PRACTICAL IMPLEMENTATION OF SOFTWARE ENGINEERING TECHNIQUES IN AN INTERDISCIPLINARY CONTEXT

Presentation Download

Contact information

mail@marcuszinn.de

LinkedIn: Marcus Zinn

Dr. Marcus Zinn

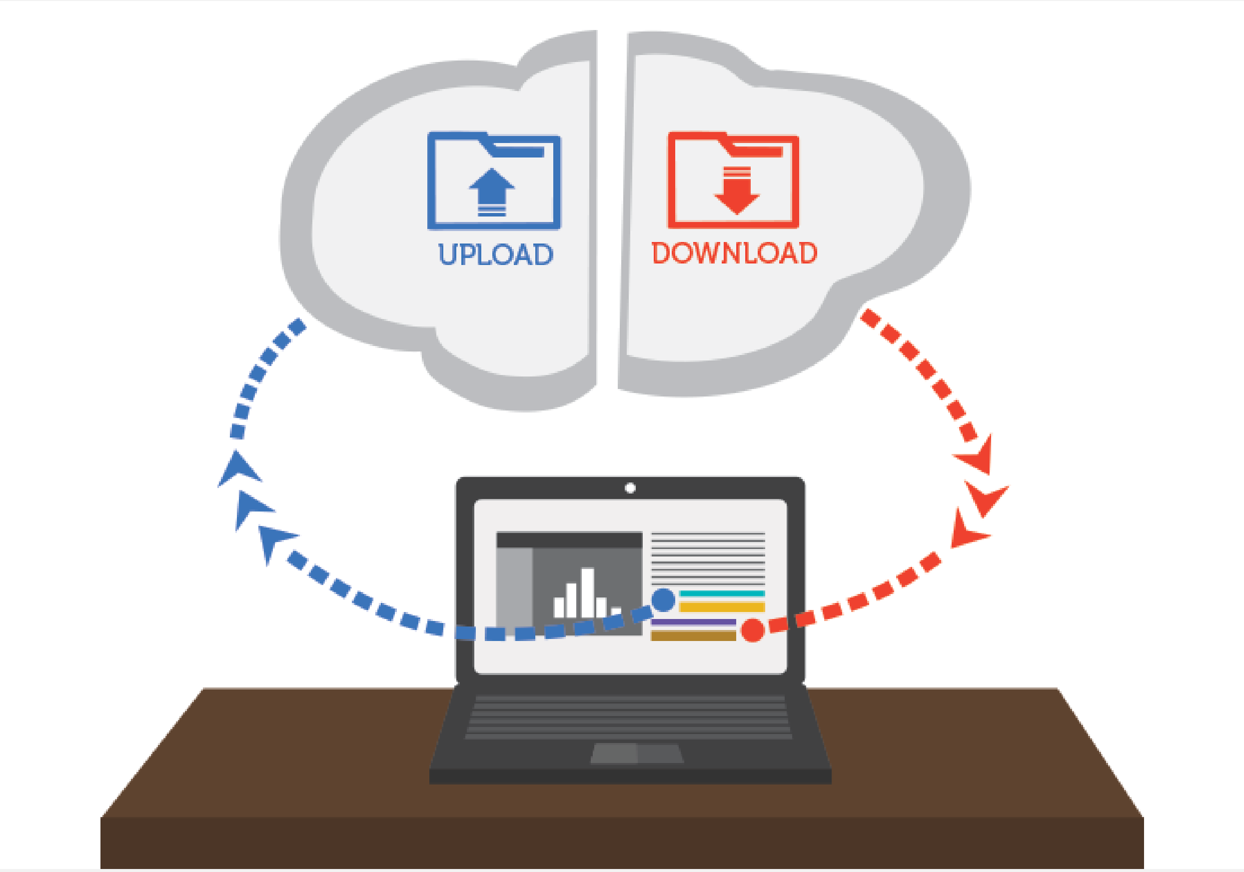
Version 19.09.2024 – University of Applied Sciences Darmstadt

OBJECTIVE OF THE PRESENTATION

- Definition and relevance of software engineering techniques and related interdisciplinary context
- Explanation of the practical application of these techniques in various disciplines
- Highlighting the challenges of the techniques in real interdisciplinary projects.
- Software engineering techniques and interdisciplinary context, summary, and conclusion.

