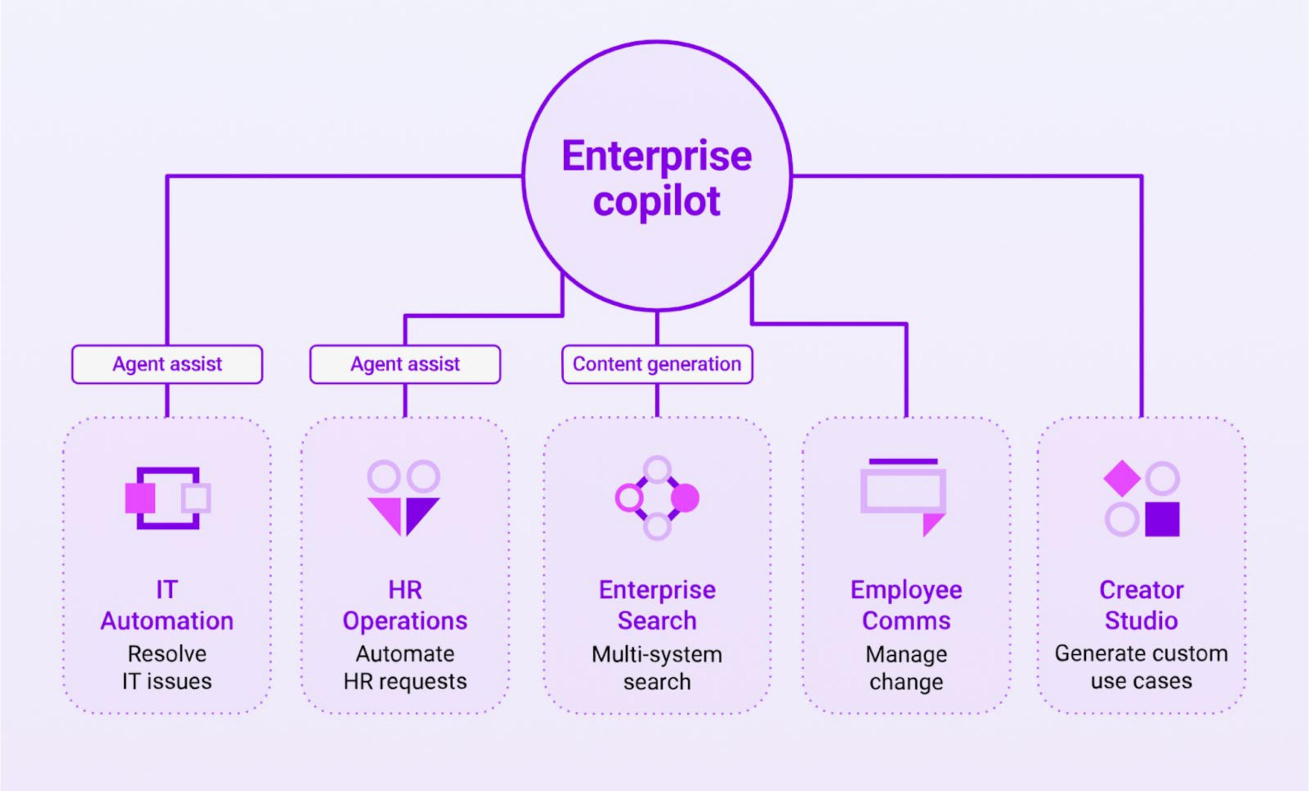
AI BASED COPILOT FOR (SOFTWARE) ARCHITECTS - RESEARCH STRATEGY INTRODUCTION

Dr. Marcus Zinn

Version 03.07.2024 – Frankfurt University of Applied Sciences

“AI based copilot for (software) architects”

I. Introduction and background



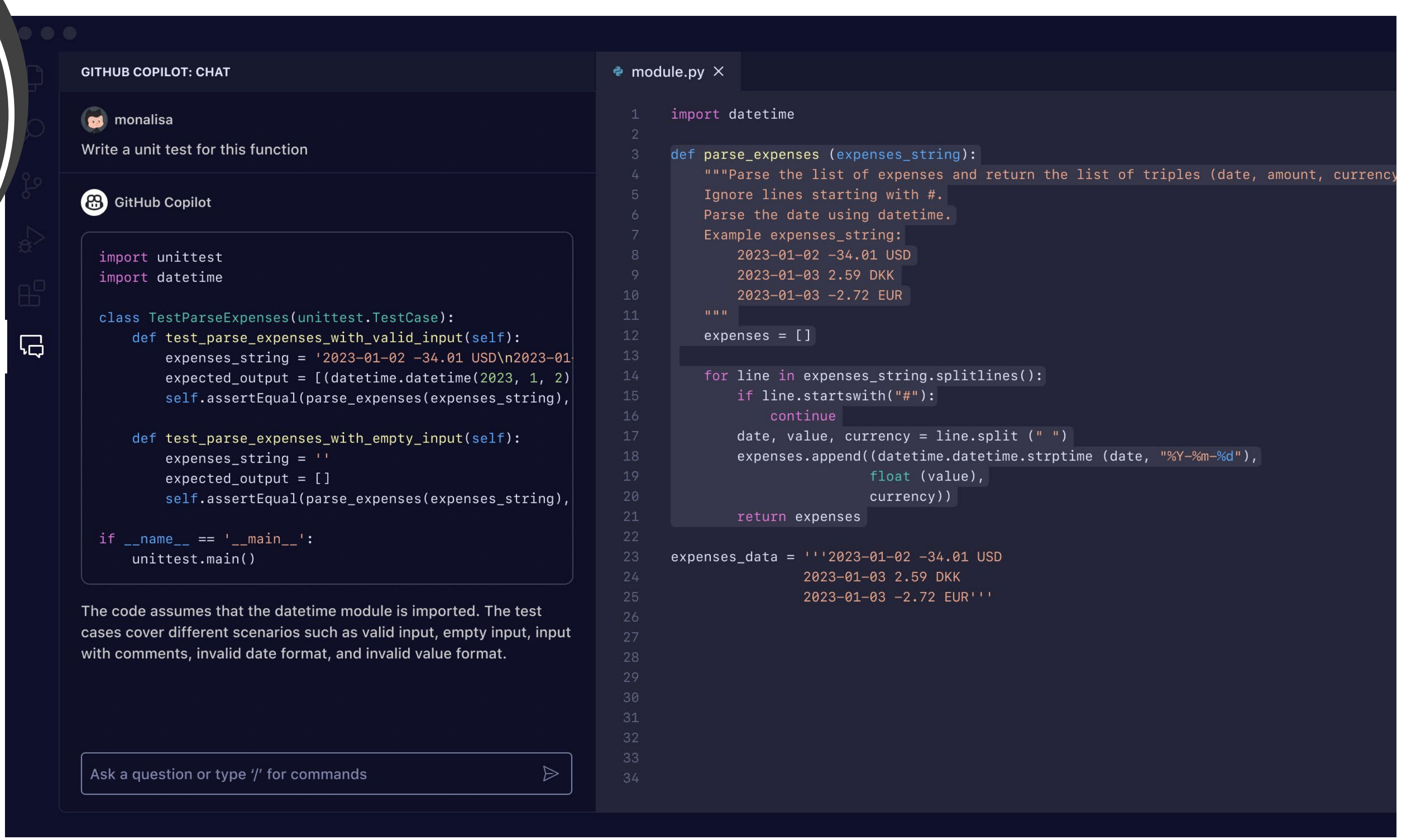
(moveworks,2023)

