

## PANEL #2

# Theme: The Future of Software + AI Symbiosis

**Moderator:** Prof. Dr. Herwig Mannaert, University of Antwerp, Belgium

### Panelists:

Prof. Ing. Luigi Lavazza, Università degli Studi dell'Insubria, Italy

CEO Scott Gallant, Effective Applications Corporation, USA

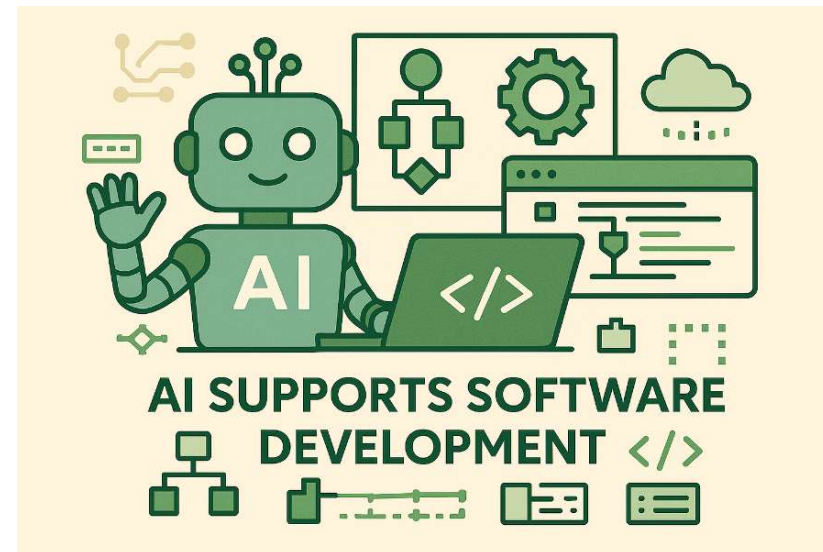
Prof. Dr. Hans-Werner Sehring, NORDAKADEMIE gAG, Germany

- *I would like to exchange ideas on the current and future relationship between AI/GPT and software engineering*
- **On the long-term impact of AI on software engineering**
  - Make human software engineering obsolete ?
  - Increase software productivity and/or complexity ?
- **On the viability of training dedicated or focused LLMs**
  - Need tremendous energy and computing resources ?
  - Remain barriers for engineers/companies to participate ?
- **On the systemic or essential issues of current LLMs**
  - Route to explainability or inevitable hallucinations ?
  - Purely deep learning or some *if then* programming ?



Herwig Mannaert  
Univ. of Antwerp

- **AI for Software Development**
  - LLMs as very sophisticated CASE tools
    - ➔ Need humans!
- **AI for Software Development (locally)**
  - So many LLMs / agents / ...
    - ➔ Specialists needed
      - ❑ know IA tools
      - ❑ know internal development
      - ❑ know how to best use AI tools in their organization
  - A new organizational issue
    - ➔ Governance of AI
- **AI for Software Development (globally)**
  - Over time, will AI help improve the global software base?