LET'S START WITH A SIMPLE QUESTION

What means upload and download?



DEFINITION AND RELEVANCE OF SOFTWARE ENGINEERING TECHNIQUES

SURVEY

NAME AN EXAMPLE OF SOFTWARE ENGINEERING TECHNIQUES. WHAT IS THEIR BENEFIT?

Criterion	SE Techniques	SE Methods	SE Principles
Definition	Specific tools or procedures	Structural approaches for the entire process	Fundamental rules and best practices
Examples	TDD, CI, Design Patterns	Agile, Scrum, Waterfall Model	DRY, KISS, SOLID, YAGNI
Focus	Addressing specific tasks in the process	Organization and structure of the process	Philosophical guidelines for good software design
Area of application	Applied within methods	Governs the entire development process	General guidelines independent of method/technique

DIFFERENTIATION OF TERMS

EXAMPLES OF SOFTWARE ENGINEERING TECHNIQUES

- **Agile Development:**
  A flexible, iterative approach to software development where requirements and solutions are developed through collaboration in cross-functional teams.
- **Test-Driven Development (TDD):**
  A method where tests are written before the actual code to define requirements and ensure a robust implementation.
- **Continuous Integration (CI):**
  A process where developers regularly integrate their code into a central repository and run automated tests to detect errors early.
- **Continuous Deployment (CD):**
  An extension of CI, where, after successful tests, the software changes are automatically transferred to the production environment.
- **Design Patterns:**
  Reusable solutions to common problems in software architecture.
- **Microservices:**
  An architecture where applications are divided into small, independent services that can be developed, deployed, and scaled independently.

INTERDISCIPLINARY CONTEXT

WHAT IS AN INTERDISCIPLINARY CONTEXT?

- **Definition:** Interdisciplinary projects are those in which various disciplines collaborate to solve a problem.
- **Examples of disciplines:** Medicine, automotive industry, finance, environmental engineering, robotics, ar, automation area
- **Role of software development:** Software plays a central role in automation, analysis, and optimization in these areas.

Context	Who is involved	Disciplines Intersecting
Healthcare Software Project	Customer: Doctor Team Member: Data Scientist Academica: Healthcare Expert	Medicine, Data Science, Software Engineering
Automotive Industry Development	Customer: Car Manufacturer Team Member: Mechanical Engineer Academica: Robotics Specialist	Mechanical Engineering, Software Engineering, Robotics
Smart City Infrastructure	Customer: Urban Planner Team Member: Architect Academica: Environmental Scientist	Urban Planning, Architecture, Environmental Science, Software Engineering
Financial Tech (FinTech)	Customer: Financial Institution Team Member: Risk Manager Academica: Economist	Finance, Cybersecurity, Software Engineering
Manufacturing Optimization	Customer: Factory Manager Team Member: Industrial Engineer Academica: Operations Researcher	Industrial Engineering, Operations Research, Software Engineering
Media & Entertainment	Customer: Movie Studio Team Member: Graphic Designer Academica: Digital Arts Expert	Graphic Design, Film, Software Engineering
Cross-Disciplinary Research Project	Customer: Research Institution Team Member: Sociologist Academica: Data Scientist	Sociology, Data Science, Software Engineering

INTERDISCIPLINARY CONTEXT EXAMPLES

EXAMPLE
EUROPEAN FUNDING PROJECT BATTWIN

- **Key Focus**
  - Battery production; Digital Twin; Digital-Twin Platform

- **Context**
  - Europe lags behind Asia in Li-ion battery manufacturing (90% production in China, Korea, Japan).
  - 25 new Gigafactories planned in Europe by 2030 (€35 billion value).
  - Defect rates in initial production phases expected to be 15-30%.
  - Demand for €150 billion in battery manufacturing equipment.

- **Objective**
  - Develop a **Multilevel Digital Twin Platform** to support **Zero-Defect Manufacturing** in battery production, reducing defect rates in Gigafactories.

- **Key Components of BATTwin**
  - **Multi-sensor Data Acquisition & Management:**
    - Digital Battery Passport data model for improved data management.
  - **Process-level Digital Twins**
    - Model critical stages like electrode manufacturing, cell assembly, and conditioning.
  - **System-level Digital Twins**
    - Simulation and analytical modeling for comprehensive system insights.
  - **User-centric, Goal-driven Workflows**
    - Increased explainability and improved user control for design and system management.

