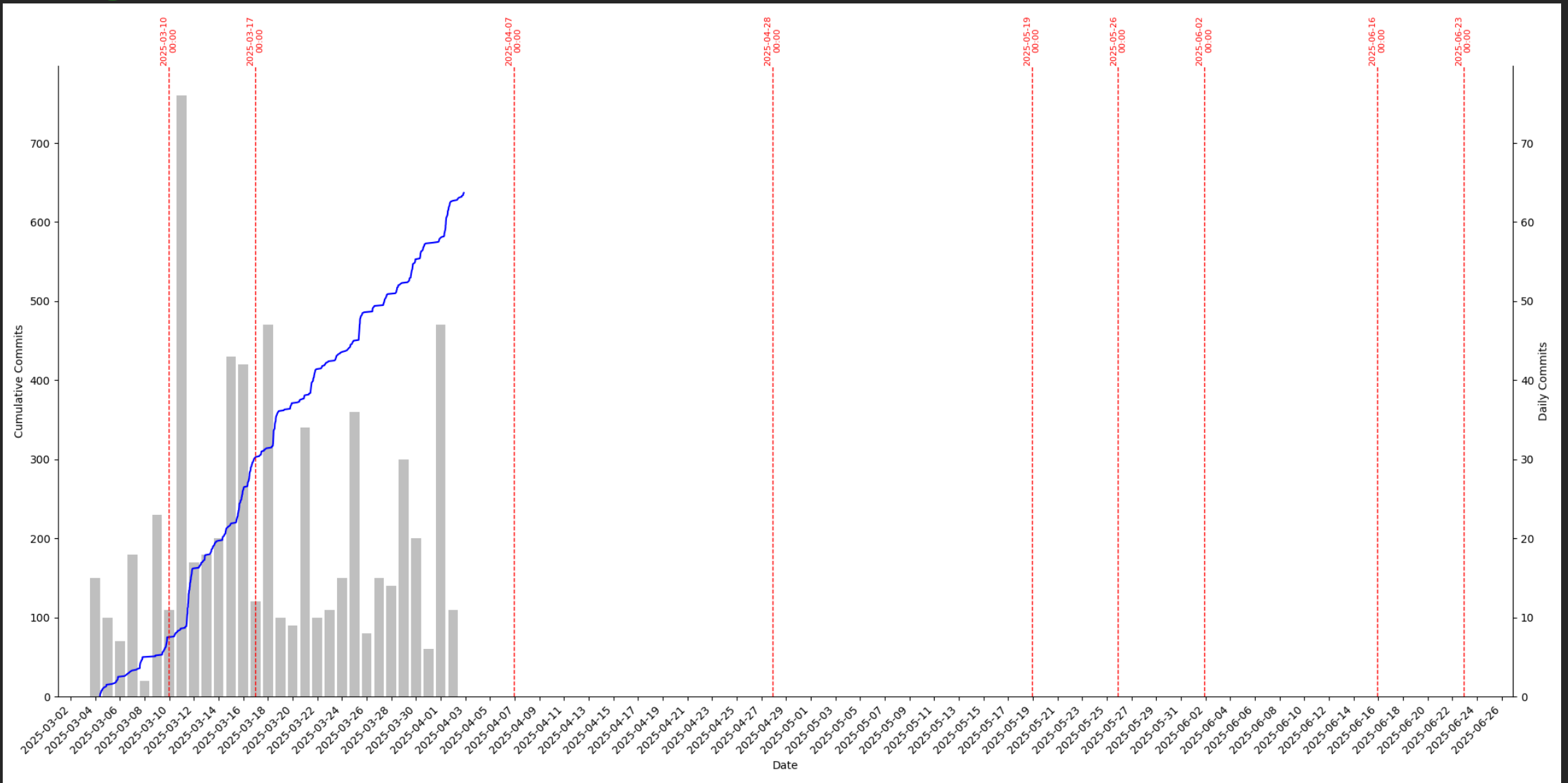
# Progress of Commits



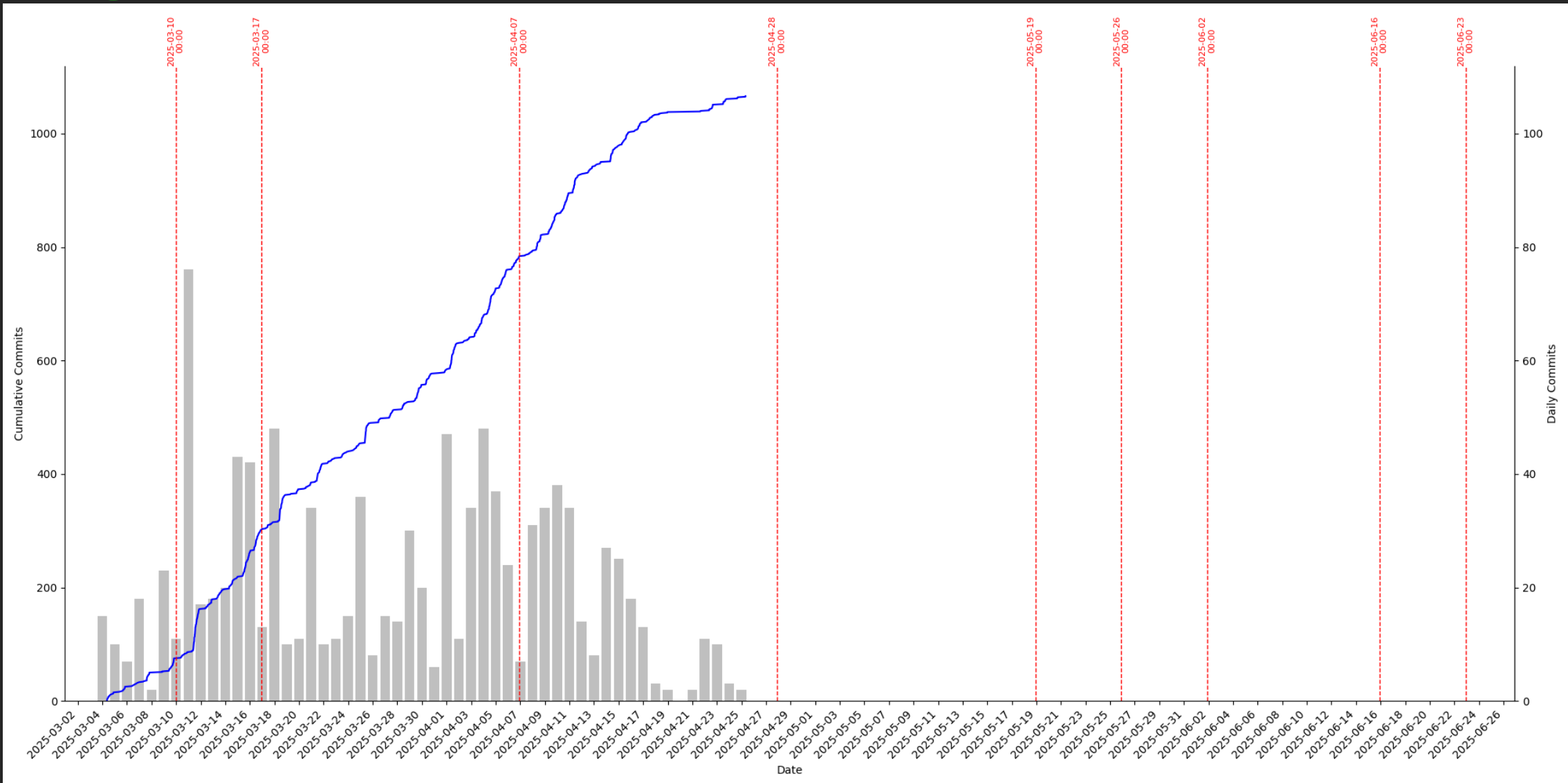
# Progress of Commits