Introduction to AI and software architecture

What is AI?
What is AI Copilot?

AI, or Artificial Intelligence, is the technology that enables machines to perform tasks that usually require human intelligence, like learning, reasoning, and problem-solving.

EXAMPLE GITHUB COPILOT



monalisa

Write a unit test for this function

GitHub Copilot

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-
        expected_output = [(datetime.datetime(2023, 1, 2)
        self.assertEqual(parse_expenses(expenses_string),

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string),

if __name__ == '__main__':
    unittest.main()
```

The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format.

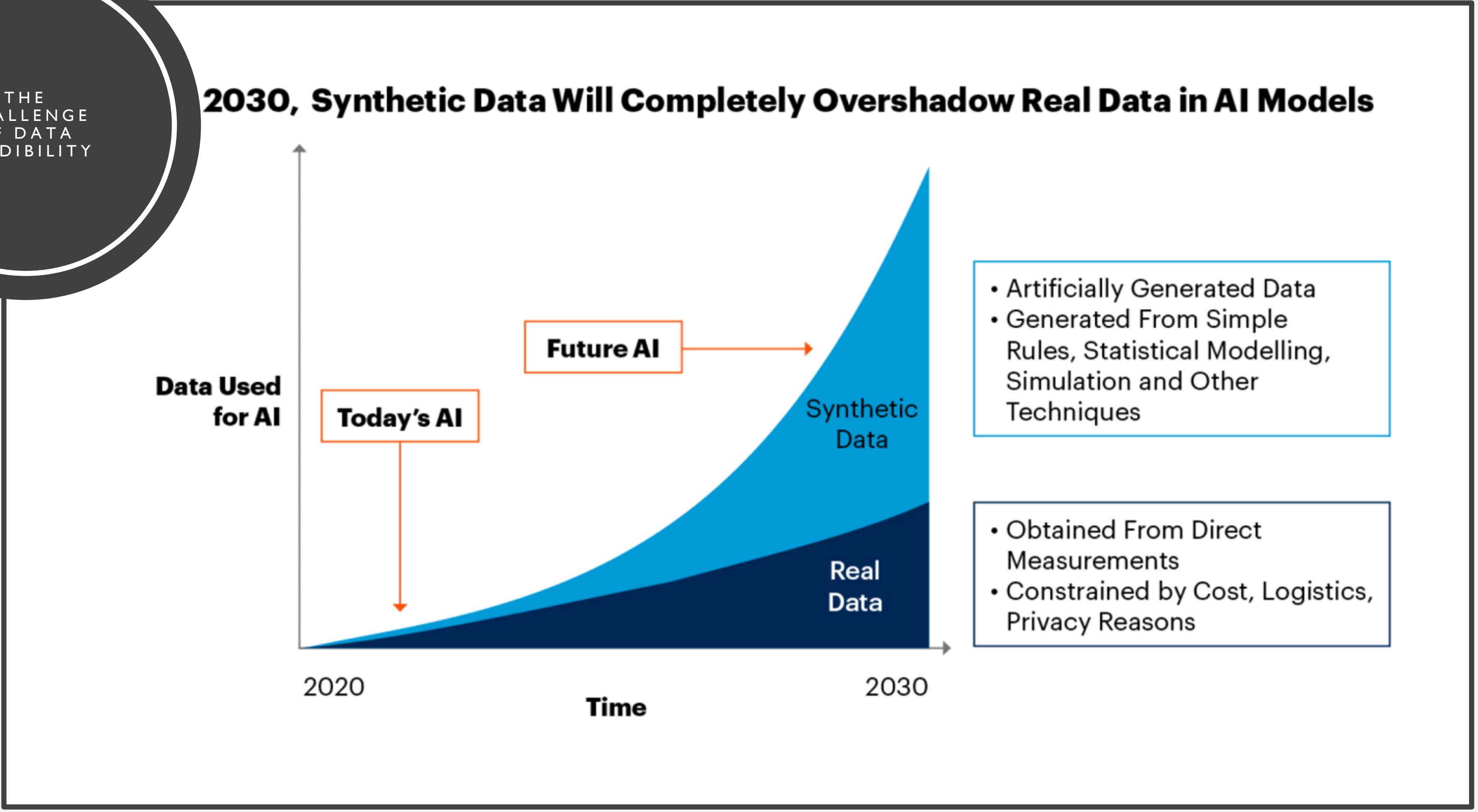
```
import datetime

def parse_expenses (expenses_string):
    """Parse the list of expenses and return the list of triples (date, amount, currency
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2023-01-02 -34.01 USD
        2023-01-03 2.59 DKK
        2023-01-03 -2.72 EUR
    """
    expenses = []

