# Data-Driven Software Engineering

Generative AI In Software Engineering

# About me

| | |
|---|---|
| 2011 - 2015 | **Ph.D. in Computer Science -** University of Insubria (Italy) |
| 2014 - 2015 | **Visiting Researcher -** University of Kaiserslautern and FRAUNHNOFER IESE (Germany) |
| 2015 - 2017 | **Post-doctoral Researcher -** Free University of Bozen-Bolzano (Italy) |
| 2018 - 2019 | **Post-doctoral Researcher -** Tampere University (Finland) |
| 2020 - 2022 | **Post-doctoral Researcher -** LUT University (Finland) |
| 2022 - 2024 | **Assistant Professor (tenure track) -** University of Oulu (Finland) |
| From 2024 | **Associate Professor (tenure track) -** University of Oulu (Finland) |
| 2011 - 2015 | **Co-founder/Project Manager -** OpenSoftEngineering s.r.l. |

## Introduction to Generative AI

# What is Generative AI?

- **Generative AI** refers to a category of artificial intelligence models designed to generate new data that resembles existing data. Unlike traditional AI, which focuses on classification or prediction, generative AI is about creating new and original content.

- These models **learn** the underlying patterns of data (such as text, images, or music) and use those patterns to generate similar content. By learning complex data distributions, they can create realistic, high-quality outputs.

- Examples of Generative AI Models Include generative Adversarial Networks (**GANs**), Variational Autoencoders (**VAEs**), and Transformer-based models like GPT.

## Generative AI vs. Traditional AI

- **Traditional AI**: Typically solves classification, regression, or clustering problems. For example, identifying objects in images, predicting future values, or grouping similar items.

- **Generative AI**: Focuses on creating new data that mimics the properties of the original data. Generative models are designed to deeply understand data distributions, allowing them to produce outputs similar to those in the training data.

- Traditional AI answers questions or provides decisions based on given data, while generative AI *produces new data*, providing creative abilities in applications like content generation, design, and entertainment.

# Why Generative AI is Important

- **Creative Applications**: Generative AI can automate creative tasks, such as designing artwork, composing music, or generating realistic photos. This has opened up new possibilities in fields like entertainment, digital art, and advertising.

- **Healthcare and Drug Discovery**: In fields like drug discovery, generative AI helps scientists simulate molecules and predict new drug formulations, speeding up the research process.

- **Data Augmentation**: Generative AI creates synthetic data to augment datasets. This is especially useful in domains with limited data, like medical imaging or autonomous driving, where gathering real-world data can be expensive or impractical.

- **Enhancing Human Creativity**: By collaborating with generative AI, humans can push creative boundaries. Artists, musicians, and writers can use AI to inspire new ideas and explore creative spaces that might not be accessible otherwise.

# Overview of Generative AI Techniques

- Popular Techniques: GANs, VAEs, and Transformers. Each technique has unique approaches for handling data and generating new content.
- GANs (Generative Adversarial Networks): Two-part model structure with a generator that produces fake data and a discriminator that distinguishes between real and fake data.
- VAEs (Variational Autoencoders): Models that encode input data into a compressed latent space and then decode it back, useful for generating smooth variations of data.
- Transformers: Attention-based models that process sequential data, making them crucial for language-based generative tasks like text generation and translation.
- Other Techniques: Includes Markov Chains and Recurrent Neural Networks (RNNs), which were popular in early generative models but are now mostly outpaced by newer architectures like transformers.

## Key Terms in Generative AI

- Latent Space: A lower-dimensional space where data is represented in compressed form. Models like VAEs utilize this space for generating new variations of input data.

- Distribution Learning: Generative models learn the distribution of the training data, allowing them to sample from this learned distribution and create similar new data.

- Stochasticity: Randomness is introduced in the generation process to ensure diversity in the outputs. For example, GANs produce slightly different images each time due to this stochastic process.

- Decoder: In models like VAEs, the decoder is responsible for reconstructing the input from its latent representation, which helps in generating new variations of the input data.