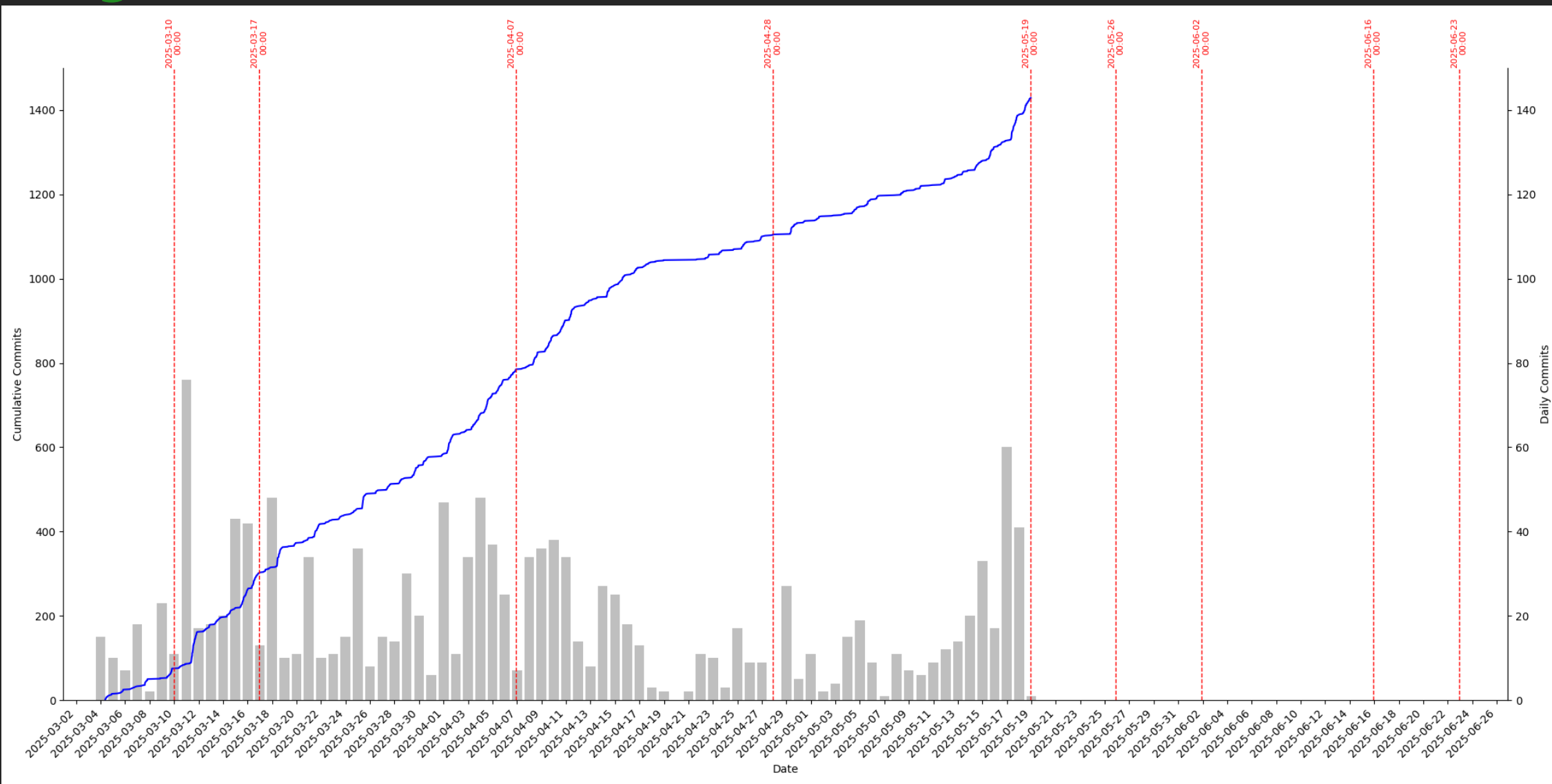
# Progress of Commits



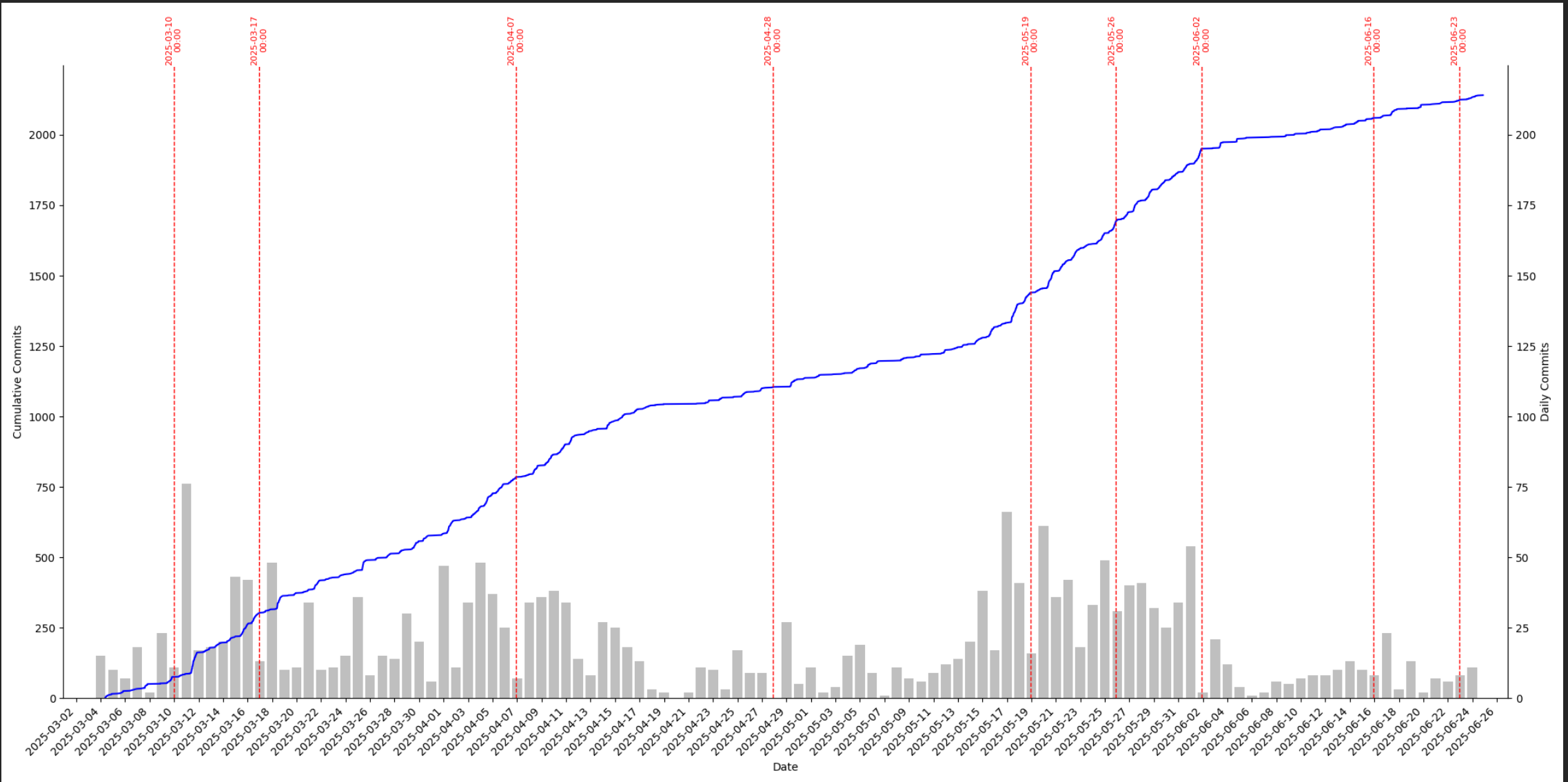
# Progress of Commits



# Progress of Commits



# Progress of Commits



# Student Detailed List View

- > only visible to supervisors
- > fast overview of all relevant test cases
- > collapsing paragraphs for details
- > links to the student's submission

Student: mdlab\_....

