A CASE STUDY/PROJECT REVIEW IN CONSOLIDATING 180 MOBILE/CLOUD APPS INTO 3 IIOT APPS: EXPLORING CHALLENGES, REQUIREMENTS AND USE CASES FOR FUTURE AI-POWERED SOFTWARE ARCHITECTURE CO-PILOTS

Dr. Marcus Zinn

Business Roles

- 5,5 years mobile leader (IIOT Mobile Apps)
- 5 years leader for patents (focusing AI and Softwarepatents)
- 12 years software/system architecture
- .....

Hobbies

- Lecturer since 2004 (Mobile Apps, Informatics, Software Engineering,..)
- Support(Supervision for bachelor / master thesis : 28 (including 6 from THAB)
- IIOT Beekeeping
- AI based image generation (Stable difffusion / comfyUI)
- ....

Version 05.04.2024 – TH Aschaffenburg – University of applied science

Lecture Content

- Part 1 - Introduction
  - **Define Software Architecture**
  - **Examples about Software Architecture patterns**
  - **Requirements Best Practices for Software Architecture**

- Part 2 – Case study/project review analysis „180 apps into 3 apps“
  - **What happened? Are you serious?**
  - **Existing IIOT Apps and planned solutions**
  - **Challenges and used solutions**
    - Basic of mobile technologies
    - Challenges in mobile development
    - Used solutions/software architectures

- Part 3 - Outlook AI Powered SA
  - **AI Powered Software Engineering**
  - **AI Powered Software Architecture / Design Example**
  - **Prospective Role of AI in Software Architecture**

Introduction

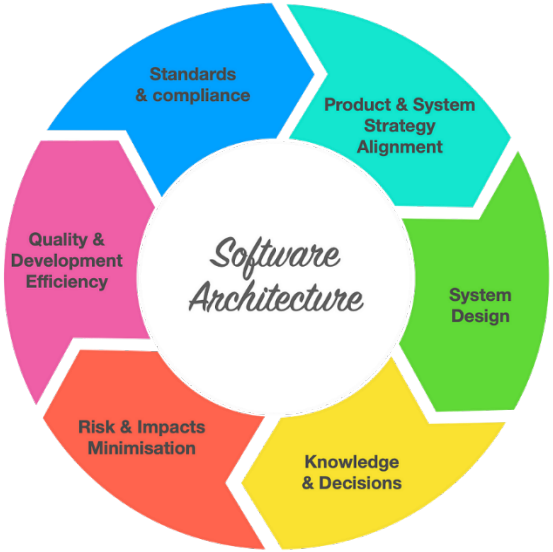
Purpose of the Lecture, and introduction to the topic and the lecturer

Introduction
Define Software Architecture

- “In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called architecture.”

  — Martin Fowler

- Software architecture typically involves:
  - Defining the software’s fundamental abstractions, modules, and interfaces.
  - Establishing guidelines and principles to ensure that the system is designed to meet its functional and non-functional requirements.



Software architecture (Medium, 2023)

Introduction
Define Software Architecture

Focusing InApp View

SOFTWARE DESIGN	SOFTWARE ARCHITECTURE
The process of creating a specification of a software artifact that helps to implement the software	The process of creating high level structures of a software system
Creates a software artifacts describing all the units of the system to support coding	Converts the software characteristics into high level structure
Creational, structural and behavioral are some software design patterns	Microservice, serverless and event driven are some software architecture patterns
Helps to implement the software	Architecture helps to define the high level infrastructure of the software

Focusing (High level) System View

Software Design vs.Software Architecture (algodaily, 2021)

Introduction
Define Software Architecture

Software Architecture Tools (NetSolutions, 2023)

Introduction
Define Software Architecture

Famous software architecture patterns

- Layered Architecture Pattern
- Event-driven Architecture Pattern
- Microkernel Architecture Pattern
- Microservices Architecture Pattern
- Space-based Architecture Pattern

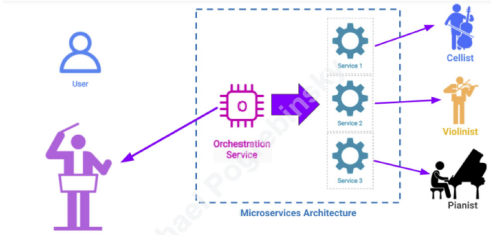
When to choose a Microservices Architecture Pattern?

- You're managing multiple corporate data centers with well-defined boundaries.
- You're building apps with immense and rapidly growing data systems
- You're developing new businesses and web applications quickly.
- You're re-writing monolithic applications to a more sustainable pattern.
- You're building Websites with small components.

- Cloud Architecure Patterns (Example for domain specific patterns)
  - **Scalability Patterns**
    - Load Balancing Pattern
    - Pipes and Filters Pattern
    - Scatter Gather Pattern
    - Execution Orchestrator Pattern
    - Choreography Pattern
  - **Performance Patterns for Data-Intensive Systems**
    - Map Reduce Pattern
    - 8The Saga Pattern
    - Transactional Outbox Pattern
    - Materialized View Pattern
    - CQRS Pattern
    - CQRS + Materialized View for Microservices Architecture
    - 13Event Sourcing Pattern
  - **Software Extensibility Architecture Patterns**
    - Sidecar & Ambassador Pattern
    - Anti-Corruption Adapter Pattern
    - 16Backends for Frontends Pattern
  - **Reliability, Error Handling and Recovery Software Architecture Patterns**
    - Throttling and Rate Limiting Pattern
    - Retry Pattern
    - Circuit Breaker
    - Dead Letter Queue (DLQ)
  - **Deployment and Production Testing Patterns**
    - Deployment Patterns:
    - Production Testing Patterns

