



Hochschule **RheinMain**

LLM ASSISTED NO-CODE HMI DEVELOPMENT FOR SAFETY-CRITICAL SYSTEMS

Insights of a Short Impirical Study

Prof. Dr. Matthias Harter
November 2023



Hochschule **RheinMain**

A (VERY) SHORT RÉSUMÉ

Contact information at the end of this presentation...

SHORT RÉSUMÉ

Some call it CV...



Hochschule RheinMain

Name

Prof. Dr. Matthias Harter

Fields of interest / profession

- Patents and IP
- AI, AGI and humanity
- ASICs, Circuits and Systems
- Aviation, Simulators

since 07/11

Professor for Embedded
Systems and Microcomputers
Hochschule RheinMain
University of Applied Sciences

10/12 – 09/18 Head of the Department of
Electrical Engineering and
Information Technology

10/17 – 10/23 Head (founder) of new study
program „Electrical and
Aviation Engineering“





Hochschule RheinMain

STATE OF THE ART: EMBEDDED SOFTWARE FOR SAFETY-CRITICAL APPLICATIONS

Examples from Aviation Engineering

A320 COCKPIT

Safety-Critical Embedded SW development today



Hochschule RheinMain



CESSNA 172 (4 SEATS)

Replacement of analog instruments



DEVELOPMENT OF COCKPIT INSTRUMENTS

How is it done today?



Hochschule RheinMain

Scenario (example):

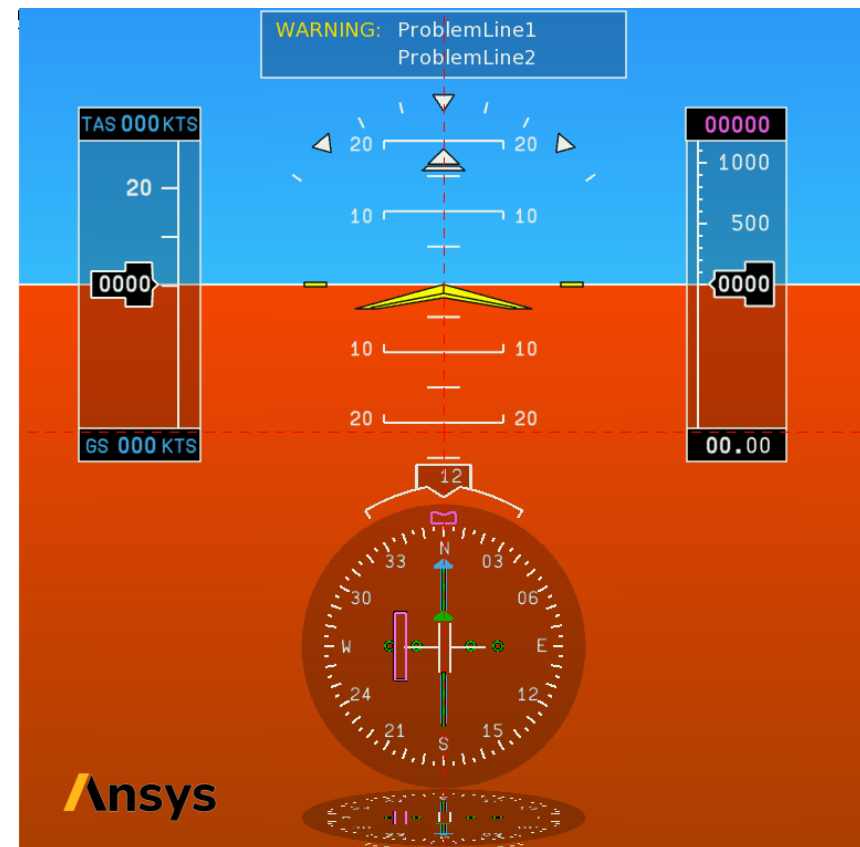
Artificial horizon used as Primary Flight Display (PFD) in an aircraft's cockpit.