Funded by
the European Union

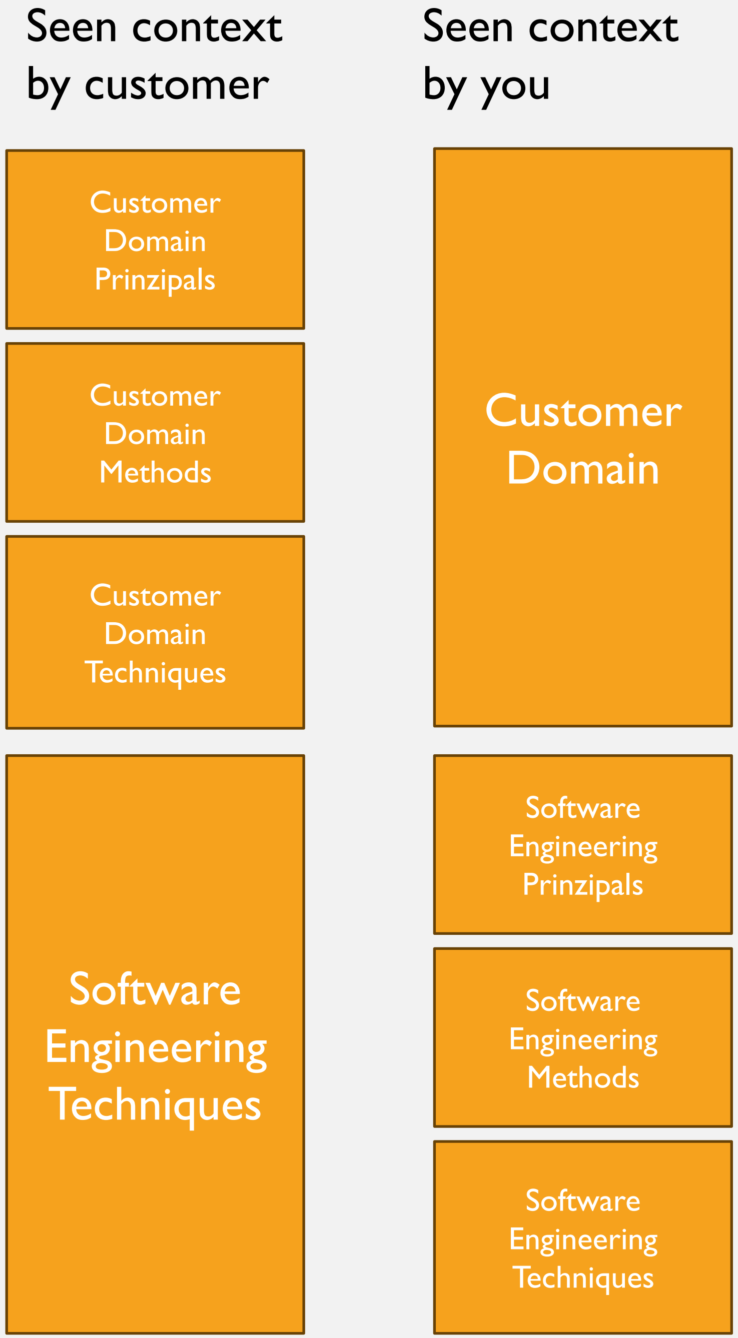
The BATTwin project has received funding from the European Climate, Infrastructure and Environment Executive Agency under grant agreement No. 101137954