## GenAI Examples

# Text Generation with Generative AI

- Text generation models take a prompt or starting text and expand upon it to generate human-like paragraphs, articles, or dialogues.

- It is primarily based on **Transformer architectures**, such as GPT-3, which use attention mechanisms to produce coherent and contextually relevant text.

- Applications includes content creation, customer service, digital marketing, and even creative writing. Text generation models can generate entire articles, answer questions, or simulate conversations in chatbots.

## Image Generation with Models like DALL-E

- Models are trained on large datasets of images and their corresponding descriptions. They can generate images from textual prompts by learning the relationship between text and images.

- Applications: Useful in fields like advertising, where brands can generate custom visuals for marketing campaigns, or in entertainment for creating concept art.

- DALL-E, a model that can create highly detailed and creative images based on complex prompts, such as “a futuristic cityscape at sunset.”

## Music Generation with Models like MuseNet

- AI can compose original music by learning the structure of musical compositions, such as chord progressions, rhythm, and melody.

- These models can aid composers in generating new pieces, assist musicians in ideation, or provide background music for video games and movies.

- MuseNet by OpenAI, which can generate music in various styles and instruments.

## Types of Generative Models

# Overview of Generative Models

- Definition: Generative models are designed to produce new instances that resemble the data they were trained on. Unlike discriminative models, which classify or label data, generative models create data based on learned patterns.

- Main Types: The three most widely used types of generative models are:
  - Generative Adversarial Networks (GANs): Widely used for image and video generation.
  - Variational Autoencoders (VAEs): Useful for data compression and anomaly detection.
  - Transformers: Primarily used in natural language processing for text generation and translation.

- Other Types: Less common but foundational techniques include Markov Chains, which are probabilistic models, and Recurrent Neural Networks (RNNs), which handle sequential data but are less efficient than Transformers for large datasets.