State of the art:

Automation of the development process with **certified** tools, e.g., Ansys SCADE

- Visual part specified as graphical models based on OpenGL primitives
- Usage of pre-defined widgets (fast!)
- Functionality as SysML models
- SW code is *generated* from models, (almost) no hand-written source code

=> Safety-critical! Can AI assist? How?



ANSYS SCADE

Certified tool, used by Airbus, Boeing, etc.
for Model Based Systems Engineering (MBSE)



The screenshot displays the ANSYS SCADE software interface, which is used for Model Based Systems Engineering (MBSE). The central workspace shows a graphical representation of a flight display, labeled "SCADE DISPLAY". This display features a central circular gauge with a blue upper half and an orange lower half, marked with numerical values (10, 20, 30). To the left and right of the gauge are vertical scales with values 40, 20, 0, -20, -40 and 5, 0, -5 respectively. The display is overlaid with a "SCADE DISPLAY" logo and an "OpenGL/SC Compliant" logo.

On the left side, a "Primitives" panel lists various graphical elements like Line, Arc, Circle, Ellipse, Crown, Rectangle, Shape, Text, Text Area, Rich Text, Bi-font, and Bitmap. Below this, there are sections for Masks, Containers, References, and Interactors.

On the right side, a "High Level Requirements" panel shows a hierarchical tree structure of requirements, including "Flight Control Unit", "Flight Controller", "Altitude error contribution", "Speed error contribution", "Elevator Command", "Throttle Command", "Alarm Manager", "Altitude Alarm", "Speed Alarm", "Display Logic", and "Primary Flight Display".

Below the requirements panel, a "Properties" panel shows details for the selected "symbology_layer" container, including its origin, layer ID, ratio, user unit, and specification size.

At the bottom, a "Console" panel displays a table of variables and their values:

Name	Type	Representation	Value	Scope	Group	Memory
PN_PITCH_ANGLE	real	n/a	0.0	Input		
PN_ROLL_ANGLE	real	n/a	0.0	Input		
PN_ALTI	real	n/a	0.0	Input		
PN_AIRSPEED	real	n/a	0.0	Input		

„IN THE BEGINNING WAS THE WORD“

Requirements written in natural language are perfect for LLMs (e.g., GPT-4)



Hochschule RheinMain

The screenshot displays the OpenGL/SC software interface. The central canvas shows a flight display simulation with a central speed scale, vertical altitude scales on the left and right, and a top heading scale. The interface includes a menu bar (File, Edit, View, Layer, Object, Transform, Project, Tools, Window, Help), a toolbar, and a left sidebar with a project tree and a primitives list. A red rectangle highlights the 'High Level Requirements' panel on the right, which contains a hierarchical tree of requirements. Below this panel is a 'Properties' section for the selected 'symbology_layer' container.

High Level Requirements Word

- 2. Requirements
 - Flight Control Unit
 - Flight Controller
 - Altitude error contribution
 - FCU_01
 - FCU_04
 - Speed error contribution
 - FCU_02
 - FCU_03
 - Elevator Command
 - FCU_05
 - Throttle Command
 - FCU_06
 - Alarm Manager
 - Altitude Alarm
 - FCU_07
 - Speed Alarm
 - FCU_08
 - Display Logic
 - FCU_09
 - Primary Flight Display

Properties

Container		
symbology_layer		
Origin	0.0	0.0
Layer Id	1	
Ratio	1 x 1	
User unit	666 x 620	
Specification size	666 x 620 pixels	

Console

Name	Type	Representation	Value	Scope	Group	Memory
PN_PITCH_ANGLE	real	n/a	0.0	Input		
PN_ROLL_ANGLE	real	n/a	0.0	Input		
PN_ALTI	real	n/a	0.0	Input		
PN_AIRSPEED	real	n/a	0.0	Input		



SCENARIO FOR THE FUTURE

How realistic (how wise) is it to use AI in safety-critical applications?

FUTURE:

Graphical and functional model, with interface to Model Based Systems Engineering (MBSE) tool

Step 1

Requirements written in natural language are fed into LLM (e.g., GPT-4).