    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split (" ")
        expenses.append((datetime.datetime.strptime (date, "%Y-%m-%d"),
                        float (value),
                        currency))
        return expenses

expenses_data = '''2023-01-02 -34.01 USD
                2023-01-03 2.59 DKK
                2023-01-03 -2.72 EUR'''
```

THE CHALLENGE OF DATA CREDIBILITY

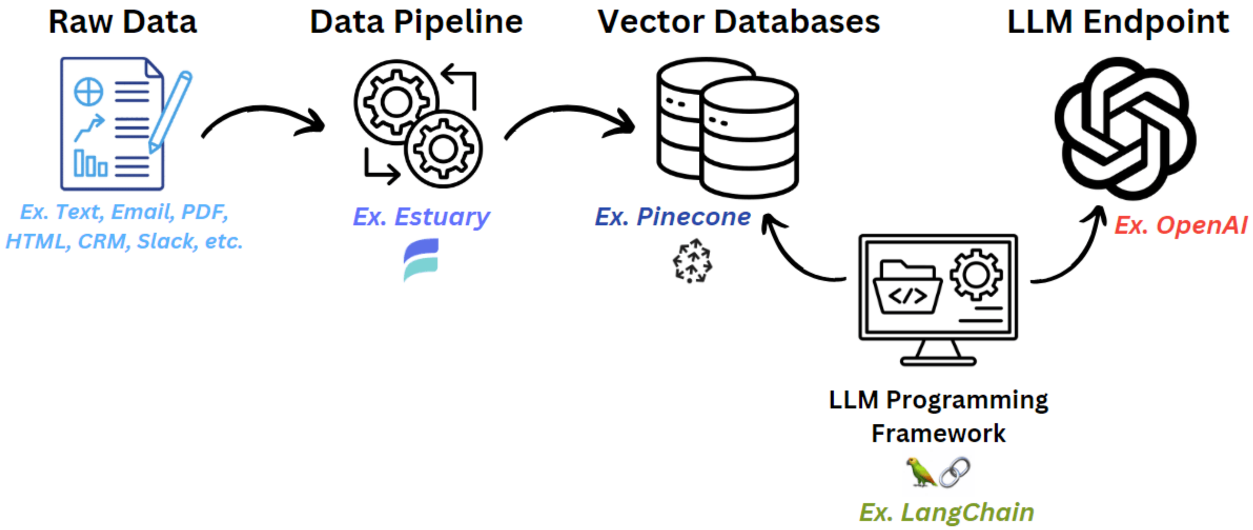


2030, Synthetic Data Will Completely Overshadow Real Data in AI Models

Source: Gartner 2021

NATURE
OF AI
TRAINING

The Modern LLM Tech Stack in 2023



Raw Data

Ex. Text, Email, PDF, HTML, CRM, Slack, etc.

Data Pipeline

Ex. Estuary

Vector Databases

Ex. Pinecone

LLM Endpoint

Ex. OpenAI

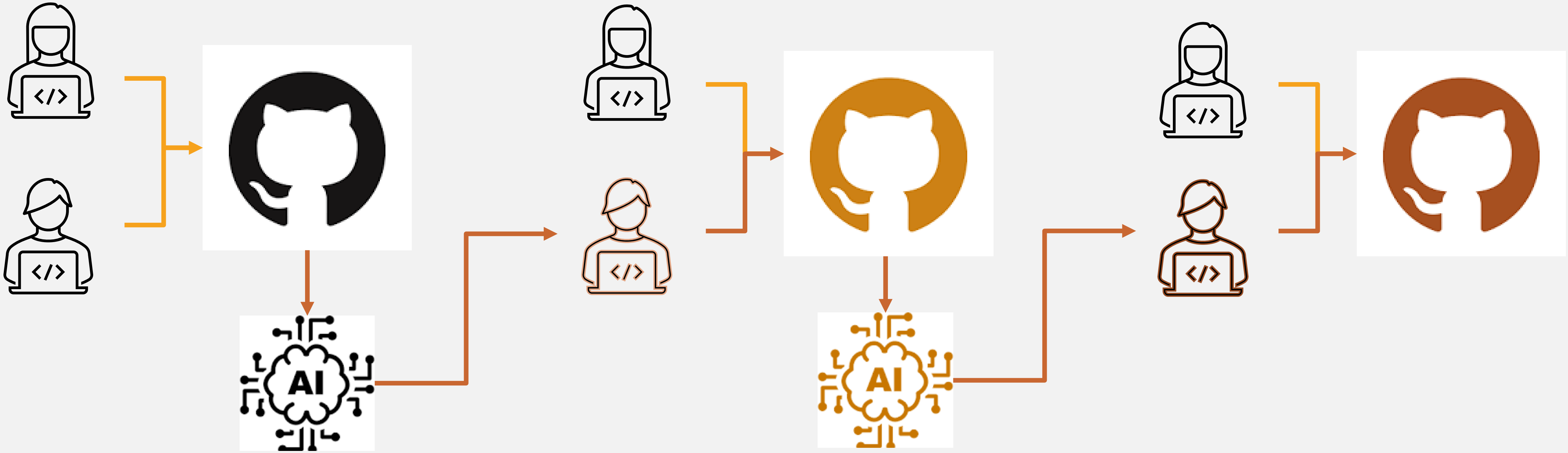
LLM Programming Framework

🦜🔗

Ex. LangChain

General

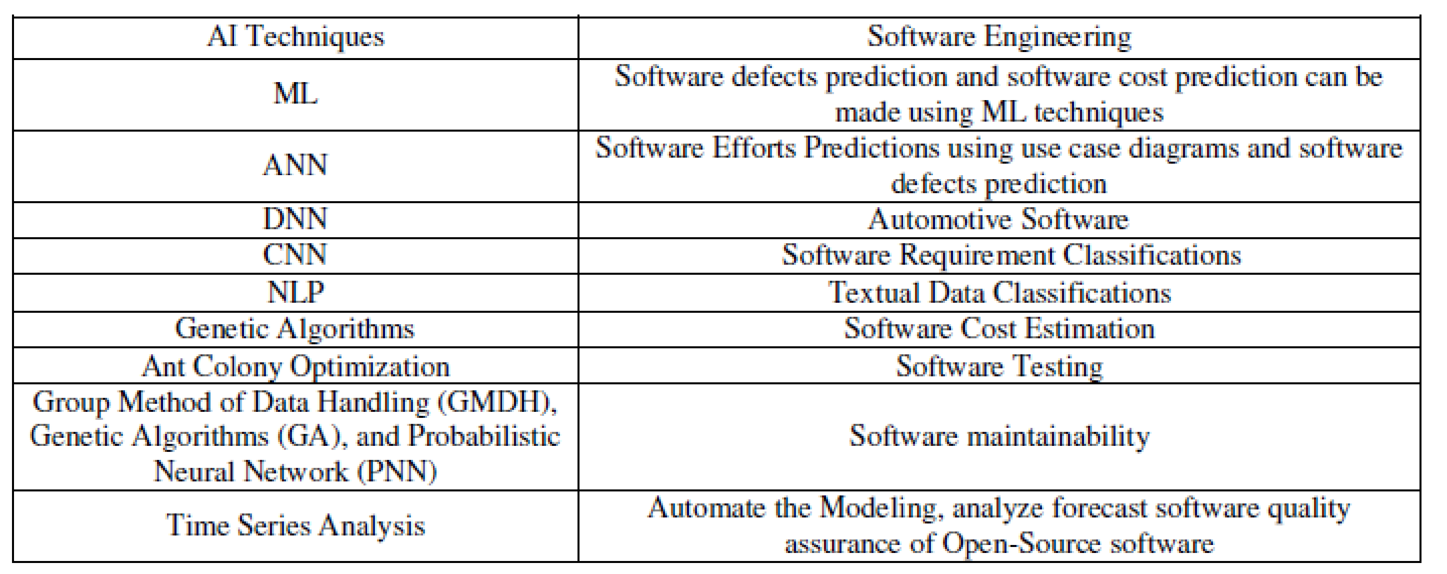
THE CHALLENGE OF DATA CREDIBILITY EXAMPLE GITHUB



INTRODUCTION TO AI AND SOFTWARE ARCHITECTURE
FAMOUS AI TOOLS OR LARGE LANGUAGE MODELS

- **Image generation AIs: Midjourney, Stable Diffusion, etc.**
- **Music generation AIs: Suno, example, etc.**
- **Coding support Ais: Copilot, example, etc.**
- **Mathematatial solver: Example, example, etc.**

RELEVANCE OF THE TOPIC
AI TECHNIQUES FOR SOFTWARE ENGINEERING



AI Techniques	Software Engineering
ML	Software defects prediction and software cost prediction can be made using ML techniques
ANN	Software Efforts Predictions using use case diagrams and software defects prediction
DNN	Automotive Software
CNN	Software Requirement Classifications
NLP	Textual Data Classifications
Genetic Algorithms	Software Cost Estimation
Ant Colony Optimization	Software Testing
Group Method of Data Handling (GMDH), Genetic Algorithms (GA), and Probabilistic Neural Network (PNN)	Software maintainability
Time Series Analysis	Automate the Modeling, analyze forecast software quality assurance of Open-Source software

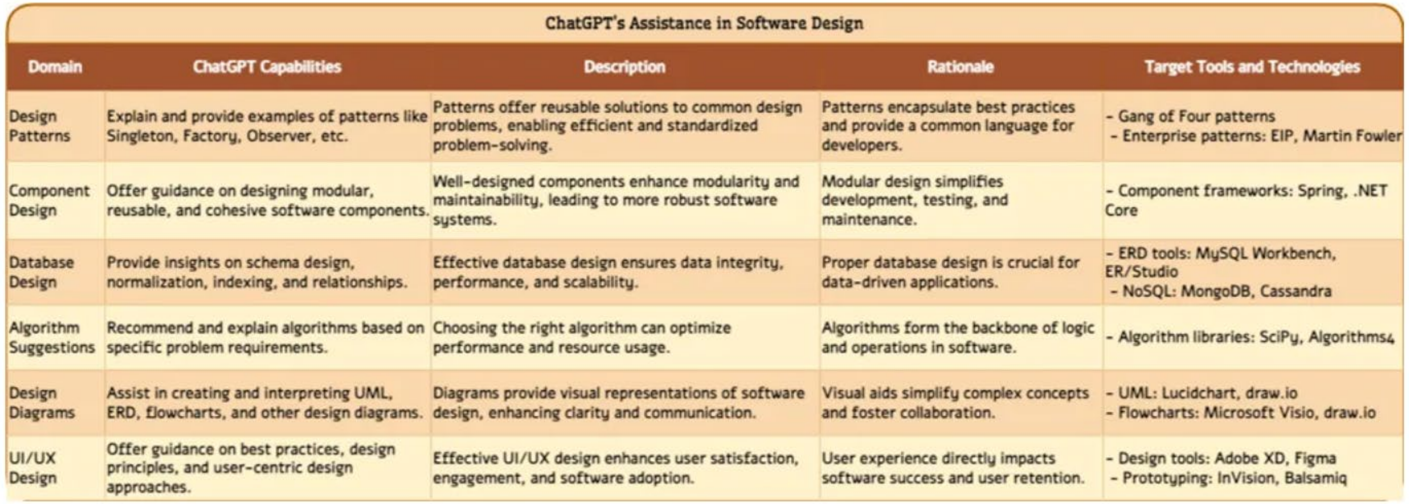
(Mustaqeem, et al., 2023)

General

Introduction to AI and software architecture - SoftwareArchitecture

- **Enterprise-Level Architectures:**
  - **Enterprise Architecture (EA):** Framework for aligning IT and business goals across an entire organization.
  - **Solution Architecture:** Detailed design of specific IT solutions within the enterprise context.
- **Infrastructure Architectures:**