Contributor	Role	Contribution
Politecnico di Milano	University	Leading in engineering and digital twin technologies
University of Oldenburg	University	Expertise in energy systems and environmental science
Verkor SA	Industrial Pilot	Testing battery production in industrial setting
COMAU SPA	Automation Solutions Provider	Robotics and automation expertise
The Royal Institute of Technology (KTH)	University	Research on digital twins and manufacturing systems
Upcell Alliance	European Battery Manufacturing Alliance	Industry network supporting battery manufacturing
Cambridge Nanomaterials Technology Ltd.	Nanotechnology Company	Nanomaterials research for battery innovations
HUN-REN SZTAKI	Research Institute	Expertise in AI and automation for digital twins
Sunlight Group Energy Storage Systems	Energy Storage Manufacturer	Specialist in energy storage systems and batteries
Ansys UK Ltd.	Simulation Software Company	Simulation and modeling software for optimization
Sivas University of Science and Technology	University	Research in manufacturing technologies
Syxis VSI	Software Solutions Provider	Development of software workflows for digital twins
N-ABLE	Business Strategy Consultancy	Consultancy on industrial strategy for scaling
Schneider Electric Automation GmbH	Automation and Energy Management	Energy management and automation solutions
Bureau Veritas Italia Spa	Certification and Compliance Company	Quality control and compliance in production
Communauté d'universités "HESAM UNIVERSITE"	University Consortium	Research in advanced manufacturing systems
Arts et Métiers Institute of Technology (ENSAM)	Institute of Technology	Focus on industrial engineering and manufacturing
CESSI	Research Institute	Research in systems automation
CESI School of Engineering	Engineering School	Research and innovation in energy systems
Conservatoire National des Arts et Métiers	Educational Institution	Expertise in industrial systems and lifelong learning

BATTWIN PARTICIPANTS (INTERDIZIPLINARY)

PROBLEMS OF INTERDISCIPLINARY COMMUNICATION

I.E. COMMUNICATION OF TERMS



Aspect	Description	Challenge	Solution
Communication of terms	Terms have different meanings and emotional/political connotations across disciplines.	Misunderstandings about key terms can hinder collaboration. Uncertainty about which terms are important makes discussion difficult.	Create a glossary and clarify the meaning of key terms as the project progresses.
Diversity of narratives	Each discipline has its own forms of presentation and publication methods.	Different writing styles (e.g., scientific: results-focused; humanities: narrative and moral conclusions) make understanding difficult.	Develop an understanding of the different narrative styles and adjust the structure to meet the expectations of the other disciplines.
Presentation of results	Results are presented in different forms, e.g., text-oriented vs. graphical.	Graphical representations can lose nuances and limit interpretation, while text-based representations may overwhelm with details.	Reflect on the form of presentation and regularly discuss it within the team to minimize misunderstandings.
Assumptions about data quality	Different disciplines have different standards for evaluating data.	The quality of data is assessed differently by each discipline, which can lead to distrust or misunderstanding.	Build trust, establish clear communication rules, and hold regular meetings to discuss data quality.
Interdisciplinary research takes time	Interdisciplinary projects require more time to establish communication processes and develop new questions.	Time pressure and different priorities make it difficult to establish productive collaboration.	Plan time for communication and establish a person as an "interdisciplinary hub" responsible for moderating and documenting collaboration.
Role of consulting	External consulting can help identify and address differences in scientific theory.	Misunderstandings due to different theoretical approaches, e.g., constructivism vs. positivism, often turn discussions into debates over terminology.	Use external consulting to identify and resolve conflicts at the theoretical level early on.
Language and understanding	Each discipline develops its own language.	Technical jargon and different terminologies hinder understanding.	Develop a communication style based on everyday language and practical examples to facilitate understanding. Constitute a common language through practice.