LLM generates API calls for the development tool to create and connect the instances of the **graphical and functional model** of the embedded SW.

Alternative: LLM generates models directly in native file format (e.g., XML) of MBSE tool

Step 2

Models are transferred into internal representation by development tool.

Models are analyzed and edited **by human engineer** in the development tool, only **when necessary** (human-in-the-loop policy).

Step 3

C/Ada source code is generated automatically by code generator (e.g., KCG).

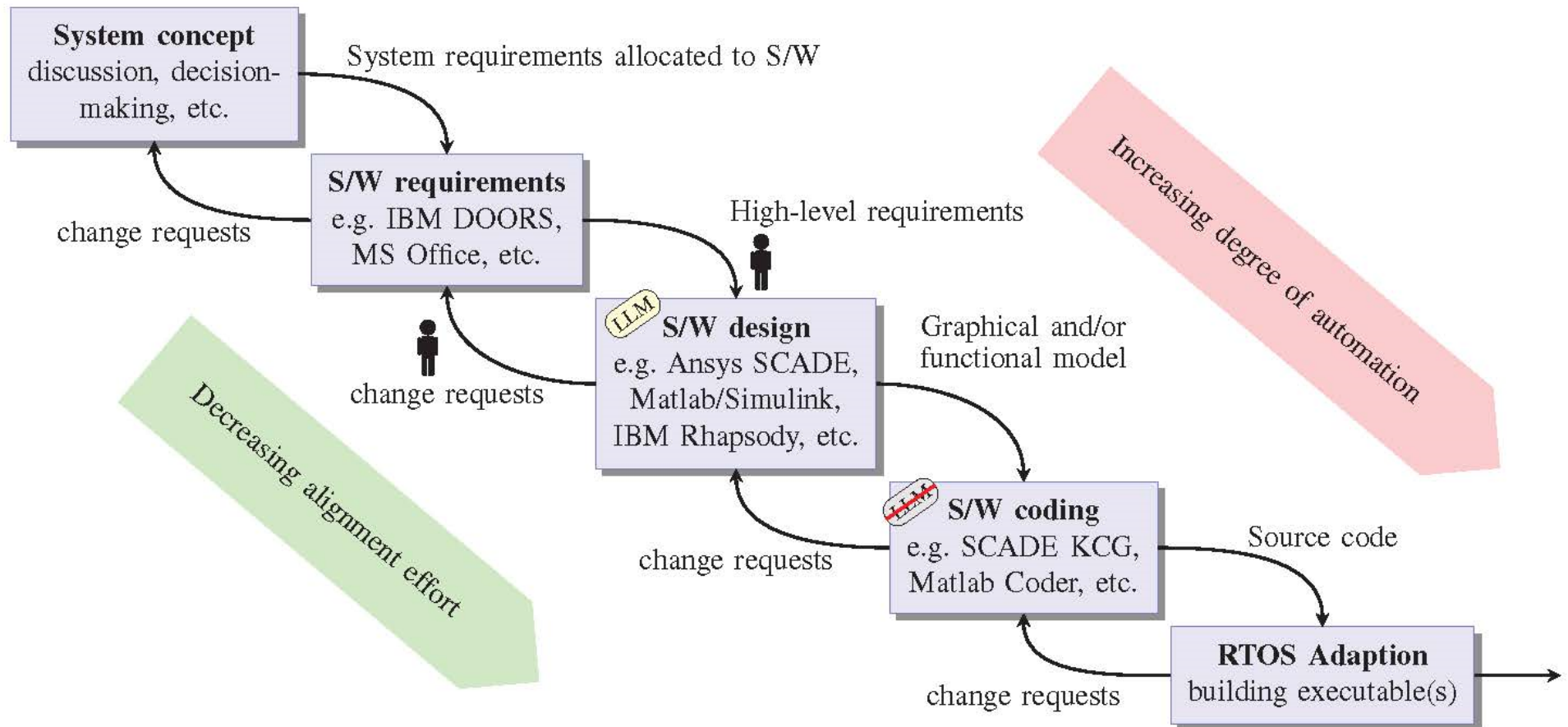
Source code is compiled into binary (executable) for adaption by Real-Time-OS (RTOS)

V-MODEL OF DEVELOPMENT PROCESS

Crucial: Usage of LLM (AI) in which step?