Luigi Lavazza  
Univ. Insubria

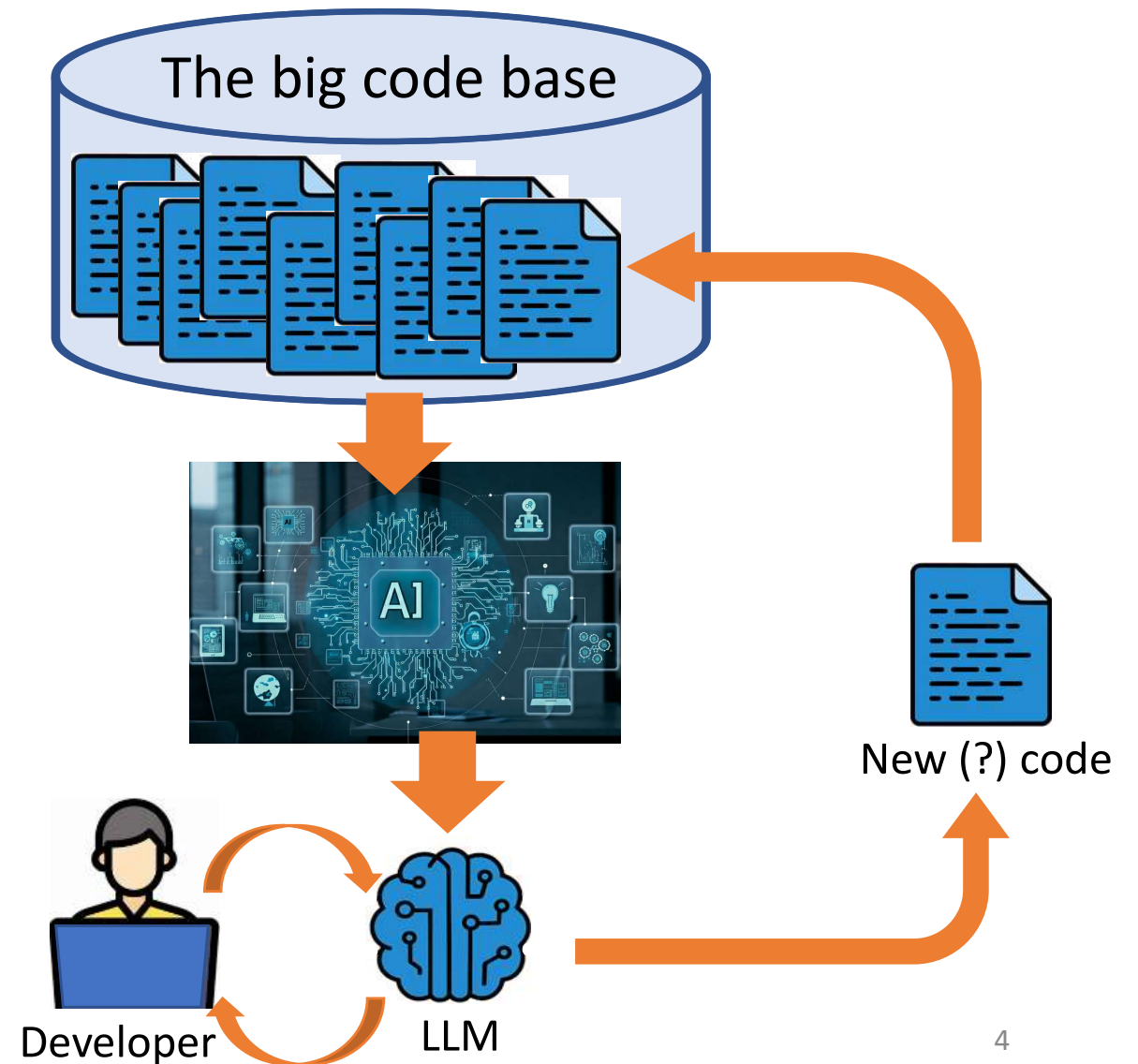
## AI cyclic development

- AI software + software code -> AI models

New, assuming that some human elaborates the code yielded by the AI

- AI models -> **new software code**
- New AI software + new software code -> new AI models
- ... (and so forth) ...

*If AI learns from the software code produced by the same AI, will the process stop improving at some point?*



## ■ Background from my domain's perspective

### ■ Military software context:

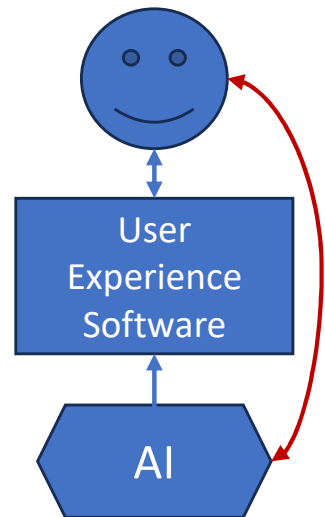
- AI trustworthiness via predictability and explainability: usage for decision support, threat detection/situation awareness, unmanned autonomous vehicles, accurate training and simulation without negative training.
- Most militaries have stated that AI will not be involved in lethal decision making, but how long can that be true with evolving threats?
- AI-provided information can lead humans to make lethal decisions. How many times have we made a wrong turn because of the phone's erroneous directions? (1983 Soviet nuclear false alarm incident)

