Qualitätssicherung und Qualitätsmanagement

KI für QS ⇒ AI4QA

Sommersemester 2025

Prof. Dr. Andreas Metzger

PALUNO The Ruhr Institute for Software Technology
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Hinweis: Diese VL-Einheit ist nicht klausurrelevant!

Motivation

Why AI4QA at all?

Insights from a roundtable of experts from industry, academia, and government [Bannon & Laplante, 2024]:

AI to...

- ...automate repetitive tasks
- ...speed up workflows
- ...provide a "copilot" that can code, test, and anticipate challenges
- ...reduce mundane and difficult tasks: Approx. 40%–70% of the time is spent on creating and testing code

Sejal Amin is the chief technology officer (CTO) at Shutterstock. Contact her at sejalamin@gmail.com.

Mitch Ashley is the CTO at The Futurum Group and the CTO at Techstrong Group. Contact him at mashley@futurumgroup.com.

Sasha Czarkowski is the vice president of technology at Ergonautic. Contact her at sasha@ergonautic.ly.

Patrick Debois is a GenAI and DevOps specialist and an author. Contact him at patrick.debois@jedi.be.

Dave Farley is the founder and director of Continuous Delivery Ltd. and an author. Contact him at dave@continuous-delivery.co.uk.

Nathen Harvey is a DORA lead and developer advocate at Google Cloud. Contact him at nathenharvey@google.com.

David Linthicum is an enterprise technology innovator, educator, and author. Contact him at david@davidlinthicum.com.

Kris Saling is the acting director/chief of staff at the Innovation Directorate, U.S. Army Recruiting Command. Contact her at kristin.saling@gmail.com.

David Sisk is the managing director of application modernization and application architecture at Deloitte Consulting. Contact him at dasisk@deloitte.com.

Why AI4QA at all?

Applications of AI for QA:

- Increased *automation*; e.g.,
  - Test case generation
  - Automatic code reviews
  - And also code generation (i.e., constructive QA)
- *Continuous* testing; e.g.,
  - Seamless integration of AI into DevOps (AI-augmented DevOps) and CI/CD pipelines

Why AI4QA now?

Rapid increase of the performance and quality of modern AI algorithms that are increasingly applied to a broader range of problems:

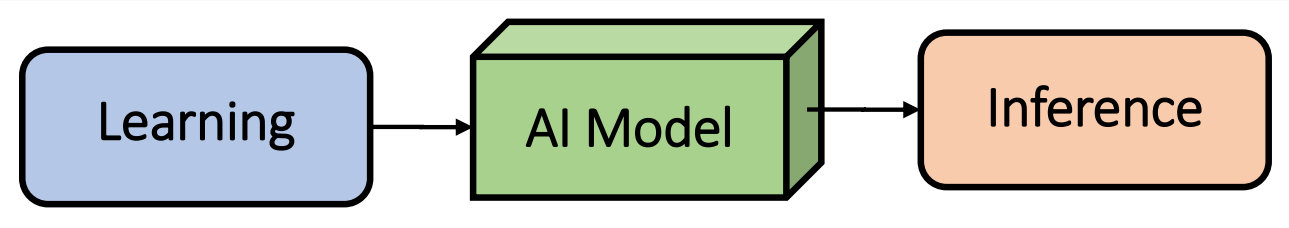
- **Deep Supervised Learning**
  - Learning how to label a given input
  - Training data: labeled training data (data + label)
  - *Example applications:* image classification, spam detection, fraud detection
- **Deep Reinforcement Learning (Deep RL)**
  - Learning which actions to take in a given situation
  - Training data: feedback (rewards) from the environment
  - *Example applications:* game playing, robotics, control systems
- **Generative AI (GenAI)**
  - Learning how to generate outputs based on "prompts"
  - Training data: large (typically) unlabeled data sets
  - *Example applications:* text, image, video, music generation
- **Combination of AI algorithms**
  - *Breakthrough example:* Google DeepMind's AlphaGo
  - Famously defeated Lee Sedol (world's best Go player)
  - AlphaGo - The Movie | Full award-winning documentary