Hochschule RheinMain





METHODOLOGY FOR EVALUATION

Evaluation of the capabilities of current LLMs for a limited test case

TODAY: LIMITATIONS

Limited to graphical model, without direct (automatic) interface to development tool editor



Hochschule **RheinMain**

Step 1

Requirements written in natural language are fed into LLM (e.g., GPT-4).

LLM generates code (e.g., Python, TikZ/LaTeX) for the **graphical models** of the embedded SW only.

Step 2

Code is executed by interpreter and graphical models displayed.

Graphical models are analyzed and transferred into development tool editor **manually by human engineer**.

Step 3

C/Ada source code is generated automatically by code generator (e.g., KCG).

Source code is compiled into binary (executable) for adaption by Real-Time-OS (RTOS)

RESULTS: REQUIREMENTS 1...9 AND GPT-4

Variant (1 of 14) of the visual model, generated using TikZ / LaTeX code from GPT-4



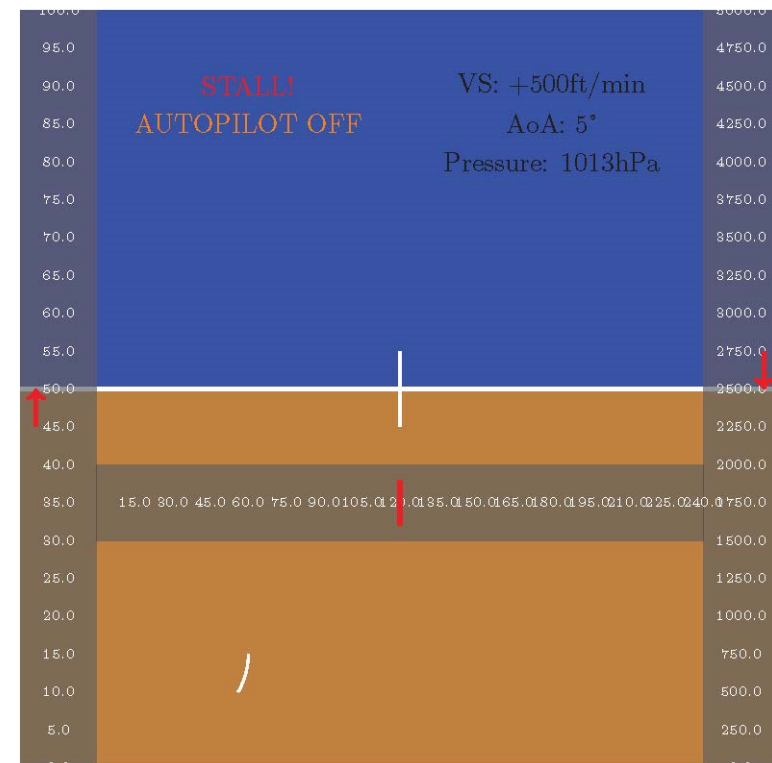
Hochschule RheinMain

1. General Layout & Dimensions:

- The PFD shall have a rectangular aspect
- ratio suitable for installation in standard cockpit instrument panels.
- The sky and earth shall be perfectly aligned at the horizon line.
- The horizon line shall be centered horizontally on the PFD, and its vertical placement shall adjust based on the aircraft's pitch angle.

2. Color and Appearance:

- The PFD shall represent the sky in blue.
- The PFD shall represent the earth in brown.
- The horizon line shall be a distinct, bold white line for easy visibility against both the sky and earth backdrops.