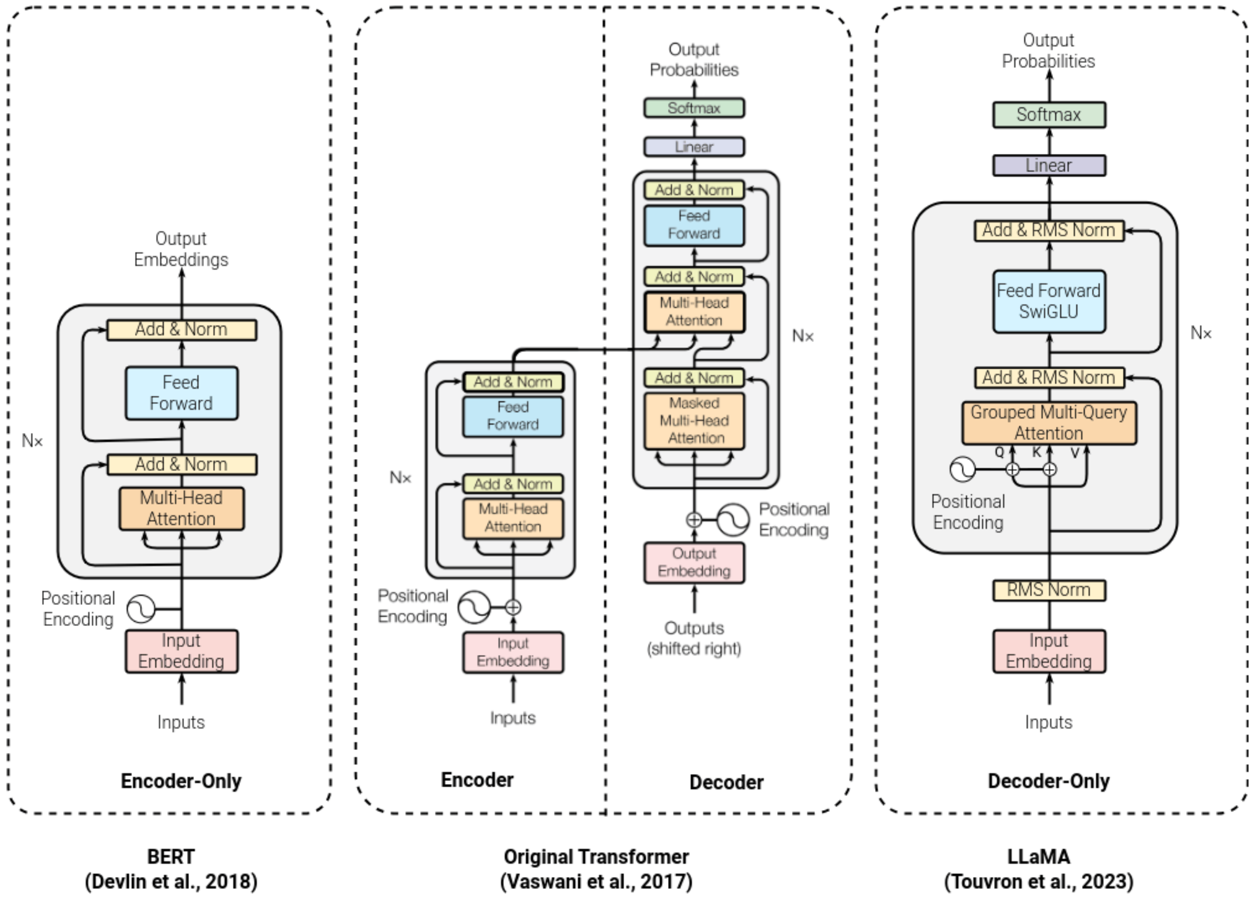
# Large Language Models

Yeah! Now is the time to talk about it!

# Introduction to Large Language Model

- Large Language Models (LLMs) are advanced AI models trained on massive datasets of text to generate and understand human language. They can handle tasks such as answering questions, summarizing text, translating languages, and even coding.

- Popular Models:
  - GPT-3 and GPT-4 by OpenAI: Known for generating coherent and contextually relevant text.
  - BERT by Google: Excel in understanding the context within text, widely used in search engines.
  - T5 by Google: Known for treating NLP tasks in a unified text-to-text format, making it versatile and adaptable.
  - LLaMA: General Purpose, Open-Source LLM from Meta



# Limitations of LLMs

- **Bias and Fairness**: Since LLMs learn from vast amounts of text data, they can inherit and amplify biases present in the training data. This can lead to problematic outputs, such as biased language or stereotypes. Mitigating bias in LLMs remains an ongoing research challenge.

- **Resource Intensive**: Training LLMs requires substantial computational power and energy, which can be costly and have environmental impacts. As models become more complex, their resource demands increase, making it essential to find efficient training methods.

- **Accuracy and Reliability**: Although LLMs can generate coherent responses, they sometimes produce incorrect or misleading information. This is because LLMs need help understanding language; they predict the most likely next word based on patterns. Therefore, they are prone to factual inaccuracies, which can limit their use in critical applications.

- **Interpretability**: LLMs function as black boxes, meaning how they arrive at specific outputs is often unclear. This lack of interpretability can be problematic in fields like healthcare or law, where understanding the reasoning behind a decision is crucial.

## LLMs in SE

## Code Generation and Completion

- How It Works: LLMs like GitHub Copilot use a model trained on vast amounts of code to assist developers by suggesting lines of code, completing function definitions, or automating boilerplate code.

- Benefits for Developers:
  - Time-Saving: Code completion speeds up the development process by reducing the need for typing and enabling developers to focus on more complex tasks.
  - Error Reduction: LLMs can help identify common syntax or logic errors, improving code quality and reducing bugs in the final product.
  - Learning Tool: Code generation tools can be educational, helping less experienced developers understand best practices and coding conventions.

- Limitations: While useful, LLM-generated code may contain security vulnerabilities, errors, or unconventional approaches. Developers must review and refine the generated code to ensure reliability and security.

# Code Review and Documentation

- **Automating Code Reviews**: LLMs can assist in code review by detecting potential issues, suggesting best practices, and providing feedback on code structure and readability. This can help maintain consistent quality across a codebase.