AI Fundamentals

Basic AI Concept

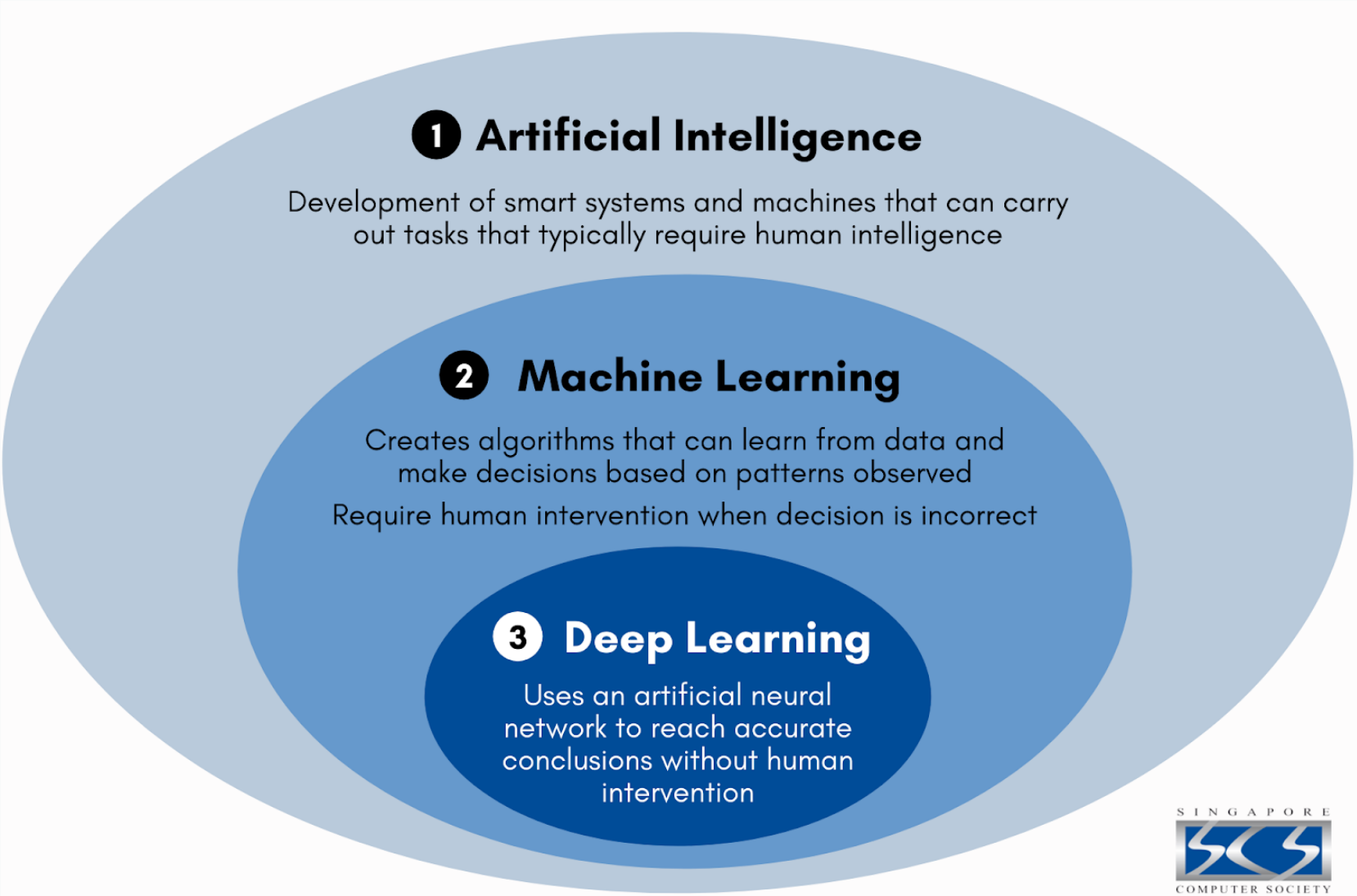
In general, two main phases:

- **Learning (aka. Training)**:
  Create the AI model
- **Inference (aka. Prediction)**:
  Use the AI model to generate labels, actions, output



Classes of AI Algorithms

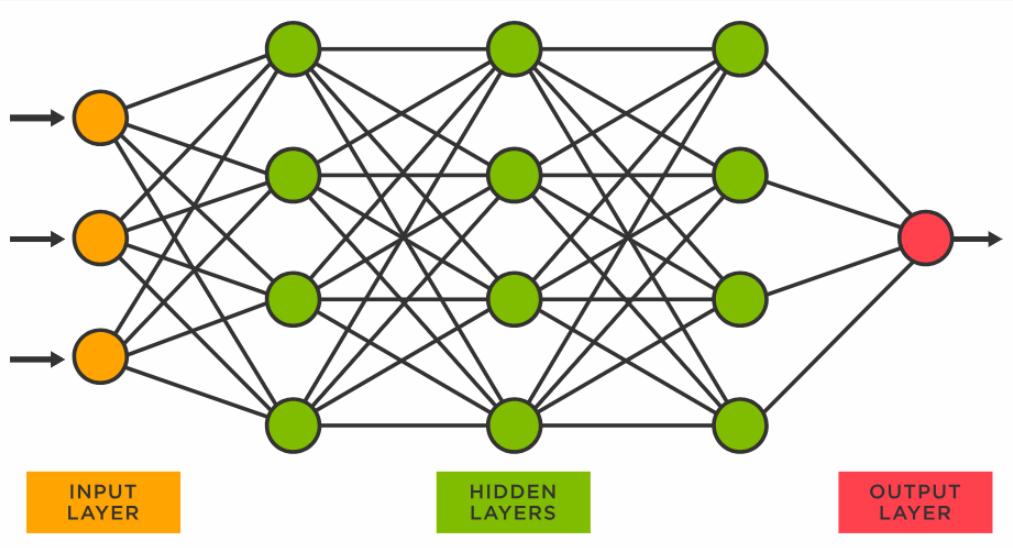
Modern, successful AI algorithms = *Deep Learning*:



Deep Learning

Deep Learning AI model = *Artificial Neural Network* (ANN):

- Inspired by the biological neural networks that constitute animal brains
- Consist of interconnected nodes ("neurons") organized in layers
- Each neuron receives weighted inputs from other neurons, processes them, and applies an activation function to produce an output
- Learning via adjusting weights of the connections between neurons