Execution Orchestrator Pattern

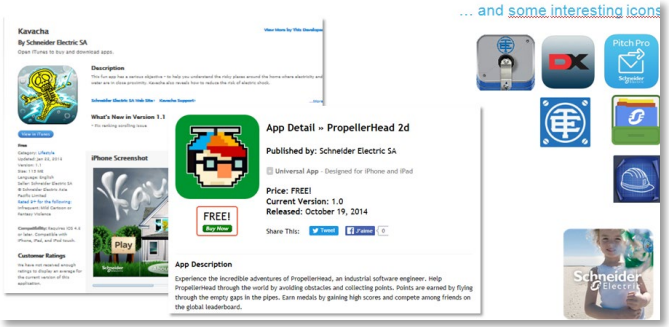
- **Problem:**
  - Running a complex flow of operations across multiple service/microservices
- **Solution:**
  - A centralized service acting as the "brain."
  - The only service aware of the
    - Context
    - Execution Steps
  - Responsible for handling issues and retries



Part 2 - Case study analysis „180 apps into 3 apps“
What happened? Are you serious?

- What happened ?
  - Apple called us

- Your 400 apps are
  - Partly not used
  - Partly the same
  - Partly not follow our rules
  - ....

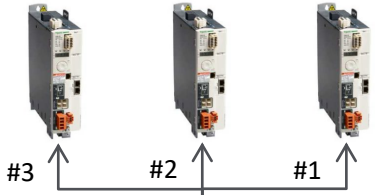


App Examples

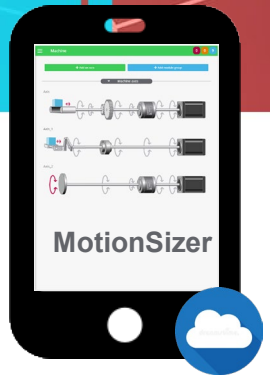
- Why is this a challenge?
  - Technology incompatible
  - No Funding
  - Project closed already
  - No Experts
  - No Documentation
  - Architecture incompatible
  - ....

- First solutions
  - Build a corporate group
  - Agreement with apple on consolidation plan

Part 2 - Case study analysis „180 apps into 3 apps“
Existing IIOT Apps and planned solutions
Challenge of functions

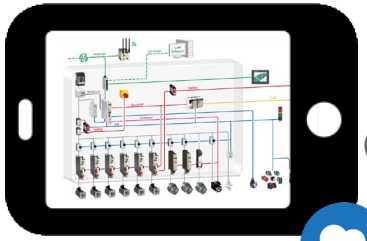


LED Blinking Scanner

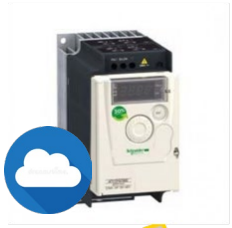


Drive Mobile KeyPad

Sercos ID



System Viewer (Netplan)