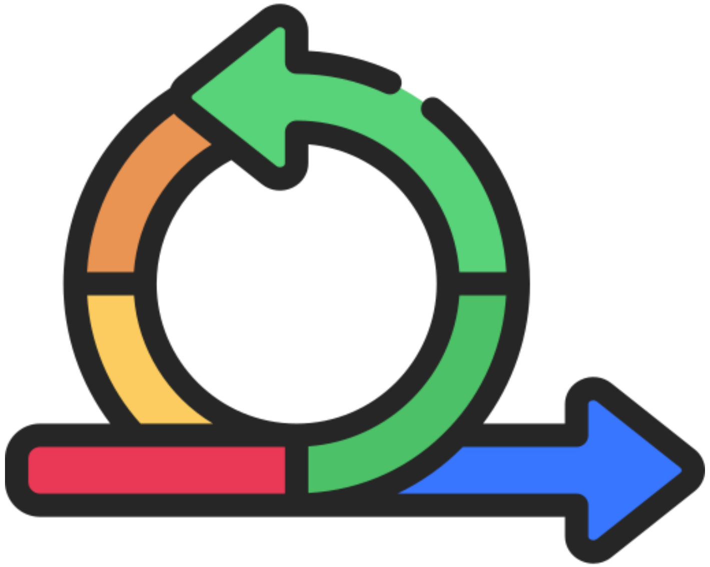
PROBLEMS OF INTERDISCIPLINARY COMMUNICATION

EXPLORING THE PROFOUND EFFECTS OF SOFTWARE ENGINEERING ON DIFFERENT INDUSTRIES

EXPLORING THE PROFOUND EFFECTS OF SOFTWARE ENGINEERING ON DIFFERENT INDUSTRIES

Industry	Key Areas	Description	Benefits/Impact
Healthcare	Electronic Health Records (EHR)	Software solutions for digital management and storage of patient data.	Enhanced data accessibility, reduced errors, and improved overall efficiency.
	Health Monitoring	IoT devices and software-driven wearables for real-time monitoring of vital signs.	Early detection of health issues through continuous monitoring.
Finance	Online Banking	Digital banking platforms enabling secure transactions, fund transfers, and account access.	Faster and safer transactions, increased convenience for customers, and improved accessibility.
	Algorithmic Trading	Software algorithms that enable high-frequency trading, executing trades within microseconds.	Improved trading accuracy, reduced costs, and increased trading volumes.
Manufacturing	Automation	Software-controlled robots and machines improving operational efficiency.	Reduced human error and increased productivity in manufacturing processes.
	Supply Chain Management (SCM)	Sophisticated software optimizing supply chain processes and logistics.	Enhanced inventory management, better demand forecasting, and reduced logistics costs.
Entertainment	Streaming Services	Platforms like Netflix and Spotify delivering digital content directly to consumers.	Revolutionized how content is consumed, providing seamless access to music, movies, and TV shows.
	Video Games	Advanced software techniques for realistic graphics and immersive multiplayer experiences.	Enhanced user engagement through immersive and interactive gaming experiences.
	Virtual Reality (VR) / Augmented Reality (AR)	Software innovations enabling immersive VR/AR experiences.	Transformed user interactions with digital content, creating new forms of entertainment.

CASE STUDIES OF SOFTWARE ENGINEERING TECHNIQUES IN INTERDISCIPLINARY CONTEXTS.



AGILE DEVELOPMENT – PRINCIPLES & APPLICATION.

- **Definition**
  - Agile development is an iterative approach in which software is developed in small, incremental steps.
- **Characteristics**
  - Iterative cycles (Sprints)
  - Continuous improvement
  - Close collaboration with customers/users
- **Advantages:**
  - Flexibility and quick adaptation to new requirements
  - Fast feedback cycles
- **Graphic**
  - Illustration of a typical Scrum cycle (Product Backlog, Sprint Planning, Daily Standups, Review).