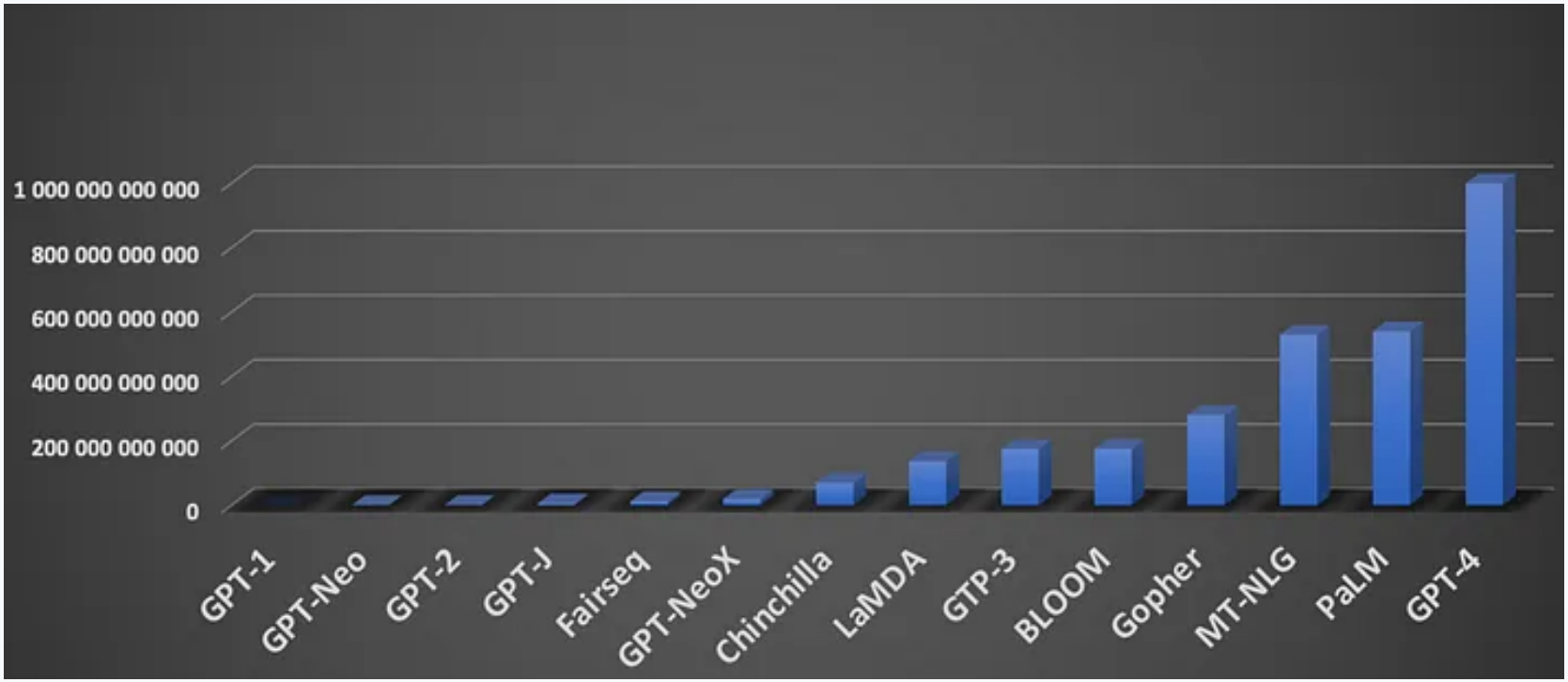


Source: https://www.index.dev/blog/what-is-llm

Deep Learning

Growing ANN size of modern AI systems:

- Example: Number of learnable parameters (= weights + biases[1]) of modern LLMs



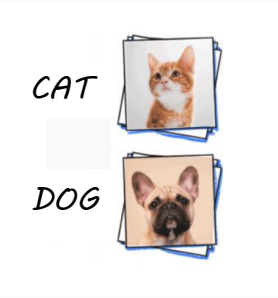
Source: Medium

[1] Biases shift activation function, helping ANN learn more complex patterns.

Deep Learning

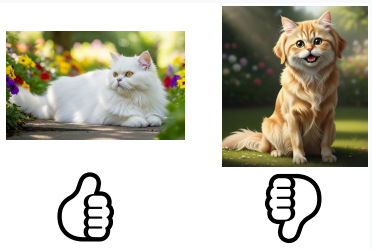
Three main types of Deep Learning:

- Deep Supervised Learning: Learning from labeled data



- Generative AI: Learning from unlabeled data

- Deep Reinforcement Learning: Learning from feedback



AI4QA Applications

GenAI for Code / Test Case Generation

By using **Large Language Models** (LLMs), such as:

- OpenAI ChatGPT (USA)
- Google Gemini (USA)
- Meta LLaMA (USA)
- Anthropic Claude (USA)
- AI2 Tülu (USA)
- Perplexity.AI (USA)
- xAI Grok (USA)
- InceptionLabs Mercury (USA)
- Alibaba Qwen (China)
- DeepSeek (China)
- Aleph Alpha Luminous (Germany)
- Mistral Le Chat (France)
- Open Euro LLM (EU – in preparation)
- …

Info: Access to "commercial" versions of several LLMs via UDE's ChatAI

GenAI for Code Generation

Simple example:

LIVE Demo…

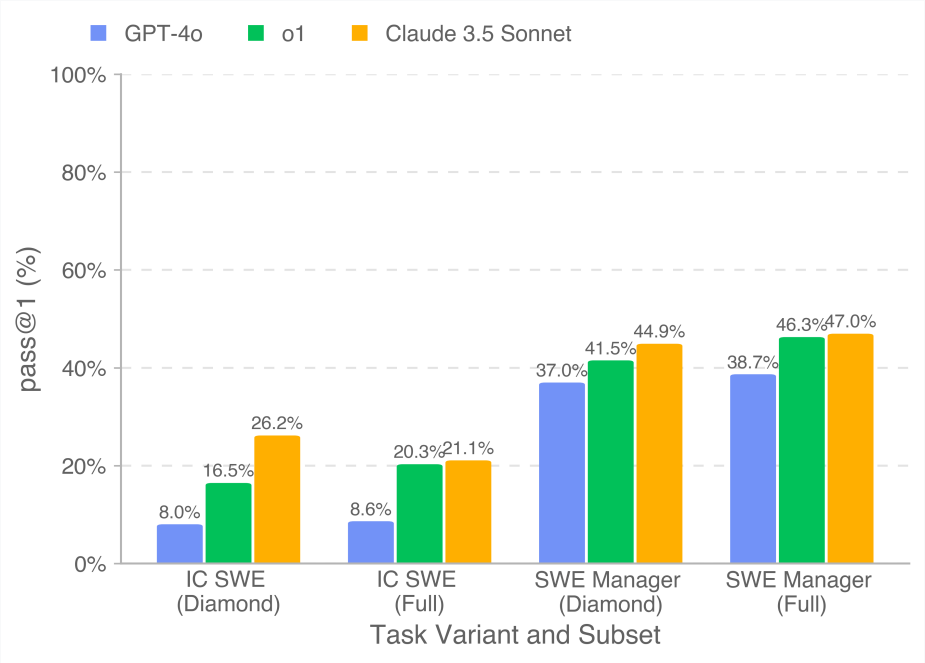
Prompts:

1. Generate Java code that converts Fahrenheit to Centigrade
2. Catch wrong temperature values
3. Improve the code
4. Improve the code
5. Improve the code

GenAI for Code Generation

Current performance of GenAI [May 2025 Paper]:

- OpenAI's SWE Lancer Benchmark: Analysis of how GenAI can complete 1,400 real development and management tasks
- Best performing AI model could successfully complete 26.2% of the development tasks



(IC = individual contribution)

GenAI for Test Case Generation

LIVE Demo

Prompt: The following Java program is given:

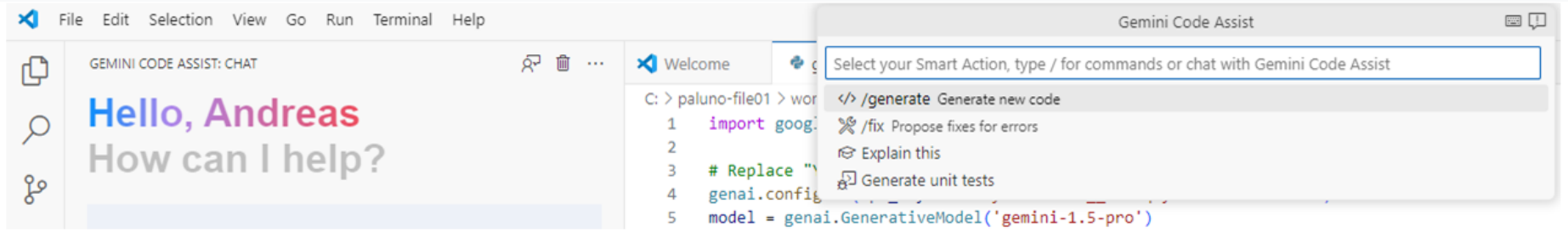
```
public class TemperatureConverter {
  public static double fahrenheitToCelsius(double fahrenheit) {
    return (fahrenheit - 32) * 5/9;
  }

  public static void main(String[] args) {
    double fahrenheit = 98.6; // Example temperature
    double celsius = fahrenheitToCelsius(fahrenheit);
    System.out.printf("%.1f F is equal to %.1f C%n", fahrenheit, celsius);
  }
}
```

Generate a set of JUnit test cases that meet the following criteria:
Representative: Informative about untested cases
Error-prone: High probability of revealing errors
Non-redundant: Highest possible new coverage with each new test case
Economical: Optimal within the given budget

GenAI for Test Case Generation

Integration into development environments; e.g., VisualStudio:



Note: The above-presented approaches to test case generation do not use the software's specification/requirements, which should be the actual source for determining the expected output!
Instead, they use their understanding of programming paradigms, common patterns, the function's name, docstrings, and comments to infer the software's likely purpose.

GenAI for Automatic Code Review

Example tools:

- GitHub Copilot:
  - Code suggestions
  - Auto-completions (constructive QA during programming)
- Amazon CodeGuru Reviewer:
  - Code analysis for potential bugs, security vulnerabilities, and performance issues
  - Recommendations for code improvement
- Snyk DeepCode AI:
  - Identify *security* bugs
  - Suggest fixes
  - Uses combination of AI techniques (deep supervised learning to recognize patterns in code associated with bugs + GenAI to suggest bug fixes)

GenAI for Automatic Code Review

But: code reviews are so much more than just finding defects and improving code quality!