Read values from drives

Drive Diagnostic Tool

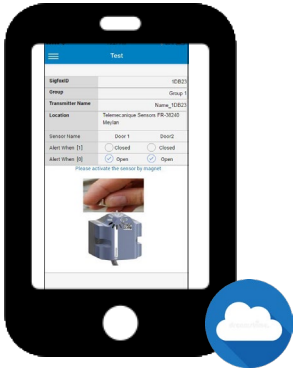
Share

Read values from PacDriveController

Share it

Mobile Diagnostic Tool and AR Diagnsotic Tool

Eliwell Connect

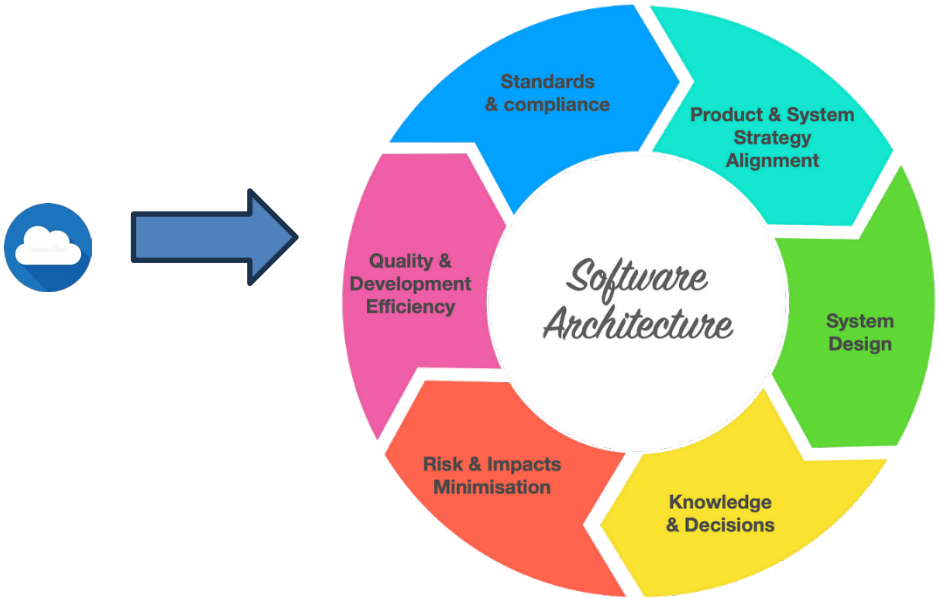


C2C & eXLhoist

Engine Plate Scanner

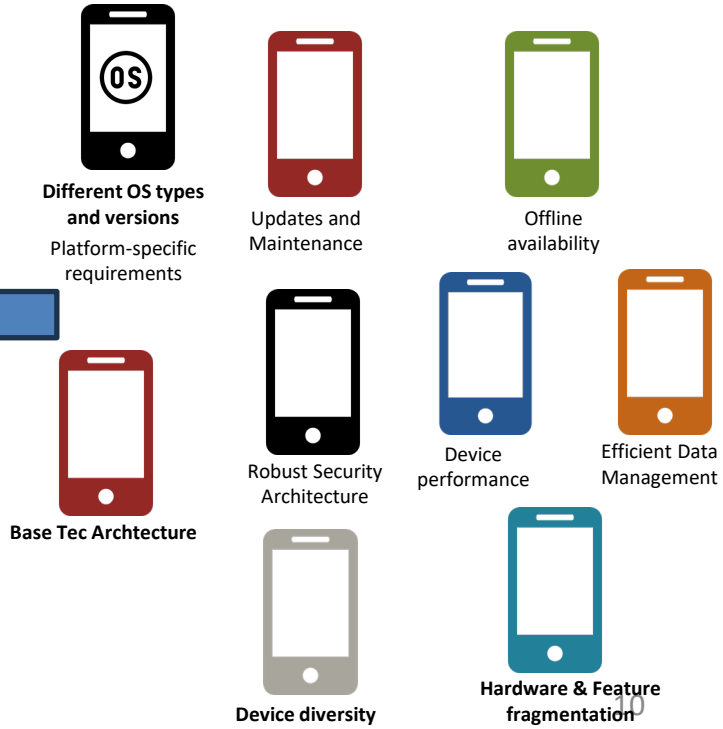
Part 2 - Case study analysis „180 apps into 3 apps“ Challenges

- Challenge: “Nature of Mobile”
- Solution Identify needs



Software architecture (Medium, 2023)

Mobile extens the challenge



Part 2 - Case study analysis „180 apps into 3 apps“ Challenges

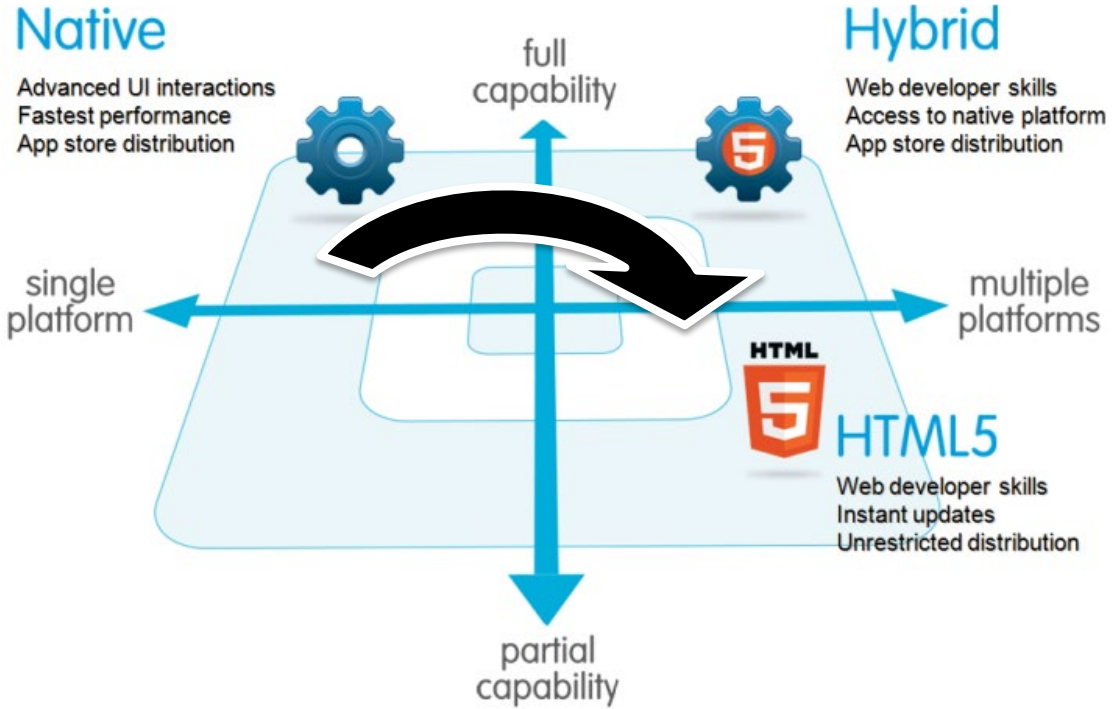
- Challenge – „Nature of mobile app types“
- Solution Identify needs

Mobile App Type	Pros	Cons
Native Apps	• Access to device-specific features • No internet connection needed • Impressive user experience on Android or iOS • The most responsive option • Efficiency driven by platform-specific features	• Higher development costs • Maintenance can be complicated • Native apps tend to be less discoverable than hybrid or web apps
Web Apps	• Easy discoverability • Faster development time than native • Simple to maintain • Lower development costs • No need to distribute software to machines that use the application • Updates made to the application and immediately available	• Slower performance than native • Unable to work offline • Not optimized for the given platform • Can't access platform features • Unable to access UI and match style like native apps
Hybrid Apps	• Less expensive to sell in app store • Simple to maintain • Lower development costs • Targets multiple platforms and reduces replication costs • Allows app to match look of native application and take advantage of the platform's UI style.	• Slower performance than native • Can't access platform features due to webview restrictions • Not optimized for the given platform