- **Documentation Generation**: LLMs can automatically generate documentation for functions, classes, and modules. They create comments or docstrings that make the code more understandable by analyzing code and identifying its purpose.

- **Enhanced Collaboration**: Automated documentation makes it easier for team members to understand each other’s code, improving collaboration.

- **Improved Readability**: Code comments and documentation clarify the intent behind complex code, aiding maintainability and simplifying future updates.

- While LLMs are powerful tools for generating code documentation, they may sometimes produce vague or inaccurate results if they misunderstand the code’s purpose. This underscores the importance of human oversight in ensuring that the documentation is clear and correct.

# Testing and Debugging with LLMs

- **Automated Test Generation**: LLMs can generate unit tests based on function definitions and expected outcomes, helping developers ensure that their code is thoroughly tested. For example, a model might generate multiple test cases for a sorting function, checking edge cases like empty or null inputs.

- **Debugging Assistance**: LLMs can analyze code and suggest possible reasons for errors. By identifying potential bugs and proposing fixes, they streamline the debugging process and reduce time spent troubleshooting.

- **Increased Test Coverage**: Automated test generation helps cover a wider range of scenarios, improving software reliability.

- **Faster Debugging**: LLMs provide insights into common errors and debugging tips, helping developers resolve issues more quickly.

- **Limitations**: While LLMs can assist in debugging, they may not always provide accurate suggestions for complex issues. Developers must use their expertise to validate and apply the model's recommendations.

# Integrating LLMs into DevOps

- **CI/CD Automation**: LLMs can be integrated into Continuous Integration and Continuous Deployment (CI/CD) pipelines to perform code assessments, security checks, and compliance validation. This helps automate quality assurance and ensures that code meets organizational standards before deployment.

- **Automated Code Assessments**: LLMs can analyze code changes, detect potential vulnerabilities, and flag issues before they are merged into the main codebase. This reduces the risk of introducing bugs or security flaws into production.

- **Streamlined Workflows**: Automating repetitive tasks reduces manual work, enabling DevOps teams to focus on higher-level tasks.

- **Improved Code Quality**: Regular code assessments ensure that code is secure, compliant, and optimized for performance, reducing the likelihood of production issues.

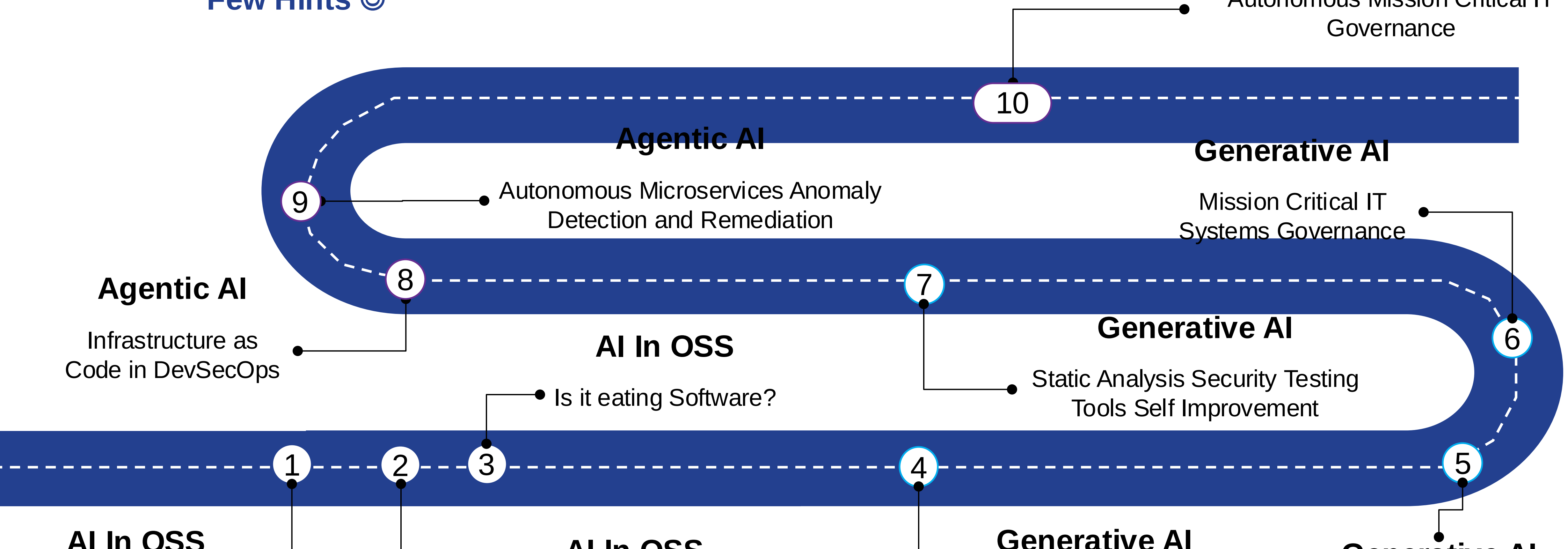
- **Challenges**: Integrating LLMs into DevOps requires careful configuration to ensure that assessments are accurate and that the model's recommendations align with organizational standards. Additionally, teams must monitor the LLM's outputs to avoid false positives or missed issues.