  - **Infrastructure Architecture:** Design of physical and virtual IT infrastructure components.
  - **Cloud Architecture:** Design leveraging cloud services for scalability and flexibility.
- **System-Level Architectures:**
  - **Client-Server Architecture:** Model with clients requesting services from centralized servers.
  - **Peer-to-Peer (P2P) Architecture:** Distributed network with peers acting as both clients and servers.
  - **Distributed Architecture:** Components on different networked computers coordinating actions via messaging.
- **Software Architectures:**
  - **Application Architecture:** Structure and behavior of applications, including interaction patterns.
  - **Microservices Architecture:** Decomposed applications into loosely coupled, independently deployable services.
  - **Service-Oriented Architecture (SOA):** Services provided by application components via a communication protocol.
  - **Event-Driven Architecture (EDA):** Systems responding to events or state changes asynchronously.
  - **Layered (Tiered) Architecture:** Division into layers (e.g., presentation, business logic, data access) for modularity.
  - **Monolithic Architecture:** Traditional single-unit development with tightly coupled components.
- **Data Architectures:**
  - **Data Architecture:** Management and organization of data resources, including databases and data flow.

Relevance of the topic
AI in software design example



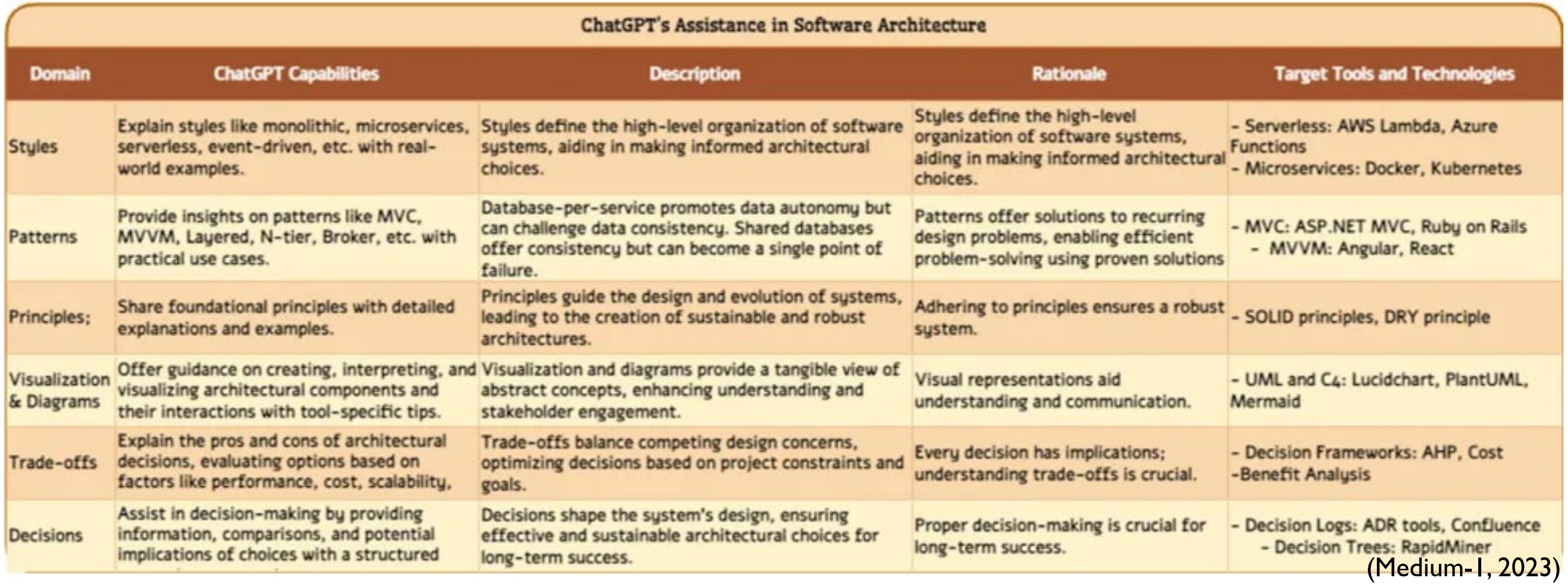
ChatGPT's Assistance in Software Design

Domain	ChatGPT Capabilities	Description	Rationale	Target Tools and Technologies
Design Patterns	Explain and provide examples of patterns like Singleton, Factory, Observer, etc.	Patterns offer reusable solutions to common design problems, enabling efficient and standardized problem-solving.	Patterns encapsulate best practices and provide a common language for developers.	- Gang of Four patterns - Enterprise patterns: EIP, Martin Fowler
Component Design	Offer guidance on designing modular, reusable, and cohesive software components.	Well-designed components enhance modularity and maintainability, leading to more robust software systems.	Modular design simplifies development, testing, and maintenance.	- Component frameworks: Spring, .NET Core
Database Design	Provide insights on schema design, normalization, indexing, and relationships.	Effective database design ensures data integrity, performance, and scalability.	Proper database design is crucial for data-driven applications.	- ERD tools: MySQL Workbench, ER/Studio - NoSQL: MongoDB, Cassandra