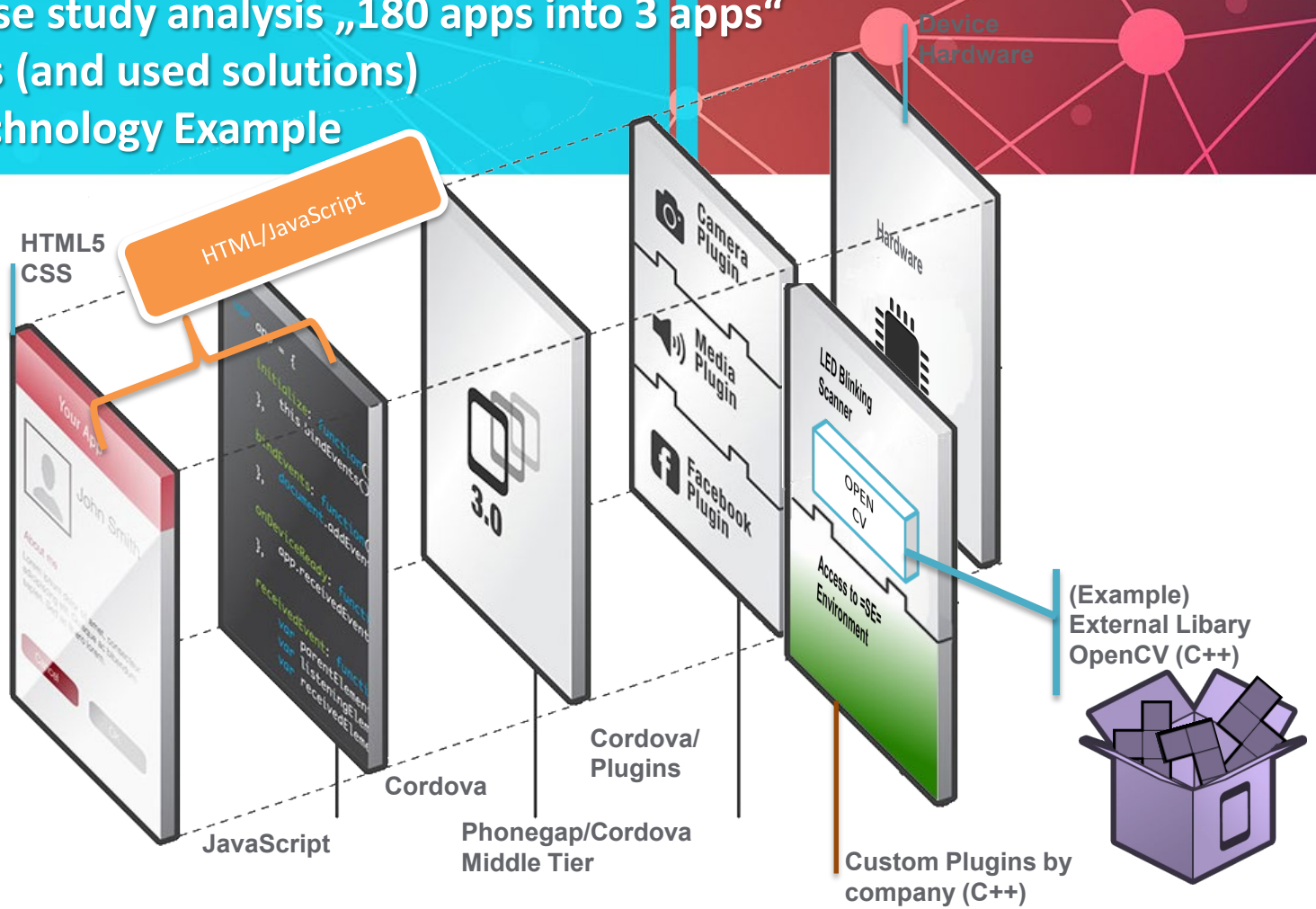
Source: ibm.com

BusinessofApps

Types of Mobile Apps
(Business of Apps, 2023)

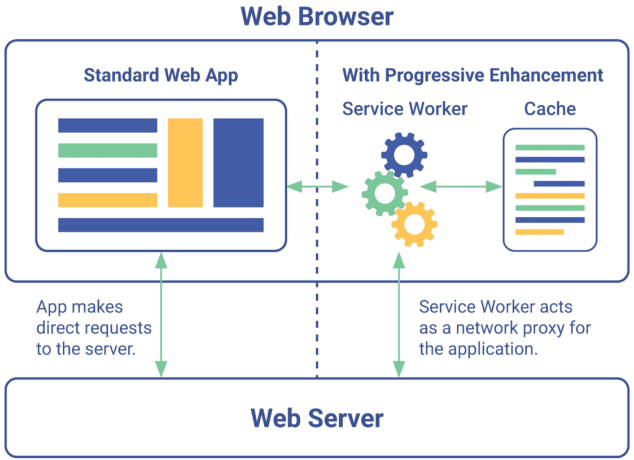


Part 2 - Case study analysis „180 apps into 3 apps“
Challenges (and used solutions)
Hybrid-Technology Example



Part 2 - Case study analysis „180 apps into 3 apps“
Challenges
Other used technology example