# Bridging (Gen)AI and Software Engineering

From Open Source Adoption to Mission-Critical Applications

# The Path Ahead

## Few Hints ☺



**AI In OSS**
Adoption and impact on development

**AI In OSS**
Impact on Technical Debt

**AI In OSS**
Is it eating Software?

**Generative AI**
Evidence-Based Software Engineering

**Generative AI**
Security Risk Analysis of Mission Critical IT Systems

**Generative AI**
Mission Critical IT Systems Governance

**Generative AI**
Static Analysis Security Testing Tools Self Improvement

**Agentic AI**
Infrastructure as Code in DevSecOps

**Agentic AI**
Autonomous Microservices Anomaly Detection and Remediation

**Agentic AI**
Autonomous Mission Critical IT Governance

## AI Libraries in OSS

### Adoption and Impact on Development (1/2)

- Open Source Software (OSS) emerged in the 1980s as a revolutionary alternative to proprietary software.

- Artificial Intelligence (AI) is increasingly present in OSS projects.

- Assess the adoption of AI libraries in Python and Java OSS projects.
  - Examine how AI libraries shape development in terms of:
  - Technical Ecosystem - Community Engagement

# AI Libraries in OSS

## Adoption and Impact on Development (2/2)

- Conducted a large-scale analysis of **6,323** OSS repositories.
- Scarce Adoption
  - AI libraries are not widely adopted in OSS projects.
- Enhanced Community Engagement:
  - Projects with AI libraries show more issues, pull requests, commits, and forks.
- Complex Technical Ecosystem:
  - Heightened dependency networks.
  - Increased workflow proliferation.

## AI Libraries in OSS

### Is AI Eating Software?

- Rapid integration of AI in software development brings opportunities and challenges.
- Technical debt is a critical issue affected by AI adoption.

- Investigate how AI usage contributes to or mitigates technical debt in OSS.

- Conduct an exploratory, large-scale analysis of GitHub projects.
- Focus on temporal trends and their impact on TD accumulation.

## AI Libraries in OSS

### Impact of Technical Debt

- Rapid integration of AI in software development brings opportunities and challenges.
- Technical debt is a critical issue affected by AI adoption.

- Investigate how AI usage contributes to or mitigates technical debt in OSS.

- Conduct an exploratory, large-scale analysis of GitHub projects.
- Focus on temporal trends and their impact on TD accumulation.

# Generative AI

## Evidence-Based Software Engineering

– Capabilities of Textual-GAI

- Allows researchers to explore new generative scenarios.
- Simplifies and accelerates time-consuming text generation and analysis tasks.

– Role of GAI in Evidence-Based Software Engineering

- Investigated and envisioned how GAI can support EBSE researchers.
- Working on empirically validating a comprehensive suite of models to effectively support EBSE researchers in managing literature reviews and data analysis.