Algorithm Suggestions	Recommend and explain algorithms based on specific problem requirements.	Choosing the right algorithm can optimize performance and resource usage.	Algorithms form the backbone of logic and operations in software.	- Algorithm libraries: SciPy, Algorithms4
Design Diagrams	Assist in creating and interpreting UML, ERD, flowcharts, and other design diagrams.	Diagrams provide visual representations of software design, enhancing clarity and communication.	Visual aids simplify complex concepts and foster collaboration.	- UML: Lucidchart, draw.io - Flowcharts: Microsoft Visio, draw.io
UI/UX Design	Offer guidance on best practices, design principles, and user-centric design approaches.	Effective UI/UX design enhances user satisfaction, engagement, and software adoption.	User experience directly impacts software success and user retention.	- Design tools: Adobe XD, Figma - Prototyping: InVision, Balsamiq

ChatGPT's Assistance in Software Design

(Medium-1, 2023)

Relevance of the topic
AI in software architecture example



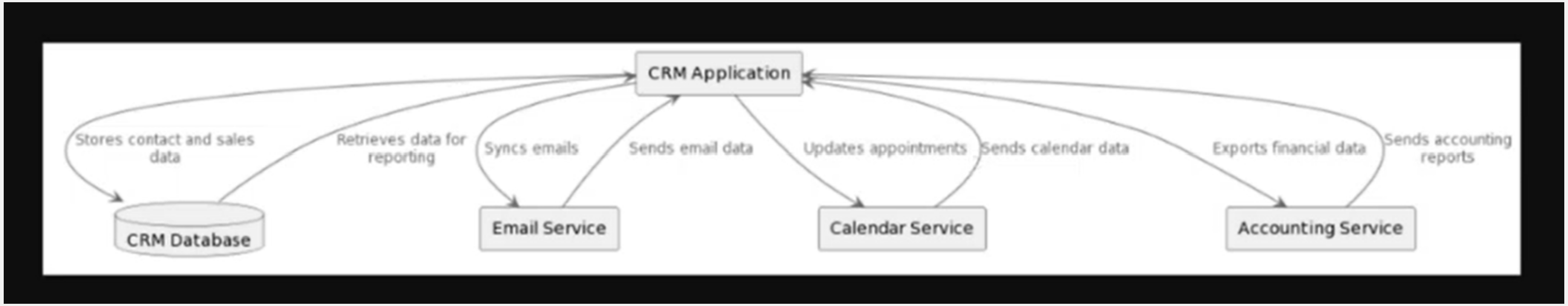
ChatGPT's Assistance in Software Architecture

Domain	ChatGPT Capabilities	Description	Rationale	Target Tools and Technologies
Styles	Explain styles like monolithic, microservices, serverless, event-driven, etc. with real-world examples.	Styles define the high-level organization of software systems, aiding in making informed architectural choices.	Styles define the high-level organization of software systems, aiding in making informed architectural choices.	- Serverless: AWS Lambda, Azure Functions - Microservices: Docker, Kubernetes
Patterns	Provide insights on patterns like MVC, MVVM, Layered, N-tier, Broker, etc. with practical use cases.	Database-per-service promotes data autonomy but can challenge data consistency. Shared databases offer consistency but can become a single point of failure.	Patterns offer solutions to recurring design problems, enabling efficient problem-solving using proven solutions	- MVC: ASP.NET MVC, Ruby on Rails - MVVM: Angular, React
Principles;	Share foundational principles with detailed explanations and examples.	Principles guide the design and evolution of systems, leading to the creation of sustainable and robust architectures.	Adhering to principles ensures a robust system.	- SOLID principles, DRY principle