RESULTS: REQUIREMENTS 1...9 AND GPT-4

100% fulfillment achieved



Hochschule RheinMain

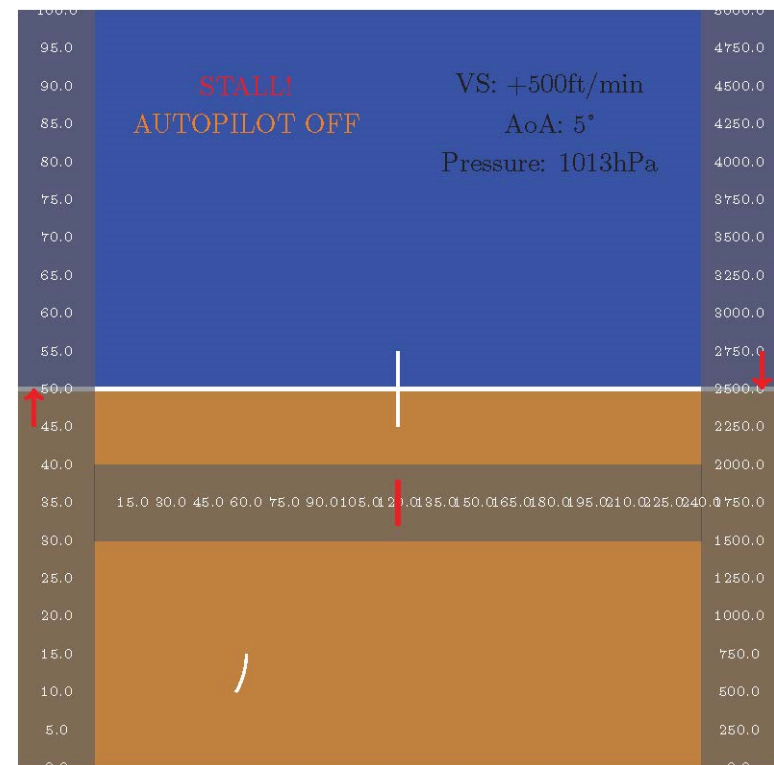
3. ... 7. *omitted for clarity*

8. Additional Flight Information:

- The PFD shall display other pertinent flight data such as vertical speed, angle of attack, and barometric pressure.
- This information should be arranged in a manner that does not clutter the primary attitude information.

9. Warning and Caution Indicators:

- The PFD shall have provisions for displaying warning (red) and caution (amber) indications for critical flight parameters, such as stall warnings or autopilot disengagement.



RESULTS: REQUIREMENTS 1...9 AND GPT-4

Same requirements: Variant with 84% fulfillment



Hochschule RheinMain

Requirements **not** met:

6. Heading Indicator:

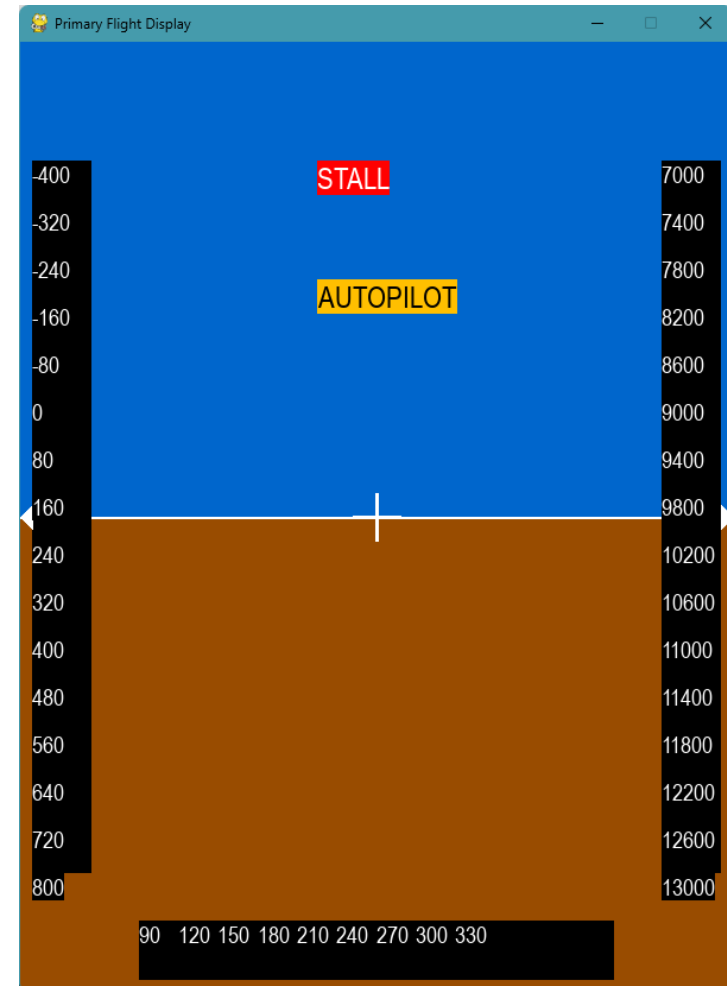
- ...
- The current heading shall be indicated by a **fixed pointer or triangle**, with the tape/rose rotating behind it.