# Generative AI

## Security Risk Analysis of Mission Critical IT Systems (1/2)

- Mission Critical Risk Analysis is time consuming and knowledge-hungry

- Assess the effectiveness of Large Language Models (LLMs) in mission-critical risk analysis, particularly Retrieval-Augmented Generation (RAG) and fine-tuned models.

- **What we did:**
  - Data collected from 50+ mission-critical analyses over five years, totaling 1283 samples.
  - LLMs (GPT-3.5, GPT-4, RAG and fine-tuned variations) vs. human experts in risk assessment.
  - Human experts provided both analysis and review.

## Generative AI

### Security Risk Analysis of Mission Critical IT Systems (2/2)

- Industrial Application:
  - RAG models can enhance risk assessment in mission-critical systems by rapidly surfacing hidden risks.
  - Fine-tuned LLMs are ideal for accuracy-focused scenarios.
- Future Work:
  - Further exploration into improving LLM accuracy for broader risk analysis applications.
  - Potential for training models on domain-specific countermeasures.
- Conclusion:
  - LLMs, particularly RAG models, offer valuable support in risk analysis by speeding up the process and uncovering risks missed by human experts.

# Generative AI

## Governance of Mission Critical IT Systems (1/2)

- Critical infrastructure security (healthcare, telecommunications, military coordination) is a fundamental concern, intensified by today's cyber warfare landscape.

- Importance of Mission-Critical Systems (MCSs):
  - Protecting MCSs is vital for national security.
  - These systems require prompt and comprehensive governance to ensure resilience.

- Challenges in Governance:
  - Recent events have highlighted increasing difficulties in meeting the demands of MCS security and governance.

# Generative AI

## Governance of Mission Critical IT Systems (2/2)

- Insights and Recommendations:
  - Interdisciplinary Collaboration is essential to safely integrate Large Language Models (LLMs) in MCS governance.
  - Researchers should focus on designing regulation-oriented models with an emphasis on accountability.
  - Practitioners should prioritize data protection and transparency.
  - Policymakers must establish a unified AI framework with global benchmarks to ensure ethical and secure LLM-based MCS governance.

## Agentic AI

### What?

- **Autonomy**: Empowers Generative AI to act and make decisions independently.
- **Adaptability**: Enables AI to interact with systems and adapt to changes.
- **Efficiency**: Creates self-managing systems that enhance performance and responsiveness.

# Agentic AI

## Infrastructure as Code in DevSecOps (1/2)

- Computing is deeply integrated into our daily lives through smartphones, smart homes, and connected cars.
- There’s a growing demand for development automation strategies to meet tight release schedules and rapidly deliver software projects.
- No existing studies focus on using GAI to generate IaC scripts based on DevSecOps stage artifacts.
- Different IaC tools require specific infrastructure setups for various project stages (testing, deployment, monitoring).

# Agentic AI

## Infrastructure as Code in DevSecOps (2/2)

- Envisioned Solution:
  - GAI models that utilize artifacts from each DevSecOps stage to create and refine IaC scripts.
- Impact for Practitioners:
  - Provides an automatic copilot for infrastructure design and deployment.
- Opportunities for Researchers:
  - A foundation for further empirical validation.
  - Potential to expand the possibilities enabled by this approach.

# Agentic AI

## Autonomous Microservice Anomaly Detection and Remediation (1/2)

- Microeservices offer unparalleled scalability and independent deployment in cloud computing.
- Decentralized nature introduces significant security and management challenges.
- Potential threats to system stability due to complexity and distribution.
- Quick detection and remediation to anomalies still requires a highly trained expert

## Agentic AI

### Autonomous Microservice Anomaly Detection and Remediation (2/2)

- Proposed Framework:
  - Based on MAPE-K (Monitor, Analyze, Plan, Execute over a Knowledge base).
  - Leverages Agentic AI for autonomous anomaly detection and remediation.
  - Addresses the challenges of managing highly distributed systems.
- Practical, Industry-Ready Solutions:
  - Maintains robust and secure microservices environments.
  - Customizable skeleton allows practitioners to:
  - Enhance system stability.
  - Reduce downtime.
  - Ensure continuous, efficient operations tailored to specific needs.