Visualization & Diagrams	Offer guidance on creating, interpreting, and visualizing architectural components and their interactions with tool-specific tips.	Visualization and diagrams provide a tangible view of abstract concepts, enhancing understanding and stakeholder engagement.	Visual representations aid understanding and communication.	- UML and C4: Lucidchart, PlantUML, Mermaid
Trade-offs	Explain the pros and cons of architectural decisions, evaluating options based on factors like performance, cost, scalability,	Trade-offs balance competing design concerns, optimizing decisions based on project constraints and goals.	Every decision has implications; understanding trade-offs is crucial.	- Decision Frameworks: AHP, Cost -Benefit Analysis
Decisions	Assist in decision-making by providing information, comparisons, and potential implications of choices with a structured	Decisions shape the system's design, ensuring effective and sustainable architectural choices for long-term success.	Proper decision-making is crucial for long-term success.	- Decision Logs: ADR tools, Confluence - Decision Trees: RapidMiner

(Medium-1, 2023)

ChatGPT's Assistance in Software Architecture

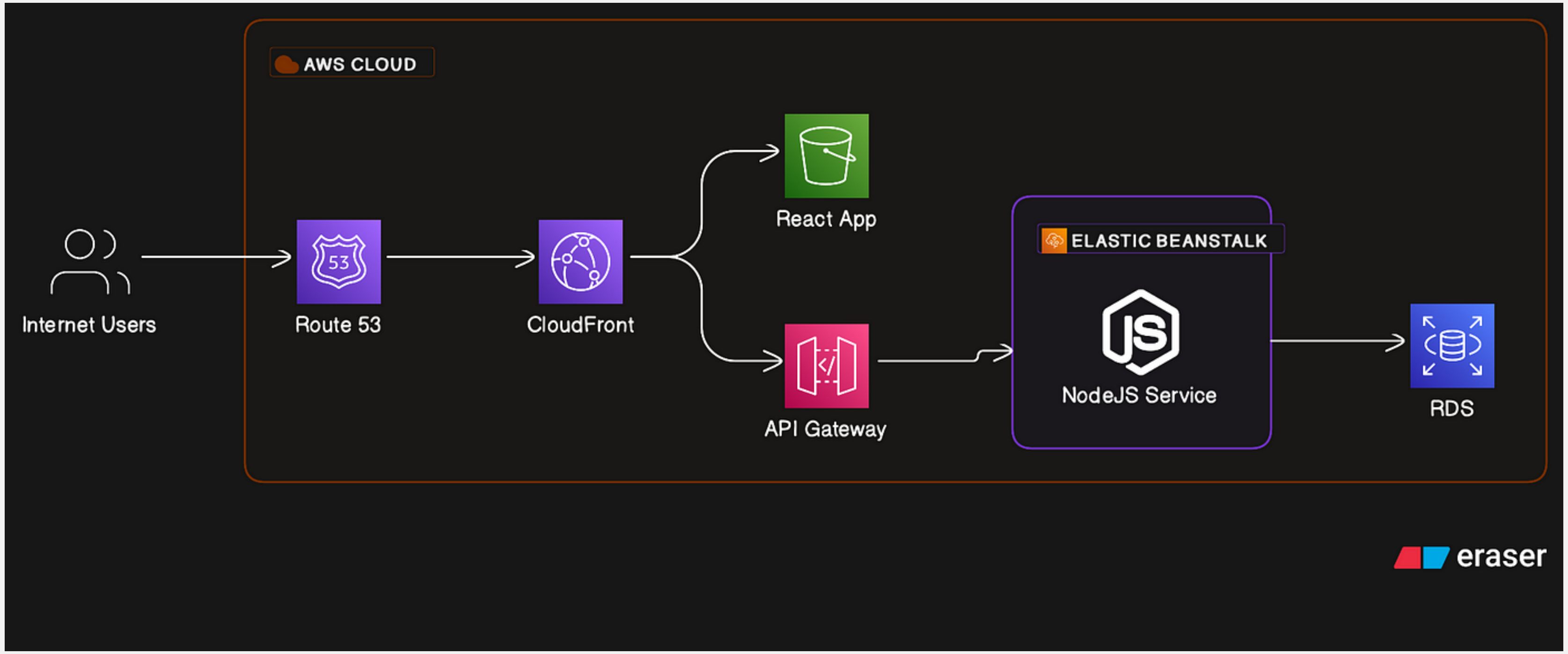
Use ChatGPT to create software architecture models diagram



Medium,2024

General

AUTO GENERATED DEPLOYMENT DIAGRAM USING DIAGRAM GPT



Medium, 2023

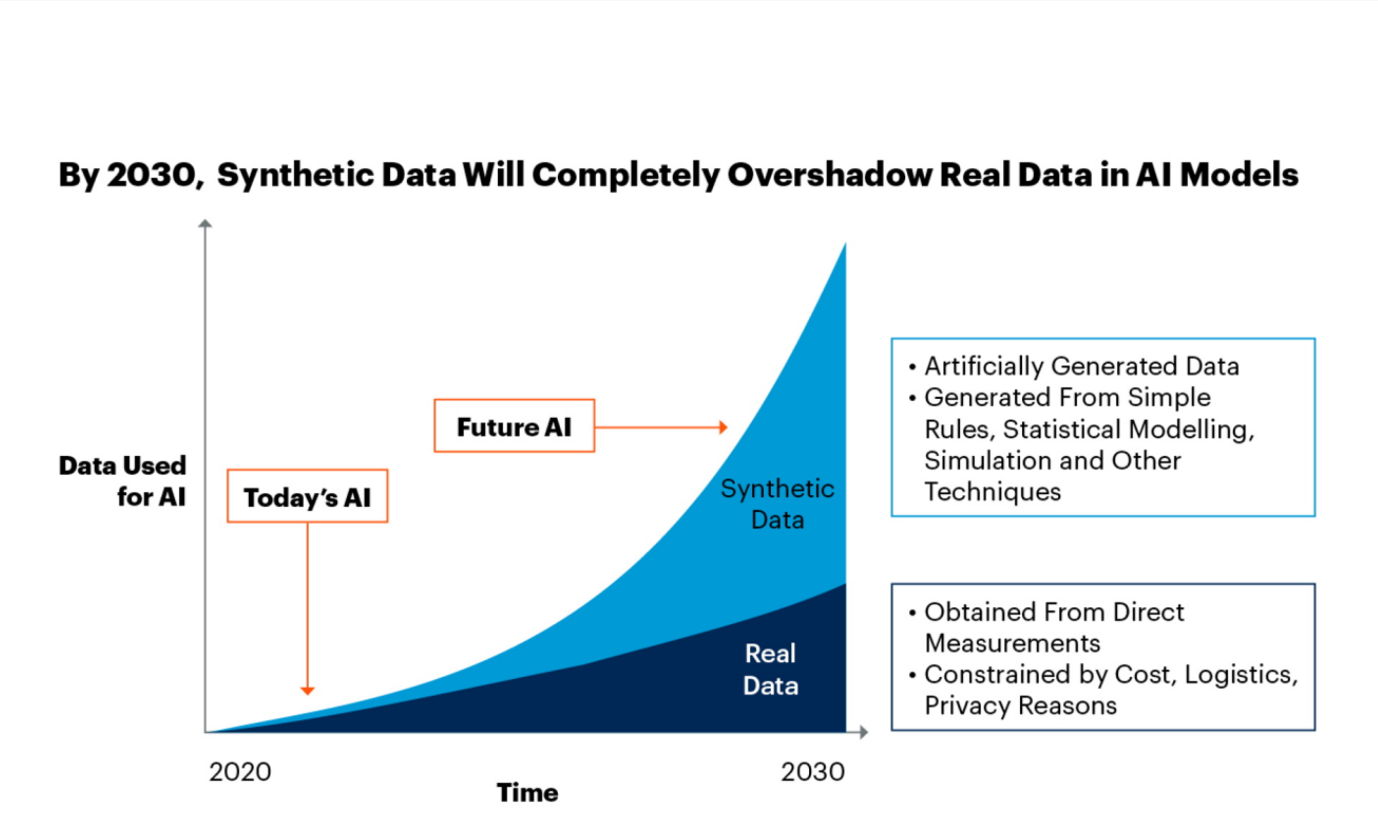
“AI based Copilot for (Software) architects ”

2. RESEARCH QUESTIONS AND OBJECTIVES

General

OVERALL RESEARCH QUESTION

How can AI support software architects without or with only short-term history?



Source: Gartner 2021

RESEARCH QUESTION DETAILS

- What specific challenges and requirements exist for software architects that can be supported by AI?
- Which AI methods and models are best suited to provide useful support without or with minimal historical data?
- How can we efficiently capture and process the necessary data and real-time information for the AI models?
- What role do human experts play in conjunction with AI systems in software architecture?
- How can we measure and evaluate the performance and benefits of the developed AI models in real projects?
- What ethical and practical implications arise from the use of AI in software architecture?

RESEARCH OBJECTIVES