7. Turn Coordinator:

- The PFD shall incorporate a turn coordinator, represented by a curved line or other suitable graphical representation, to show the **rate and direction of turn**.

8. Additional Flight Information:

- The PFD shall display other pertinent flight data such as **vertical speed**, **angle of attack**, and **barometric pressure**.



RESULTS FOR ALL 9 REQUIREMENTS

Only GPT-4 and GPT-3.5 succeeded

- GPT-4 always generated error-free code
- GPT-3.5 generated correctable code
- Other LLMs tested not ready for integration

Not to be confused:

Code generated by LLM
vs.

Code generated by KCG

LLM	Degree of fulfillment	# of error-free code variantes	# of correctable code variants
GPT-4	Min. 37% Median 74% Max. 100%	14 of 14	N/A
GPT-3.5	Min. 16% Median 39% Max. 68%	7 of 8	1 of 8 ^a
CodeLlama	0%	0 of 2 ^b	0 of 2 ^b
StarChat	0%	0 of 2 ^c	0 of 2 ^c
CodeGen2.5	0%	0 of 2 ^c	0 of 2 ^c

Footnote:

^a contained errors that GPT-3.5 corrected after being instructed

^b code output ended after approx. 5000 characters

^c timeout after several minutes without any output

RESULTS FOR REQUIREMENTS 1 TO 5

CodeLlama now produced code

- CodeLlama generated model for shorted list of requirements due to restricted context window
- Other LLMs tested still not usable

LLM	Degree of fulfillment	# of error-free code variantes	# of correctable code variants
GPT-4	100%	8 of 8	N/A
GPT-3.5	Min. 64% Median 85% Max. 96%	4 of 4	N/A
CodeLlama	Min. 14% Median 29% Max. 86%	0 of 6	6 of 6 ^a
StarChat	0%	0 of 2 ^b	0 of 2 ^b
CodeGen2.5	0%	0 of 2 ^b	0 of 2 ^b

Footnote:

^a repeatedly the same error using Qt (code could be corrected manually)