- ▶  asm\_forwarding.log
- ▶  ! asm\_ops.log
- ▶  ! asm\_trap.log
- ▶ 8.00 / 8: sv\_test\_fetch\_stage.log
- ▶ 4.00 / 4: sv\_test\_decode\_stage.log
- ▶ 4.00 / 4: sv\_test\_register\_file.log
- ▶ 4.00 / 4: sv\_test\_instruction\_decoder.log
- ▶ 10.00 / 10: sv\_test\_execute\_stage.log
- ▶ 10.00 / 10: sv\_test\_memory\_stage.log
- ▼ 15.09 / 16: sv\_test\_writeback\_stage.log

Logfile: sv\_test\_writeback\_stage.log

	MIN	REG	MAX	OPT	SUM
errors	13	1	9	0	23
calls	242	20	104	0	366
fails	12	1	8	0	21
max	6	6	4	0	16
points	5.70	5.70	3.69	0.00	15.09



24

# Grading Table

- > only visible to supervisors
- > overview of all students and all testcases
- > color indicators to speedup reading

	Points / 16	✗ / 366	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
mdlab_StudA_01	8.32	165	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudB_02	15.76	8	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudC_03	15.7	366	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
mdlab_StudD_04	16	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudE_05	1.99	309	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✗
mdlab_StudF_06	13.46	56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudG_07	15.7	366	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
mdlab_StudH_08	15.7	366	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
mdlab_StudI_09	16	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudJ_10	15.7	366	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
mdlab_StudK_11	16	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudL_12	16	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudM_13	14.64	37	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudN_14	15.84	5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudO_15	15.7	366	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
mdlab_StudP_16	15.7	366	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
mdlab_StudQ_17	16	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudR_18	12.37	101	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudS_19	16	0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mdlab_StudT_20	15.7	366	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
mdlab_StudU_21	15.7	366	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Total ✓	0	0	12	12	12	12	12	12	12	12	12	12	13	13	13	13	13	13	12
Total ✗	0	0	9	9	9	9	9	9	9	9	9	9	8	8	8	8	8	8	9



# Moodle Mapping

- points published via moodle instance
- every submission and presentation is reflected here
- transparent communication to the students

Name der Bewertung	Bewertungskategorie	Bewertung	Bereich
MANUELLER ASPEKT CPU	Exercises	3,00	0,00 - 3,00
MANUELLER ASPEKT Instruction Fetch Stage	Exercises	8,00	0,00 - 8,00
MANUELLER ASPEKT Decode Stage and Register File	Exercises	8,00	0,00 - 8,00
MANUELLER ASPEKT Instruction Decoder	Exercises	4,00	0,00 - 4,00
MANUELLER ASPEKT Execute Stage	Exercises	10,00	0,00 - 10,00
MANUELLER ASPEKT Memory Stage	Exercises	10,00	0,00 - 10,00
MANUELLER ASPEKT Writeback Stage	Exercises	15,90	0,00 - 16,00
MANUELLER ASPEKT Synthesize	Exercises	0,00	0,00 - 6,00
MANUELLER ASPEKT Final Exercise	Exercises	0,00	0,00 - 10,00
<b>GESAMTERGEBNIS</b> <b>Exercises gesamt</b>	Exercises	58,90	0,00 - 75,00
MANUELLER ASPEKT 1st	Presentation	25,00	0,00 - 25,00
MANUELLER ASPEKT 2nd	Presentation		0,00 - 25,00
MANUELLER ASPEKT 3rd	Presentation		0,00 - 25,00
MANUELLER ASPEKT 4th	Presentation		0,00 - 25,00
<b>GESAMTERGEBNIS</b> <b>Presentation gesamt</b>	Presentation	25,00	0,00 - 25,00
<b>GESAMTERGEBNIS</b> <b>Kurs gesamt</b>	Microcontroller Design, Laboratory (SS) [448029]	83,90	0,00 - 100,00