### ■ Simulation software context:

- I believe that inclusion of AI in more software development projects will require new design and architecture principles which will be evolutionary much like object-oriented programming was.
- How we design modularity, parallelism, fault tolerance, and the user experience will all be significantly impacted.
- Software development tools, systems engineering tools, and user interface design tools are all being rapidly transformed with AI. Working towards context engineering (see AISED paper 30013).
- Simulations are moving from rule-based models, decision trees and state machines to AI. To validate simulations, we cannot use black-box systems. Explainable AI is being researched in this domain.



Scott Gallant



Security  
Ethics  
Privacy  
Ownership  
Bias

- **Questions that may be interesting for the discussion:**
  - How does a non-technical person know if the AI-generated software system is well architected, can scale, is secure, and quality software that can easily be improved over time?
  - How can we avoid ‘prompt purgatory’ when trying to refine code generated by AI?
  - Using AI to generate prototypes for more agile development sounds great. Would we have to start from scratch to build the delivered system with a better architecture and improved quality of software (more secure, scalable, modular, etc.)?
  - We all have different use cases and constraints for our software projects. How can we optimize the AI software generation for our specific use cases while sharing in technical advancements of the AI models?
  - Can we discretize AI’s role into smaller building blocks to better manage the technology and the process? For example, having AI agents: turn user requirements into design, build the communication layer, build the data layer, build the business logic, code review for quality benchmarks, test for security vulnerabilities, test for large scale usage, test in varying use cases, etc.

## ▪ Challenge 1: AI-driven software engineering

- AI... instead of software engineering
  - Traceability, maintainability, etc.?
  - **Software crisis 2.0**
- AI... for model generation
  - LLMs for model generation: natural language → model
  - But software architecture change from software blueprint to decision process [Shaw, Kruchten]
- AI... for model transformation
  - LLMs trained for languages of models
  - “Meta Language Models”, not necessarily large

My perspective on this panel:  
(Gen.) AI as a tool for  
software construction



Hans-Werner  
Sehring  
NORDAKADMIE,  
Germany

- **Challenge 2: User interfaces change from formal to natural language**
  - **Tremendous advantage of LLMs**
    - Ex.: no need to define formal models for requirements
    - But then, there is expert jargon, context matters
  - **Participation vs. potential risk of Platon/Sokrates, Danning-Kruger**
    - Everyone can (and will) participate in software creation
    - See data silos of the 1990's → "Vibe Coding Cleanup Specialists"
  - **Need to relearn principles and methods**
    - All the things C.S. cannot cope with, vagueness, subjectivity, etc.
    - Need to add basic sciences to C.S., engineering → humanities?



Hans-Werner  
Sehring  
NORDAKADMIE,  
Germany

- **Challenge 3: The need for an algorithmic way of thinking**
  - **Generative AI changes computation from algorithmic to descriptive**
    - good since this was always the ultimate goal of programming
    - programming languages like Prolog all the time at the borderline between programming and AI
  - **Claim: even for a declarative description of a desired solution, you need C.S. skills**
    - abstraction (classification, refinement)
    - modularization (divide and conquer strategy)
  - **Claim 2: you earn C.S. skills from programming / algorithmic thinking**



Hans-Werner  
Sehring  
NORDAKADMIE,  
Germany

## PANEL #2 DISCUSSION

# Theme: The Future of Software + AI Symbiosis

**Moderator:** Prof. Dr. Herwig Mannaert, University of Antwerp, Belgium

### Panelists:

Prof. Ing. Luigi Lavazza, Università degli Studi dell'Insubria, Italy

CEO Scott Gallant, Effective Applications Corporation, USA

Prof. Dr. Hans-Werner Sehring, NORDAKADEMIE gAG, Germany

**All attendees**