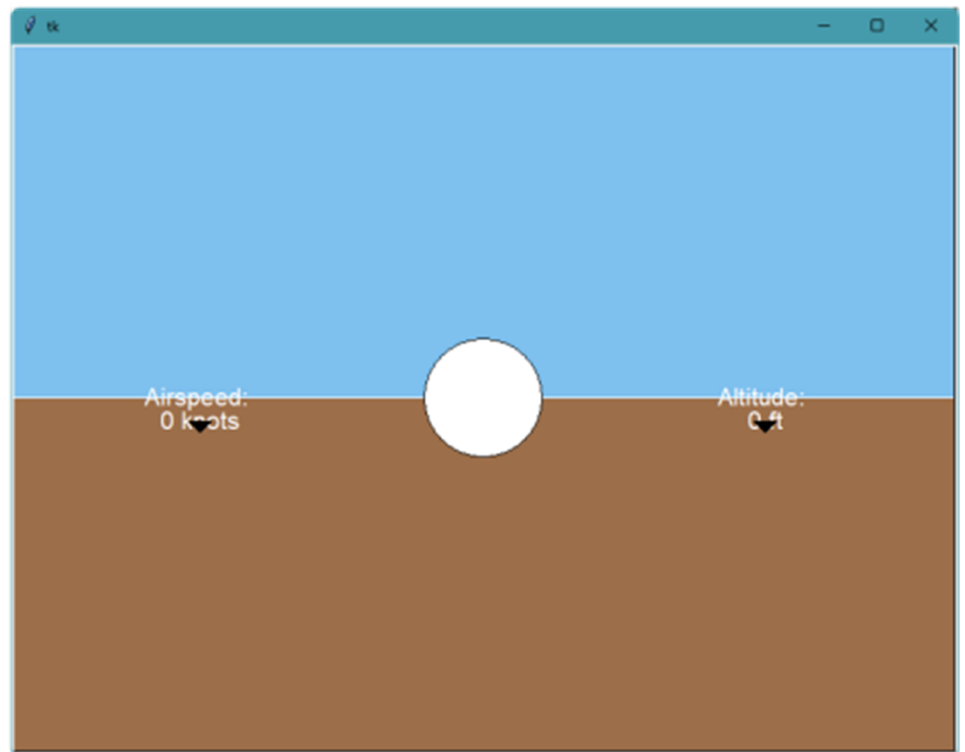
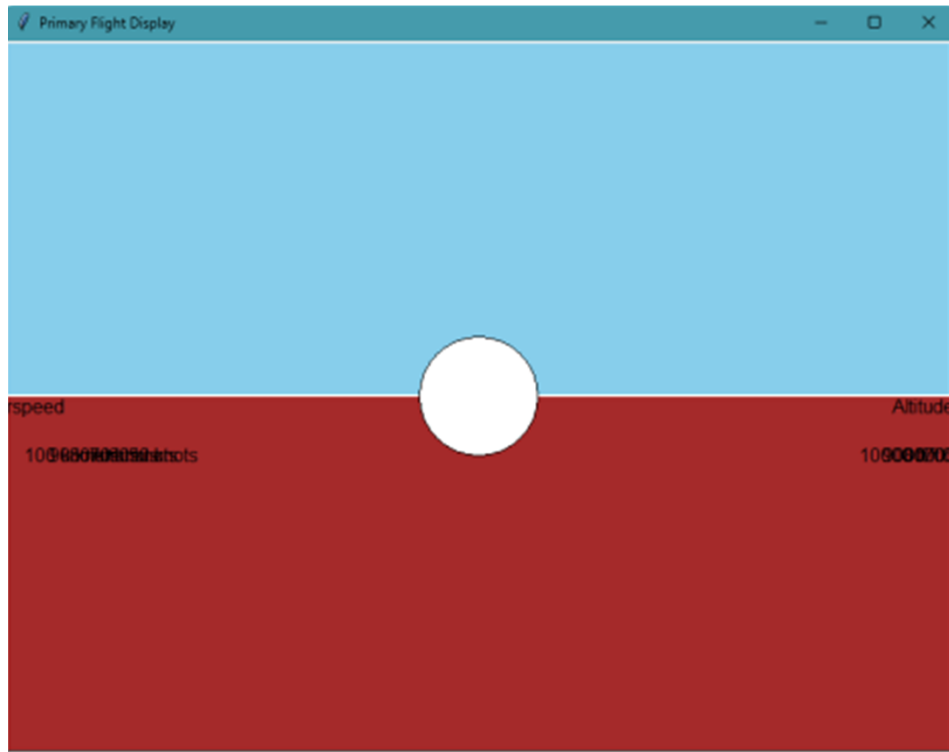
^b timeout after several minutes without any output

CODELLAMA VARIANTS

Best results (57% and 86% fulfillment) for shorted list of requirements (requirements 1...5)



Hochschule RheinMain





CONCLUSION

LLMs / AI can *assist* human engineers, but should never replace them completely

SCENARIO

AI acting as assistant to engineering teams



Hochschule RheinMain

Challenge / Risk:

What if the assistant (AI) becomes more experienced, efficient and reliable than the human team?