# Progress of a Student over Time

## Testcase Results

Repository:

Test Run: 27.05.2025 16:00

Test Deadline: 12.06.2025 00:00

### Tested Commit Information

Date: 27.05.2025 08:16

Hash: 2c20cb0

Message: adding writing of CSR registers

Committer Email:

## Module Under Test: Writeback Stage

### ▼ Details for the Writeback Stage

Points: 8.32 / 16

Summary: Some test(s) failed! (# Errors: 201, # Testcases: 366)

	MIN	REG	MAX	OPT	SUM
errors	112	10	79	0	201
calls	242	20	104	0	366
fails	94	10	61	0	165
max	6	6	4	0	16
points	3.67	3.00	1.65	0.00	8.32

## SIMPLE INSTRUCTIONS

### FENCE\_I

Test input: FENCE\_I with status\_forwards\_in = VALID and external/timer interrupt = 0/0

Signal	Is Value	Expected Value	Type
status_backwards_out	0	2	Error
jump_address_backwards_out	0x00000000	0x000400a8	Error

Test input: FENCE\_I with status\_forwards\_in = VALID and external/timer interrupt = 0/0

# Progress of a Student over Time

## Testcase Results

Repository:  
 Test Run: 01.06.2025 04:00  
 Test Deadline: 12.06.2025 00:00

### Tested Commit Information

Date: 30.05.2025 10:22  
 Hash: 0e0779c  
 Message: bugfix  
 Committer Email:

## Module Under Test: Writeback Stage

### • Details for the Writeback Stage

Points: 15.00 / 16

Summary: Some test(s) failed! (8 Errors: 23, # Testcases: 366)

	MISS	REQ	MAX	OPT	SUM
errors	13	1	9	0	23
calls	242	20	104	0	366
fails	12	1	8	0	21
max	8	8	4	0	16
points	5.70	5.70	5.69	5.00	15.09

## CSR-operations

### MEPC - set LSBs = 0!

Test Input: CSRRO with status\_forwards\_in = VALID and external/timer interrupt = 0/0, csr = MEPC

Signal	Is Value	Expected Value	Type
forwarding_out_data	0x0000000f	0x00000000	Error

Test Input: CSRRO with status\_forwards\_in = VALID and external/timer interrupt = 0/0, csr = MEPC

Signal	Is Value	Expected Value	Type
--------	----------	----------------	------

# Progress of a Student over Time

### Testcase Results

Repository:  
 Test Run: 27.05.2025 16:00  
 Test Deadline: 12.06.2025 00:00

### Tested Commit Information

Date: 27.05.2025 08:16  
 Hash: 2c20cb0  
 Message: adding writing of CSR registers  
 Committer Email:

### Module Under Test: Writeback Stage

▼ Details for the Writeback Stage

Points: 8.32 / 16

Summary: Some test(s) failed! (# Errors: 201, # Testcases: 366)

	MIN	REG	MAX	OPT	SUM
errors	112	10	79	0	201
calls	242	20	104	0	366
fails	94	10	61	0	165
max	6	6	4	0	16
points	3.67	3.00	1.65	0.00	8.32

### SIMPLE INSTRUCTIONS

#### FENCE\_I

Test input: FENCE\_I with status\_forwards\_in = VALID and external/timer interrupt = 0/0

Signal	Is Value	Expected Value	Type
status_backwards_out	0	2	Error

### Testcase Results

Repository:  
 Test Run: 01.06.2025 04:00  
 Test Deadline: 12.06.2025 00:00

### Tested Commit Information

Date: 30.05.2025 16:22  
 Hash: 0ed515c  
 Message: bugfix  
 Committer Email:

### Module Under Test: Writeback Stage

▼ Details for the Writeback Stage

Points: 15.09 / 16

Summary: Some test(s) failed! (# Errors: 23, # Testcases: 366)