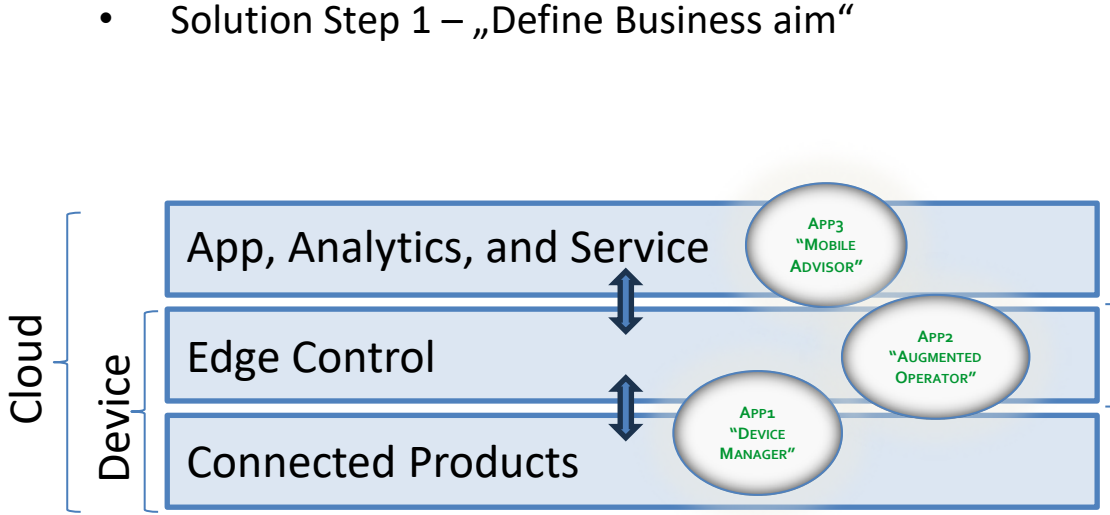
- Progressive Web Apps (PWAs)
- Integration of cloud-services
- Microservices architecture
- Augmented Reality (AR) and Virtual Reality (VR)

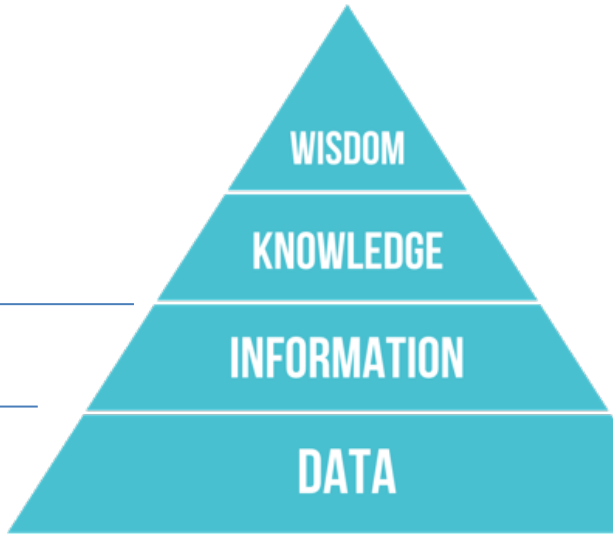


Progressive Web Apps Softwareplanetgroup, 2020)

Part 2 - Case study analysis „180 apps into 3 apps“
Used solutions

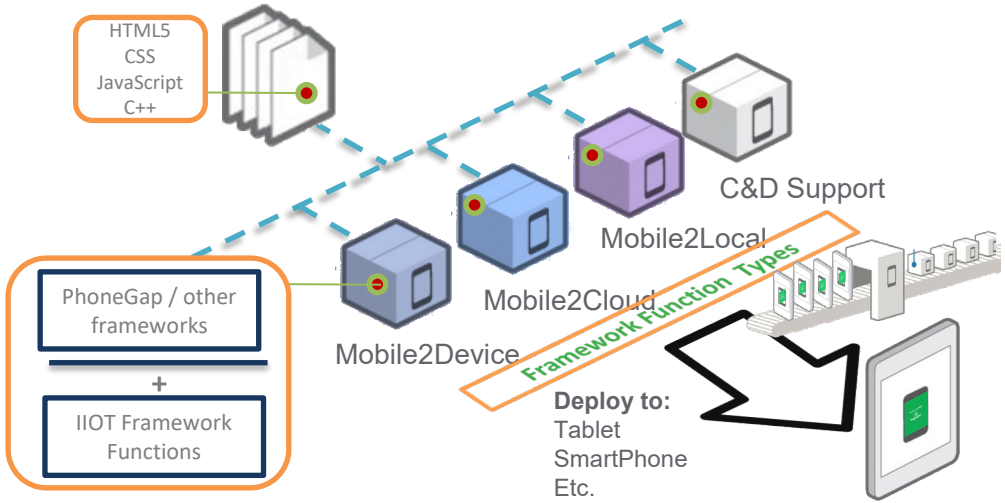
- Solution Step 1 – „Define Business aim“





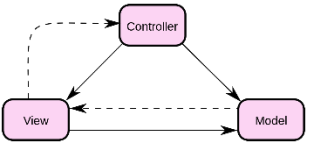
Part 2 - Case study analysis „180 apps into 3 apps“ Challenges and used solutions

- Solution Step 2 – „IIOT Framework (Mobile)“
  - Chose hybrid as base technology
  - Created all required mobile/IIOT features inside a hybrid framework