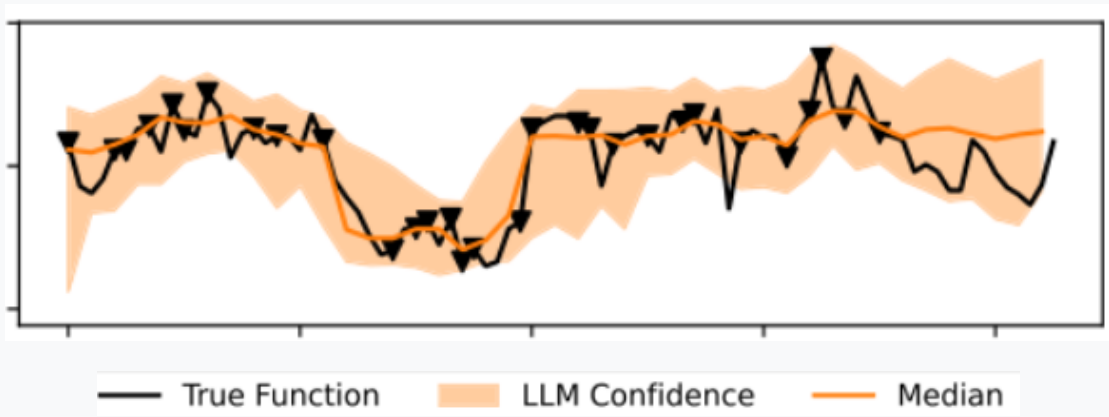
Source: Award keynote at FSE 2025: Alberto Bacchelli, Christian Bird: Expectations, outcomes, and challenges of modern code review. ICSE 2013

Discussion

Challenges and (Current) Limitations of AI4QA

Reproducibility (also see [Bannon & Laplante, 2024]):

- GenAI never produces exactly the same output (cf. "random noise")
- Even for same prompt, AI most likely will not produce the same QA artifacts/results
- Might only be controlled to a certain degree via model configuration (e.g., setting "temperature" and "top_p" of an LLM to zero)



Source: https://neurips.cc/virtual/2024/poster/95832

Challenges and (Current) Limitations of AI4QA

Bias (also see [Bannon & Laplante, 2024]):

- AI model may perpetuate biases of training data
- Could lead to inaccurate or unfair QA results
- *Example:* LLM prompt: "Draw images of watches showing 12:00"
- What most LLMs (even the most recent versions) will give you:

Challenges and (Current) Limitations of AI4QA

Clever Hans Effect:

- Gen AI models may learn from irrelevant features of the training data
  - *Example:* Learning from handwritten notes instead the actual medical image
  - Clever Hans was a horse that appeared to perform arithmetic and other intellectual tasks during exhibitions in Germany in the early 20th century
  - A psychologist demonstrated that Hans was not actually performing these mental tasks, but was watching the reactions of his trainer.
  - Hans responded directly to involuntary cues in the body language of the human trainer, who was entirely unaware that he was providing such cues.

Source: Wikipedia

Challenges and (Current) Limitations of AI4QA

Data Pollution (aka. Data Contamination):

- T evaluate and compare performance of Gen AI solutions benchmark data sets are typically used

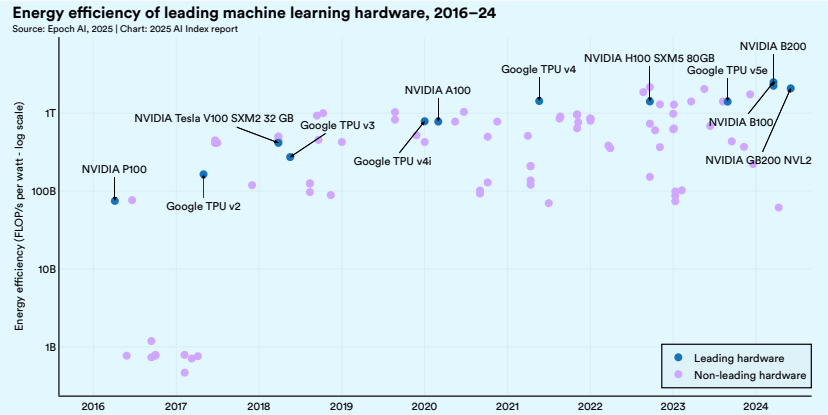
- **But:** GenAI model (e.g., LLM) may have been inadvertently or intentionally exposed to the benchmark data during training or fine-tuning!
  $\Rightarrow$ GenAI model has already seen the 'correct' solution!