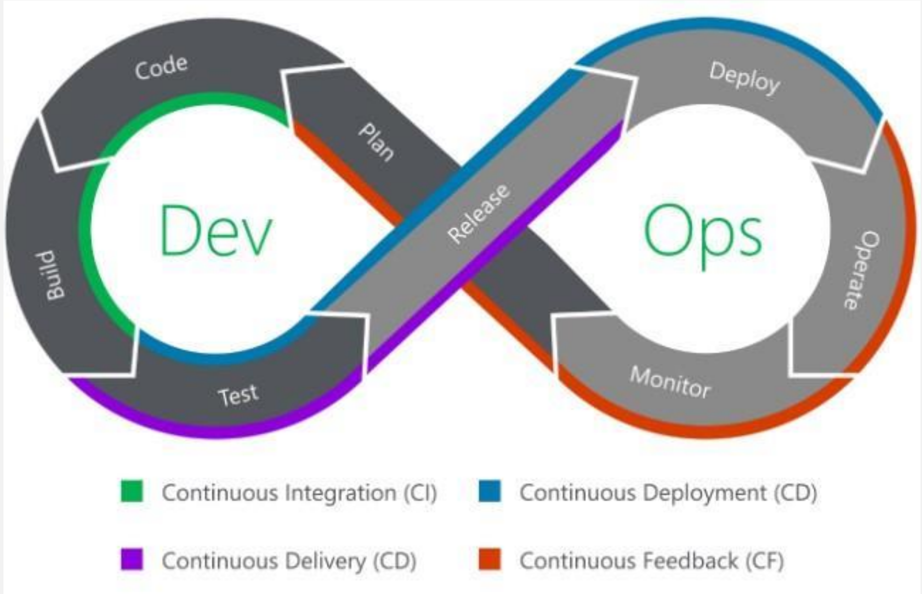
EXAMPLE 1 – AGILE DEVELOPMENT IN THE AUTOMOTIVE INDUSTRY

- **Agile Methods Revolutionize Car Manufacturing**
  - A German sports car manufacturer needed to integrate a new regulation into ongoing production.
  - The regulation required additional inspection of every connection in the vehicle.
  - This was highly complex due to the precisely timed workflows and limited space on the production line.
- **Problem**
  - Car manufacturing processes are timed to the second.
  - Introducing new inspection requirements without stopping or disrupting production was challenging.
  - The situation was complicated by the fact that multiple car models were produced on the same assembly line.
- **Agile Solution**
  - The integration was approached as an agile project to handle the complexity.
  - An interdisciplinary team worked in **sprints** to gradually implement the new requirements.
  - Collaboration with **CO-Improve** (Scrum coach) enabled dynamic adjustments to the process.
  - A **product owner** from assembly planning managed the agile backlog, continuously defining and prioritizing requirements.

EXAMPLE 1 – AGILE DEVELOPMENT IN THE AUTOMOTIVE INDUSTRY

- **Detailed Agile Implementation**
  - **Sprints and Pilot Project**: Initial tests were carried out on a part of the production line in small, iterative steps. After just six weeks, a Minimum Viable Product (MVP) was implemented, enabling the inspection of 25 connections.
  - **Feedback and Optimization**: Over a 14-day test phase, the team regularly gathered feedback from assembly workers and quality managers. Processes were iteratively improved based on this feedback.
  - **Capacity and Challenges**: Initially, team members could only devote a small portion of their time to the project. A burn-up chart highlighted the need for more capacity, which ultimately led to significant efficiency improvements.
- **Result**
  - After eight months, the agile integration of the new inspection guidelines was successfully completed. The roll-out to the entire production line was seamless, enabling flexible adjustments and rapid implementation without interrupting ongoing production.

CONTINUOUS INTEGRATION (CI) & CONTINUOUS DEPLOYMENT (CD)& CONTINOUS DELIVERY & DEVOPS



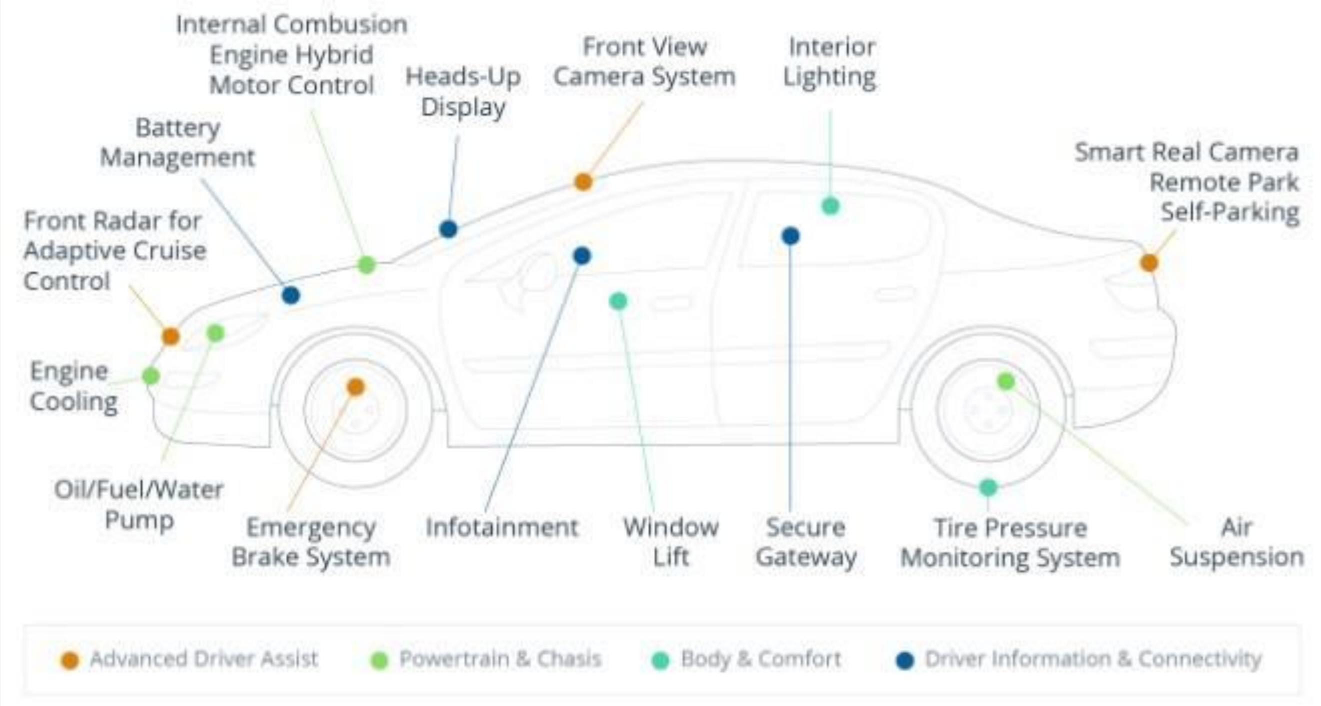
- **Continuous Integration (CI)**
  - Developers integrate their work several times a day into a central repository.
  - Automated tests are executed to catch errors early, integrating DevOps practices for quicker feedback.