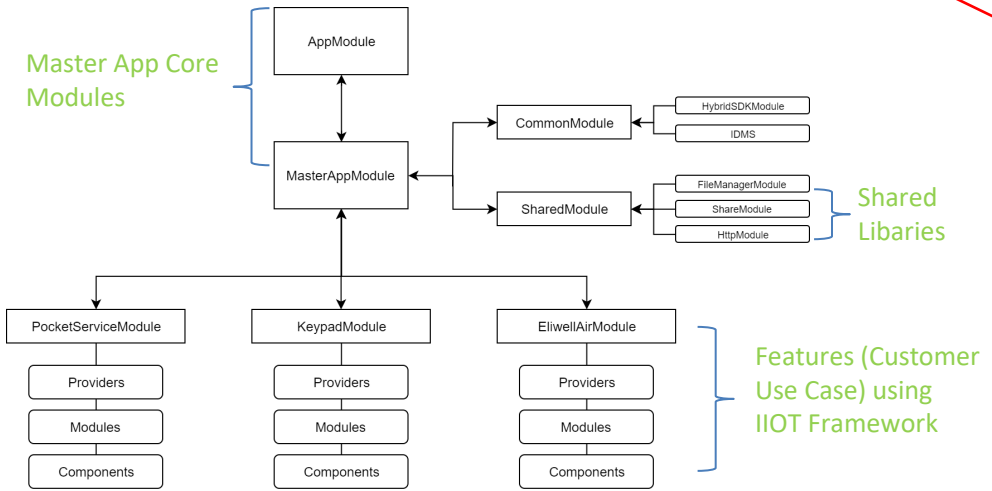


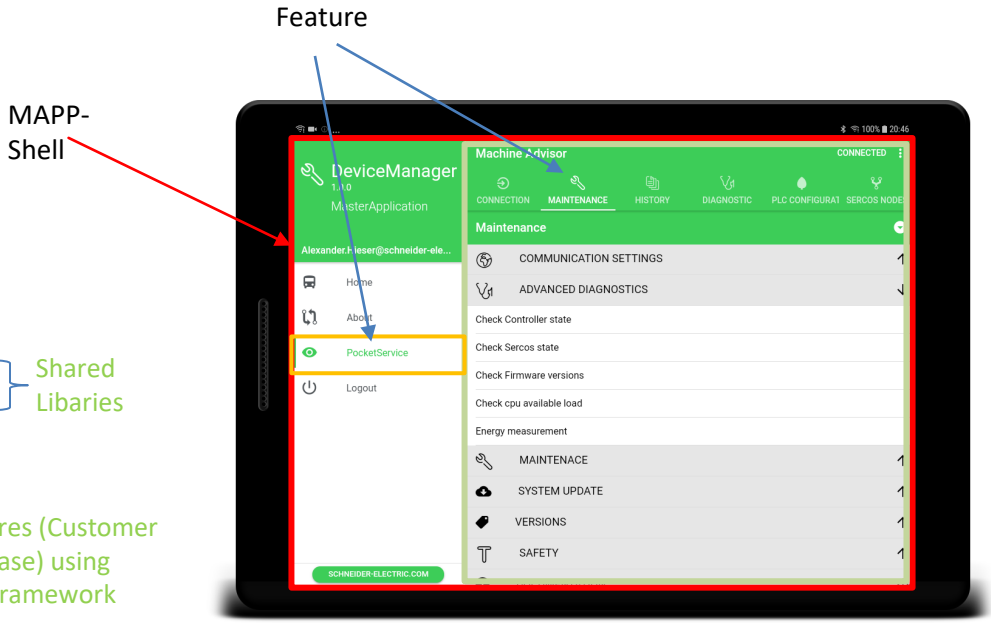
Part 2 - Case study analysis „180 apps into 3 apps“ Challenges and used solutions

- Solution Step 3b – „Mobile App Architecture Definition“



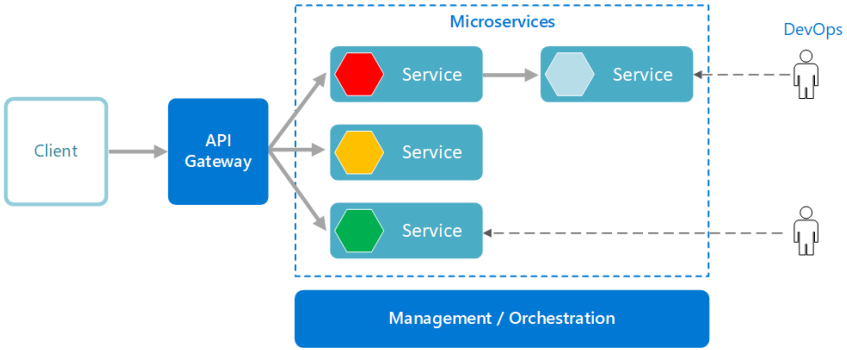
Model-View-Control



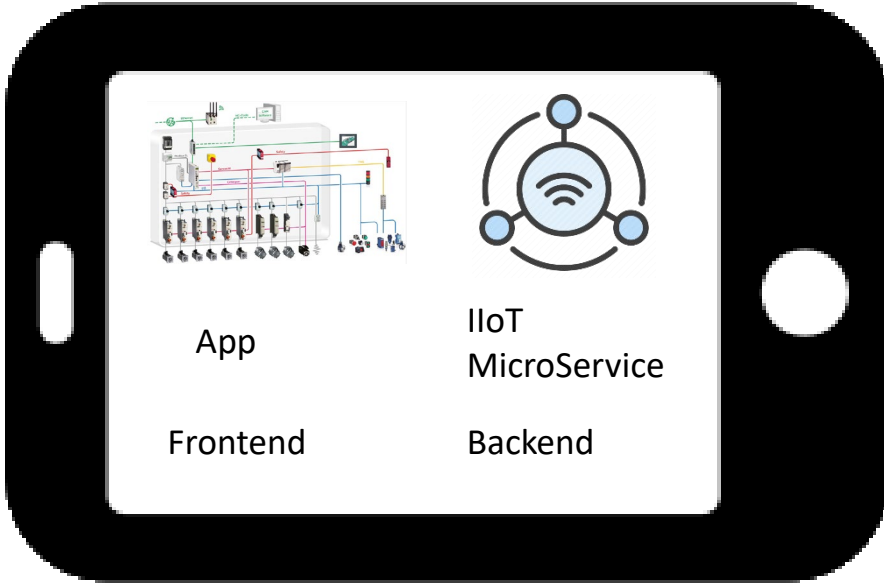


Part 2 - Case study analysis „180 apps into 3 apps“ Challenges and used solutions