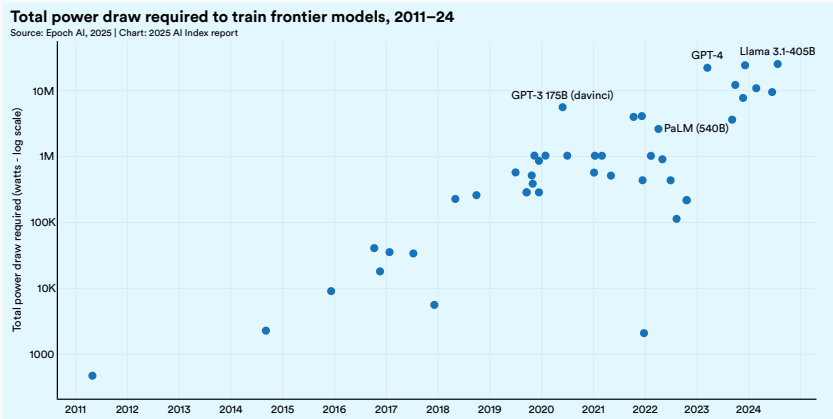
- Consequences:
  - *Artificial Overestimation:* GenAI performance will be artificially inflated
  - *Compromised Evaluation Integrity:* Purpose of benchmark is to objectively measure a technique's performance on unseen data. Data contamination undermines this integrity, making it difficult to accurately compare different techniques
  - *Poor Real-World Performance:* A GenAI model that excels on a contaminated benchmark may perform poorly in real-world applications where it encounters truly novel and diverse data.

Challenges and (Current) Limitations of AI4QA

Resource Usage / Sustainability:

- Energy efficiency of AI hardware improves by ca. 40% per year
- But: AI models become increasingly larger
  $\rightarrow$ elimination of hardware efficiency gains





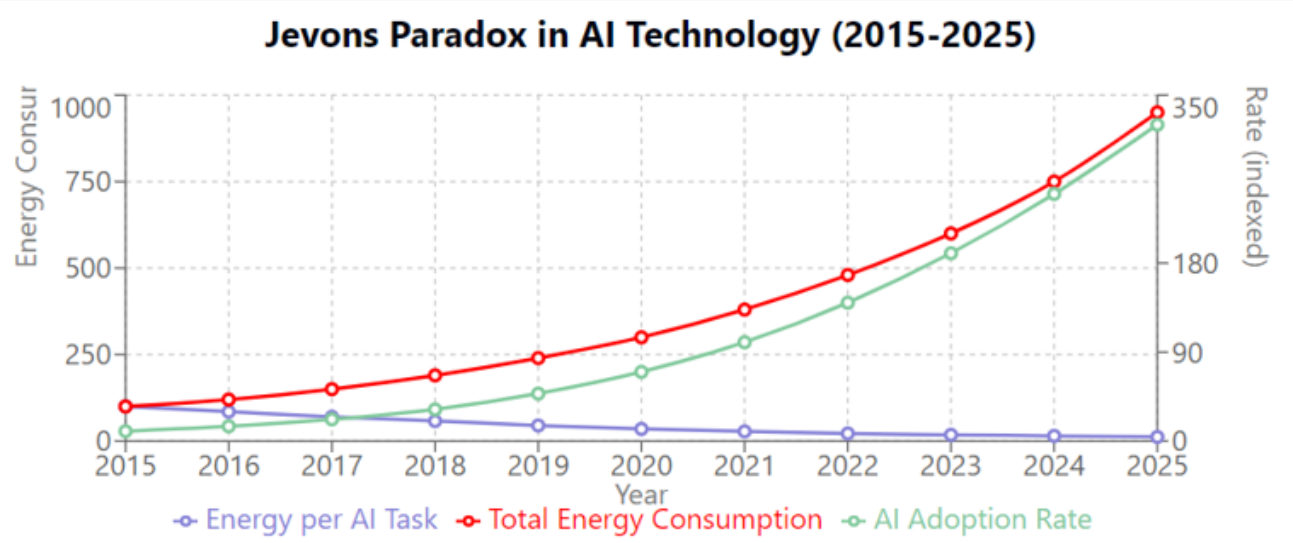
energy efficiency = FLOP per Watt

Source: Artificial Intelligence Index Report 2025, Stanford University

Challenges and (Current) Limitations of AI4QA

Resource Usage / Sustainability:

- *Jevons-Paradoxon!*
- Technical development that increases efficiency and resource needs ultimately leads to a higher resource usage
- More efficient technology → less costs → higher usage and adoption → elimination of efficiency gains

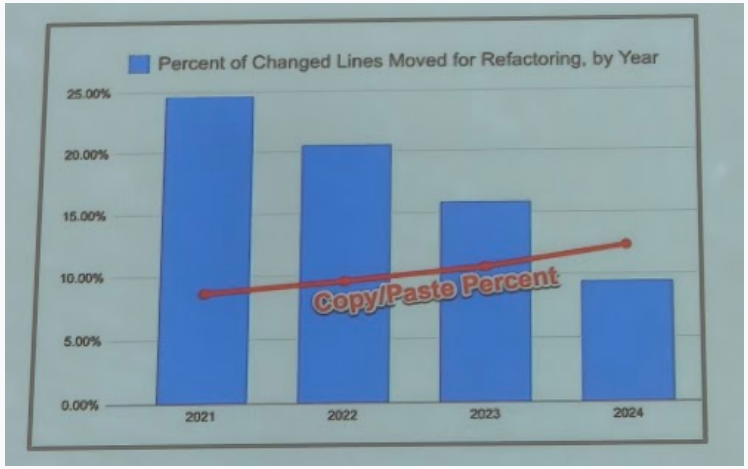


Source: [AI-generated for illustrative purposes]