- **Development of an AI model that supports software architects in real-time without decissions based on unsafe data.**
- **Evaluation of the effectiveness of this model in practical scenarios.**

“AI based copilot for (software) architects”

3. THEORETICAL FRAME

EXISTING APPROACHES

Reinforcement Learning (RL): Reinforcement Learning is a type of machine learning where an agent learns to make sequences of decisions in an environment to maximize cumulative rewards. The agent learns through trial and error, receiving feedback in the form of rewards or penalties. It aims to find the best possible policy (sequence of actions) to achieve a specific goal in a dynamic environment.

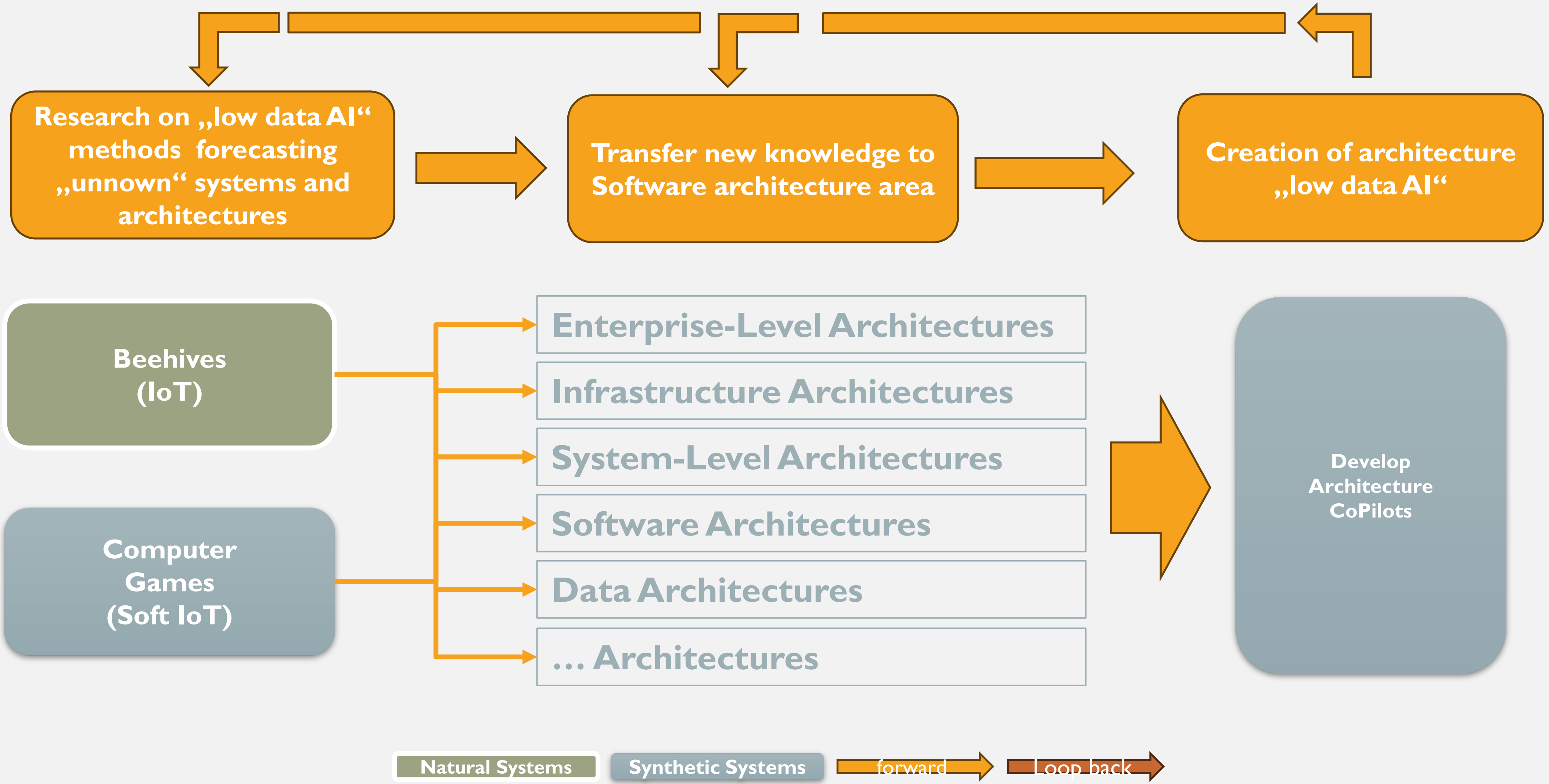
One-shot Learning: One-shot Learning is a machine learning approach where a model learns to recognize patterns from a single or a few examples, rather than requiring a large dataset for training. This is achieved by leveraging transfer learning, meta-learning, or similarity measures between data points. One-shot Learning is particularly useful in scenarios where obtaining large amounts of labeled data is impractical or expensive.

“AI based copilot for (software) architects“

4. METHODOLOGY OVERVIEW

General

METHODOLOGY
HIGH LEVEL



Research on „low data AI“ methods forecasting „unnown“ systems and architectures

Transfer new knowledge to Software architecture area

Creation of architecture „low data AI“

Beehives (IoT)

Computer Games (Soft IoT)

- Enterprise-Level Architectures
- Infrastructure Architectures
- System-Level Architectures
- Software Architectures
- Data Architectures
- … Architectures

Develop Architecture CoPilots

General

METHODOLOGY
HIGH LEVEL

Research on „low data AI“ methods forecasting „unnown“ systems and architectures

Beehives (IoT)

Computer Games (Soft IoT)

Data handling methods
- Data Cleaning
- Data Integration
- Data Reduction
- Data Normalization
- Feature Extraction
- Data Augmentation
- Outlier Handling

Analyse methods and models
- Reinforcement Learning
- Constraint Solving
- One Shot Learning
- Ect.

Used Data
- Real time values

Student support
- Bachelor thesis
- Master Thesi

Natural Systems **Synthtic Systems** forward Loop back

General

“AI based copilot for (software) architects”

5. FOCUSED RESULTS

General

EXPECTED RESULTS

- **Description of the expected performance improvements through the use of AI.**