- Solution Step 3c – „Advanced Mobile Architecture Definition“

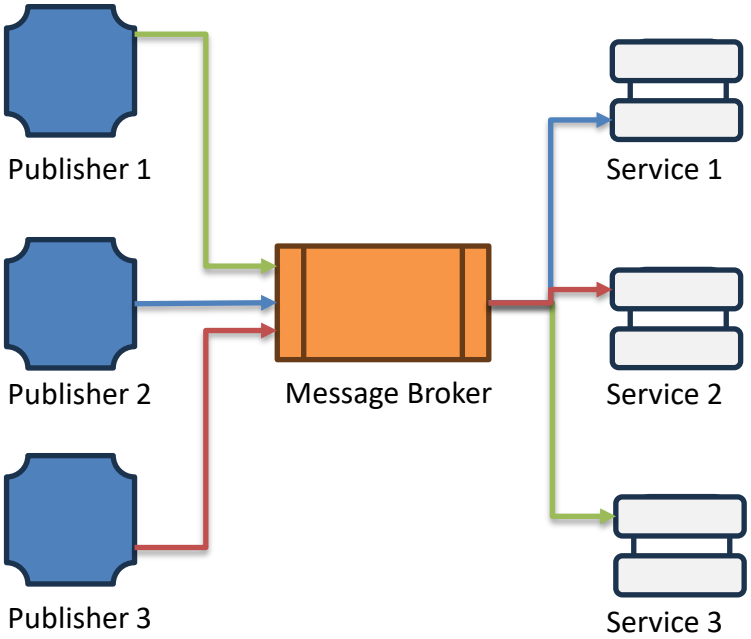


MicroService Architecture (Quelle: Microsoft, 2024)



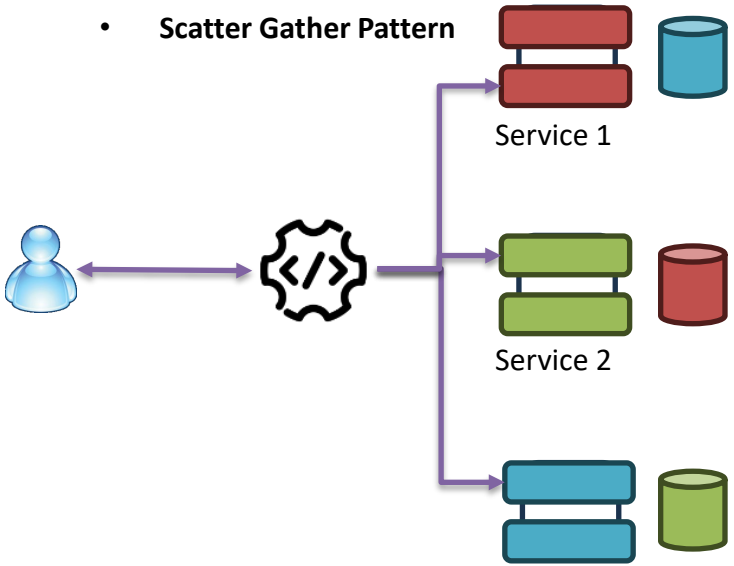
Part 2 - Case study analysis „180 apps into 3 apps“ Challenges and used solutions

- Solution Step 3d – „Cloud Architecture Definition“



- **Message Broker Pattern**

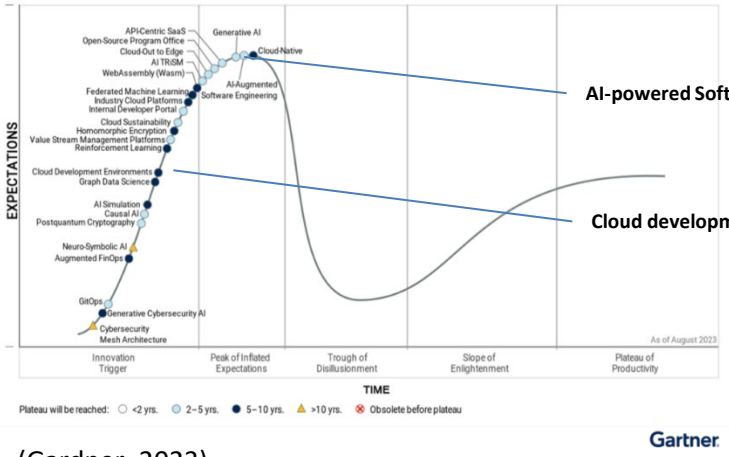
- **Scatter Gather Pattern**



(Udemy, 2024)

Outlook
AI Powered Software Engineering

Hype cycle for emerging technologies (2023)



AI-powered Software Engineering

Cloud development environment

(Gardner, 2023)

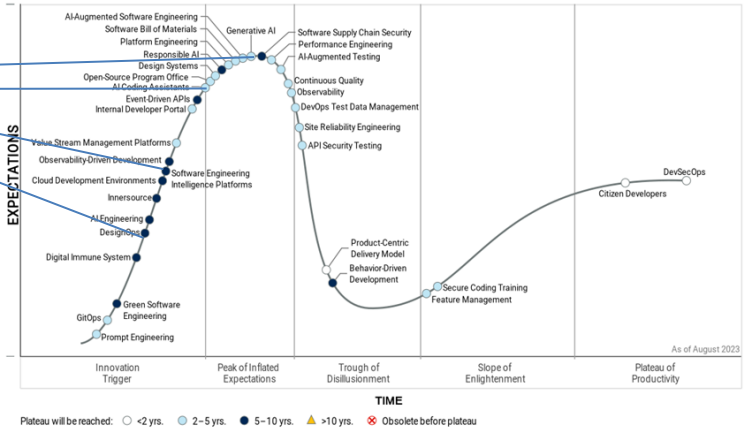
Hype Cycle for Software Engineering (2023)