	MIN	REG	MAX	OPT	SUM
errors	13	1	9	0	23
calls	242	20	104	0	366
fails	12	1	8	0	21
max	6	6	4	0	16
points	5.70	5.70	3.69	0.00	15.09

### CSR-operations

#### MEPC - set LSBs = 0!

Test input: CSRRCI with status\_forwards\_in = VALID and external/timer interrupt = 0/0, csr = MEPC

Signal	Is Value	Expected Value	Type
forwarding_out.data	0x0000001f	0x0000001c	Error

# Student View vs. Assessment View

- no points overview table
- only total points
- no output if reference solution is still being used
- mistakes/errors list only displayed if  $\text{points} > 1/3 \text{ max points}$

## Testcase Results

Repository:  
Test Run: 27.05.2025 16:00  
Test Deadline: 12.06.2025 00:00

### Tested Commit Information

Date: 27.05.2025 08:16  
Hash: 2c20cb0  
Message: adding writing of CSR registers  
Committer Email:

### Module Under Test: Writeback Stage

#### Details for the Writeback Stage

Points: 8.32 / 16

Summary: Some test(s) failed! (# Errors: 201, # Testcases: 366)

	MIN	REG	MAX	OPT	SUM
errors	112	10	79	0	201
calls	242	20	101	0	366
fails	94	10	61	0	165
max	6	6	4	0	16
points	3.67	3.00	1.65	0.00	8.32

### SIMPLE INSTRUCTIONS

#### FENCE\_I

Test input: FENCE\_I with status\_forwards\_in = VALID and external/timer interrupt = 0/0

Signal	Is Value	Expected Value	Type
status_backwards_out	0	2	Error

# Impact & Student Feedback

- overall submission quality, motivation, feeling of “ownership” increased!
- frustration when debugging submissions decreased!
- overall engagement and submission git commits skyrocketed!

→ success story to continue and improve!

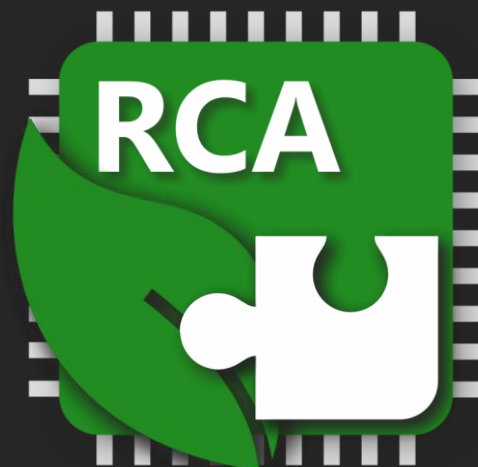
# Outlook & Improvements

- › include last assignment into test system (synthesis)
- › test even more often (best after every commit)
- › include more offline testing tools without revealing the reference solution
- › hardware-in-the-loop testing on our server
- › opening the course internationally → RVI community challenge
- › ...

→ stay tuned!

# Thank you for your Attention!

## Questions?



### References:

- [1] <https://repository.tugraz.at/oer/nytm4-grv34>
- [2] <https://github.com/tscheipel/HaDes-V>
- [3] T. Scheipel, D. Beikircher, F. Riedl, Learning by Puzzling: A Modular Approach to RISC-V Processor Design Education, RISC-V Summit Europe 2025, <https://doi.org/10.13140/RG.2.2.22964.16006>

This slide set is licensed under:  
CC BY-SA 4.0 International

<https://creativecommons.org/licenses/by-sa/4.0/>

Tobias Scheipel, TU Graz 2025

<https://www.scheipel.com>



# License



This slide set is licensed under:

CC BY-SA 4.0 International

<https://creativecommons.org/licenses/by-sa/4.0/>

Tobias Scheipel, TU Graz 2025

<https://www.scheipel.com>

## Notes on Licensing:

- This license applies to all content created by the author and not explicitly labeled as external material.
- External materials in this slide set, such as images or data from third-party sources, retain their respective licenses.
- The slide set uses pictures from <https://cocomaterial.com> licensed under CC 0.