Challenges and (Current) Limitations of AI4QA

Technical Debt:

- "…ease of AI-generated code is creating a seductive trap – while it feels productive to quickly generate new code, we're actually building technical debt faster than ever before" [Steve Haupt, andrena objects]
- Instead of working DRY (Don't Repeat Yourself) we are constantly tempted to duplicate
- Adding "generated" code is less work compared to refactoring existing code → Refactoring is "dying"



Impact of AI4QA

Team Dynamics (in general when AI is used for IT tasks):

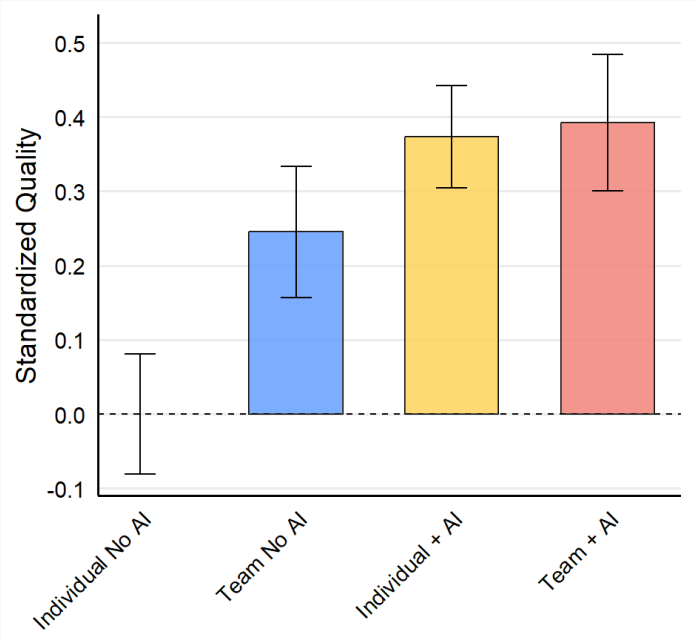
- Senior team members will become supervisors of the AIs
- Junior team members may benefit, but require continuous and accelerated learning

$\Rightarrow$ AI as copilot; e.g., "pair programmer"?

Impact of AI4QA

General effect of employing AI as a "copilot":

- Empirical assessment by Harvard Business School
- Average quality scores for tasks performed by individuals/teams with and without AI



Source: The Cybernetic Teammate: A Field Experiment on Generative AI Reshaping Teamwork and Expertise, Harvard Business Working Paper No. 25-043

Impact of AI4QA

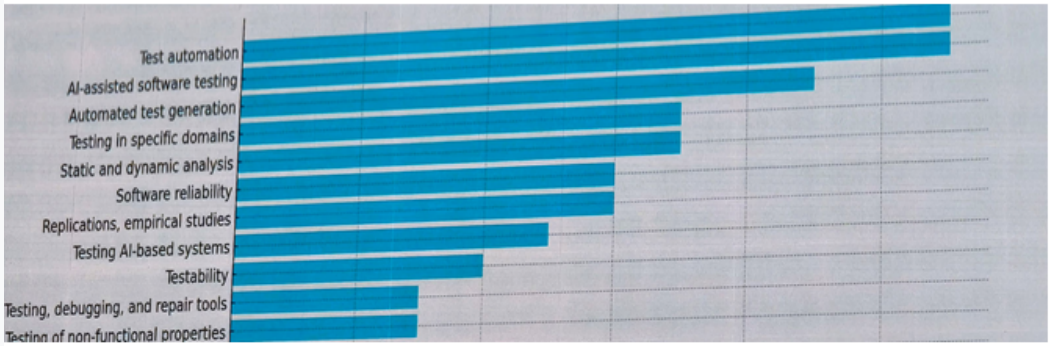
AI skills:

- Need for skilled personnel to develop, implement, and maintain AI-powered QA tools
- Need to invest in training and development to ensure QA teams have necessary skills to effectively utilize AI
- NB, the EU AI Act requires everybody to be adequately trained in AI if the organization is employing AI

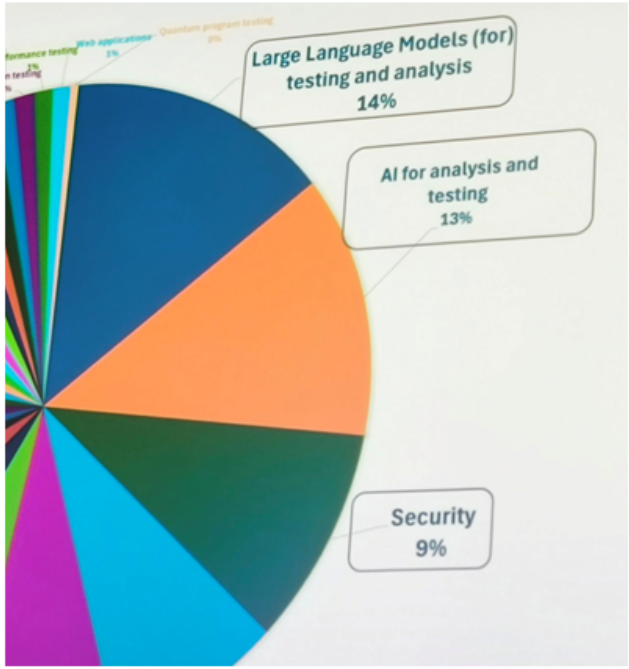
Relevance of AI4QA in Research

Statistics from the two top-tier conferences on software testing and analysis.