AI-Augmented Software Engineering

AI-Coding Assistant

Software Engineering Intelligence Platform

DesignOps



(Gardner-1, 2023)

Outlook
AI Powered Software Engineering

(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)

Outlook
AI powered software design

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)

Outlook
Present AI Powered Software Architecture

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

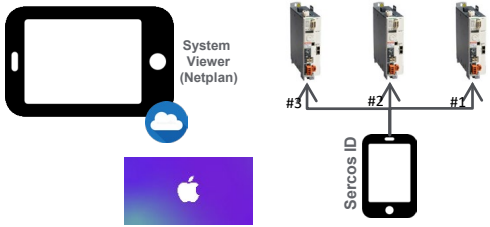
(Medium-1, 2023)

ChatGPT's Assistance in Software Architecture

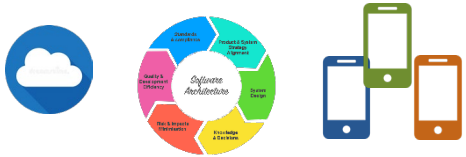
Outlook
Present AI Powered Software Architecture

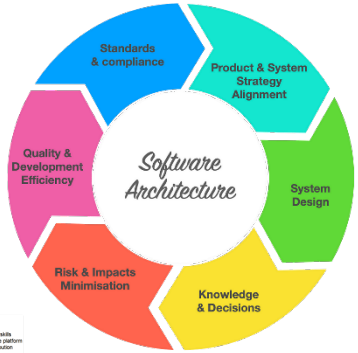
- What we expect from AI power software architecture based on the case study

1. Take care about use cases (including consolidation)

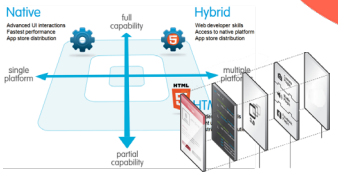


2. Take care about technology domain constraints

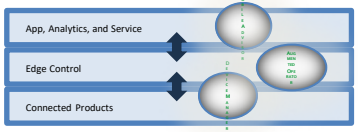




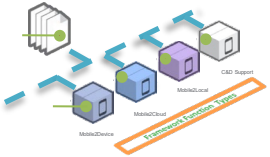
3. Include constraints by domain technology



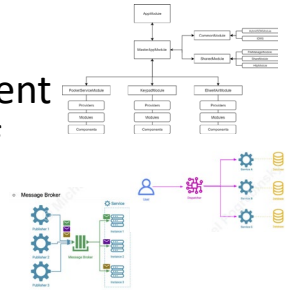
4. Include user domain solutions



5. Present alternatives



6. Work on different level and parts of the solutions



Outlook
Prospective Role of AI in Software Architecture

- Future of Generative AI in Software Architecture and Design
  - **Rise of multi-modal generative AI models**
  - **Integration of generative AI with other technologies**
  - **Democratization of generative AI**

- Potential Shifts in Architectural Practices in Light of Generative AI
  - **A greater focus on AI-driven design and optimization**

  Generative AI tools can be used to automate many tasks that are currently performed manually by software architects. This could lead to a greater focus on AI-driven design and optimization.

  - **A new generation of AI-savvy software architects**

  The next generation of software architects will need to have a good understanding of AI and how to use it to improve their work. Software architecture education will need to adapt to reflect this new reality.

A CASE STUDY IN CONSOLIDATING 180 MOBILE/CLOUD APPS INTO 3 IIOT APPS: EXPLORING CHALLENGES, REQUIREMENTS AND USE CASES FOR FUTURE AI-POWERED SOFTWARE ARCHITECTURE CO-PILOTS

Presentation available:

Contact information
Dr. Marcus Zinn
mail@marcuszinn.de
www.linkedin.com/in/marcuszinn

References

Reference	Title	Link
(Medium, 2023)	*Software architecture*	*https://blog.bitsrc.io/software-architecture-principles-rules-styles-c84b39db1421?gi=52503e4c1297*
(algodaily, 2021)	*Software Design vs.Software Architecture*	*https://algodaily.com/lessons/software-architectural-patterns-design-structures*
(NetSolutions, 2023)	*Software Architecture Tools*	*https://www.netsolutions.com/insights/why-software-architecture-matters-to-build-scalable-solutions/*
(Business of Apps, 2023)	*Types of Mobile Apps*	*https://www.businessofapps.com/app-developers/research/types-of-mobile-apps/*
(Medium 2017)	*Platform Types*	*https://medium.com/@seo.supertroninfotech/modern-apps-native-apps-vs-hybrid-apps-vs-html5-1f24b883cd86*
Softwareplanetgroup, 2020)	*Progressive Web Apps*	*https://softwareplanetgroup.co.uk/dawn-of-the-progressive-web-application/*
(Quelle: Microsoft, 2024)	*MicroService Architecture*	*https://learn.microsoft.com/de-de/azure/architecture/guide/architecture-styles/microservices*
(Gartner, 2023)	*Hype Cycle for Emerging Technologies, 2023*	*https://www.gartner.com/en/articles/what-s-new-in-the-2023-gartner-hype-cycle-for-emerging-technologies*

References

Reference	Title	Link
(Gartne-1, 2023)	*Hype Cycle for Software Engineering, 2023*	*https://www.gartner.com/en/documents/4590099*
(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*
(Mustaqeem, et al., 2023)	*In-Depth Analysis of Various Artificial Intelligence Techniques in Software Engineering: Experimental Study*	*https://jitm.ut.ac.ir/article_93632.html*