⇒ get rid of human-in-the-loop policy?

⇒ humans only for high-level requirements?





Hochschule **RheinMain**

REFERENCES AND CONTACT INFORMATION

Comments and discussion always welcome!

SELECTED REFERENCES

Full list: see paper



1. OpenAI, “Gpt-4 technical report,” ArXiv, vol. abs/2303.08774, 2023. [Online]. Available from: <https://arxiv.org/abs/2303.08774>
2. H. Touvron et al., “Llama 2: Open foundation and fine-tuned chat models,” 7 2023. [Online]. Available from: <https://www.semanticscholar.org/paper/104b0bb1da562d53cbda87aec79ef6a2827d191a>
3. R. Li et al., “Starcoder: may the source be with you!” 2023. [Online]. Available from: <http://arxiv.org/abs/2305.06161>
4. E. Nijkamp, H. Hayashi, C. Xiong, S. Savarese, and Y. Zhou, “Codegen2: Lessons for training llms on programming and natural languages,” arXiv preprint, 2023.
5. R. Bagnara, A. Bagnara, and P. M. Hill, “The misra c coding standard and its role in the development and analysis of safetyand security-critical embedded software.” CoRR, vol. abs/1809.00821, 2018. [Online]. Available from: <http://dblp.uni-trier.de/db/journals/corr/corr1809.html#abs-1809-00821>
6. R. Bagnara, M. Barr, and P. M. Hill, “Barr-c: 2018 and misra c: 2012: Synergy between the two most widely used c coding standards.” CoRR, vol. abs/2003.06893, 2020. [Online]. Available from: <http://dblp.uni-trier.de/db/journals/corr/corr2003.html#abs-2003-06893>
7. J. Holt, SysML for Systems Engineering: A Model-Based Approach, ser. Computing. Institution of Engineering and Technology, 2018. [Online]. Available from: <https://digital-library.theiet.org/content/books/pc/pbpc020e>

SELECTED REFERENCES

Full list: see paper



8. D. Iqbal, A. Abbas, M. Ali, M. U. S. Khan, and R. Nawaz, “Requirement validation for embedded systems in automotive industry through modeling,” IEEE Access, vol. PP, pp. 1–1, 01 2020.
9. H. Touvron et al., “Llama: Open and efficient foundation language models,” 2023. [Online]. Available from: <http://arxiv.org/abs/2302.13971>
10. P. Liang et al., “Holistic evaluation of language models,” 2022. [Online]. Available from: <https://arxiv.org/abs/2211.09110>
11. F. Chollet, “On the measure of intelligence,” 2019. [Online]. Available from: <http://arxiv.org/abs/1911.01547>
12. A. Srivastava et al., “Beyond the imitation game: Quantifying and extrapolating the capabilities of language models,” 2023. [Online]. Available from: <http://arxiv.org/abs/2206.04615>
13. E. Davis, “Benchmarks for automated commonsense reasoning: A survey,” 2023. [Online]. Available from: <https://arxiv.org/abs/2302.04752>
14. T. B. Brown et al., “Language models are few-shot learners,” 2020. [Online]. Available from: <https://arxiv.org/abs/2005.14165>
15. H. W. Chung et al., “Scaling instruction-finetuned language models,” 2022. [Online]. Available from: <https://arxiv.org/abs/2210.11416>

THANK YOU FOR LISTENING



Hochschule **RheinMain**

Contact

Prof. Dr. Matthias Harter
Faculty of Engineering
Department of Electrical
Engineering and Information
Technology

Am Brückweg 26
D-65428 Rüsselsheim

+49 6142 898-4223
matthias.harter@hs-rm.de