ICST 2025: Accepted Paper Topics



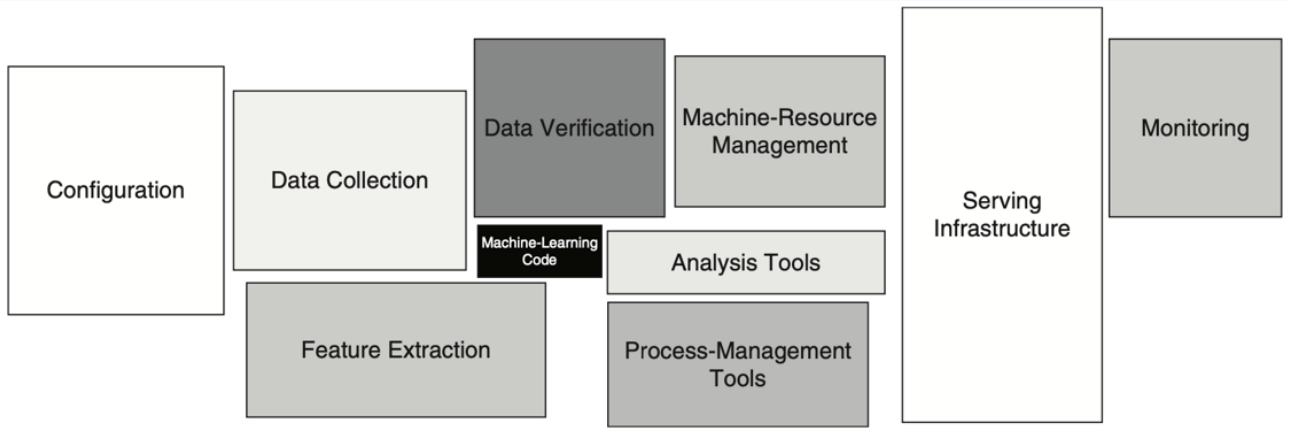
ISSTA 2025: Submitted Paper Topics



Finally, what about QA4AI?

Many future software systems will contain AI components

- $\Rightarrow$ Relevance of SE [Menzies, 2020]:
  - AI software mostly isn't about AI [Sculley et al., 2015]:
    Much of what we know about SE applies to AI



  - AI software needs software engineers:
    Software engineers are necessary to tend to AI systems
  - Poor SE leads to poor AI:
    AI tools suffer when SE is ignored
  - Better SE leads to better AI:
    AI tools benefit when core SE principles are applied

Finally, what about QA4AI?

- $\Rightarrow$ QA can be complex and time-consuming due to specifics of AI models
  - How large must the test data sets be?
  - How to handle the non-reproducibility of GenAI results?
  - How to measure the "coverage" of AI models; e.g., can we use "neuron coverage" similar to branch coverage?
  - ...
  - $\rightarrow$ *Many significant challenges for joint SE and AI research!*

References

- *Reinforcement Learning: An Introduction*, R. S. Sutton and A. G. Barto, 2nd ed., MIT Press, 2018
  (nur physikalisch verfügbar in der Bib)
- *Deep Learning*, Ian Goodfellow, Yoshua Bengio, Aaron Courville, MIT Press,2016
  (nur physikalisch verfügbar in der Bib)
- *Machine Learning kompakt: Alles, was Sie wissen müssen*, Andriy Burkov, 1. Auflage.; 2019
  Link
- *Generative Deep Learning*, David Foster, 2019
  Link
- *Generative AI in the Software Development Lifecycle*, Tracy Trac Bannon, Phil Laplante, IEEE Computer, Dec. 2024
  Link
- *Self-adaptive software: Landscape and research challenges*, M. Salehie and L. Tahvildari, TAAS, vol. 4, no. 2, 2009
  Link
- *Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective*, Danny Weyns, Wiley, 2020
  Link

References

- *Reinforcement Learning: An Introduction*, R. S. Sutton and A. G. Barto, 2nd ed., MIT Press, 2018
  (nur physikalisch verfügbar in der Bib)
- *The Five Laws of SE for AI*, T. Menzies, IEEE Software, vol. 37, no. 01, pp. 81–85, 2020.
- *Hidden technical debt in Machine learning systems*, D. Sculley et al., 29th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15), Vol. 2. MIT Press, Cambridge, MA, USA, 2015

Own Work

- *Triggering Proactive Business Process Adaptations via Online Reinforcement Learning*, Andreas Metzger, Tristan Kley and Alexander Palm, 18th Int'l Conference on Business Process Management (BPM 2020), Sevilla, Spain, September 13-18, 2020
  Link
- *Online Reinforcement Learning for Self-Adaptive Information Systems*, Alexander Palm, Andreas Metzger and Klaus Pohl, 32nd Int'l Conference on Advanced Information Systems Engineering (CAiSE 2020), Grenoble, France, June 8-12, 2020
  Link
- *Software Engineering for ECS: Towards Dev-Ops-Adapt*, Andreas Metzger, Workshop on Software in Electronics, Components and Systems-based Digitisation, Virtual, May, 2021 (invited presentation)
  Link