- **Concrete examples of how software architects can be supported by the AI model.**

General

CONTRIBUTIONS TO RESEARCH

- **Inovative Approaches in Software Architecture**: Utilizing AI with limited historical data can pioneer new methods to assist architects in designing software systems.

- **Adaptability and Self-Organization**: Drawing inspiration from bee colonies, exploring how software architectures can adapt to evolving requirements and self-organize to enhance quality attributes.

- **Gaming and Simulation Environments**: Leveraging gaming concepts to simulate and validate complex architectural designs, offering robust testing grounds before practical implementation.

- **Optimization and Efficiency**: Applying AI techniques to solve optimization challenges in software architecture, such as selecting optimal designs considering multiple criteria or minimizing costs and resource consumption.

- **Interdisciplinary Insights**: Integrating insights from fields like bee colonies and gaming with software architecture to generate innovative, interdisciplinary solutions applicable in both theoretical and practical contexts.

General

REFERENCES

Reference	Title	Link
Gartner, 2021	Maverick Research: Forget about Your Real Data – Synthetic Data Is the Future of AI	https://www.gartner.com/en/documents/4002912
Estuary, 2023	EXTEND YOUR MODERN DATA STACK TO LEVERAGE LLMS IN PRODUCTION	https://estuary.dev/modern-data-stack-LLMs/
Medium 2023	Generative AI in Software Architecture: Don't Replace Your Architects Yet!	https://medium.com/inspiredbrilliance/generative-ai-in-software-architecture-dont-replace-your-architects-yet-cde0c5d462c5
Medium 2024	Use ChatGPT to create software architecture C4 models diagram	https://medium.com/@Neopric/use-chatgpt-to-create-software-architecture-c4-models-diagram-22ef29236a19
Medium-1, 2023	Software Architecture and Design in the Age of Generative AI: Opportunities, Challenges, and the Road Ahead	https://medium.com/oolooroo/software-architecture-in-the-age-of-generative-ai-opportunities-challenges-and-the-road-ahead-d410c41fdeb8
Movework,2023	What is an AI copilot?	https://www.moveworks.com/us/en/resources/blog/what-is-an-ai-copilot

General

AI BASED COPILOT FOR (SOFTWARE) ARCHITECTS - RESEARCH STRATEGY INTRODUCTION

Contact information

mail@marcuszinn.de

LinkedIn: Marcus Zinn

Dr. Marcus Zinn

Version 03.07.2024 – Frankfurt University of Applied Sciences