- **Continuous Deployment (CD)**
  - Automated deployment of code to the production environment after it has been tested, with a final human review.
  - DevOps streamlines the testing and deployment phases to ensure smoother releases.

- **Continuous Delivery**
  - Automated deployment of code to the production environment after it has been tested, without a final human review.
  - DevOps practices minimize human intervention, promoting faster, automated delivery.

- **Advantage**
  - DevOps enables fast and frequent updates while minimizing errors through continuous integration, testing, and deployment.

IDENTIFY INTERDISZIPLINARY CONTEXT IN AN ELECTRICAL VEHICLE

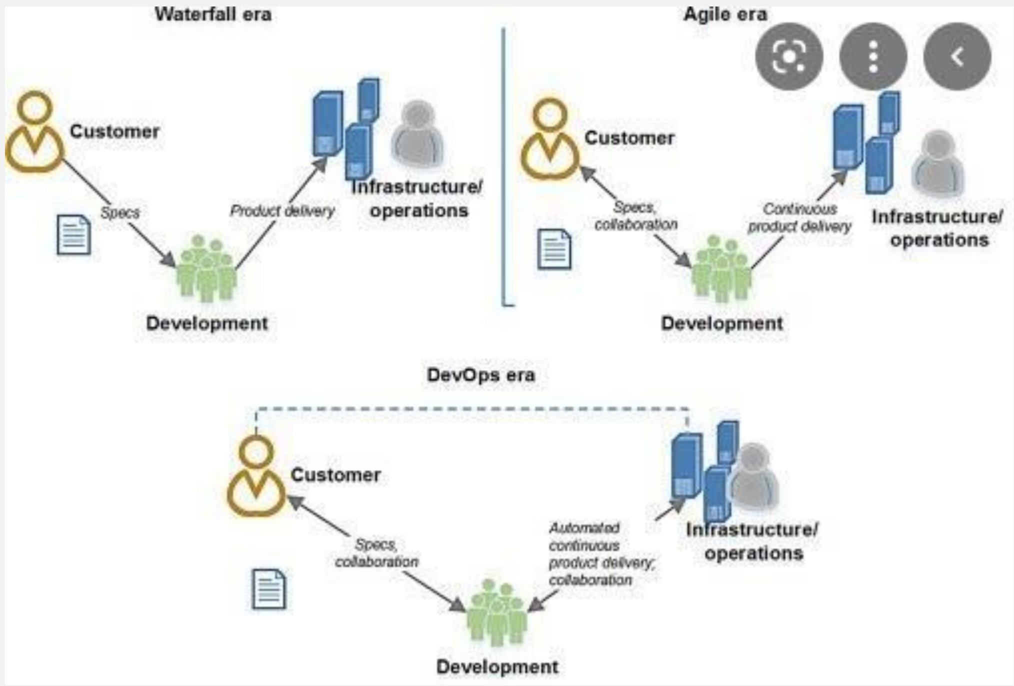


ELECTRICAL VEHICLE (EV) IN AUTOMOTIVE AREA

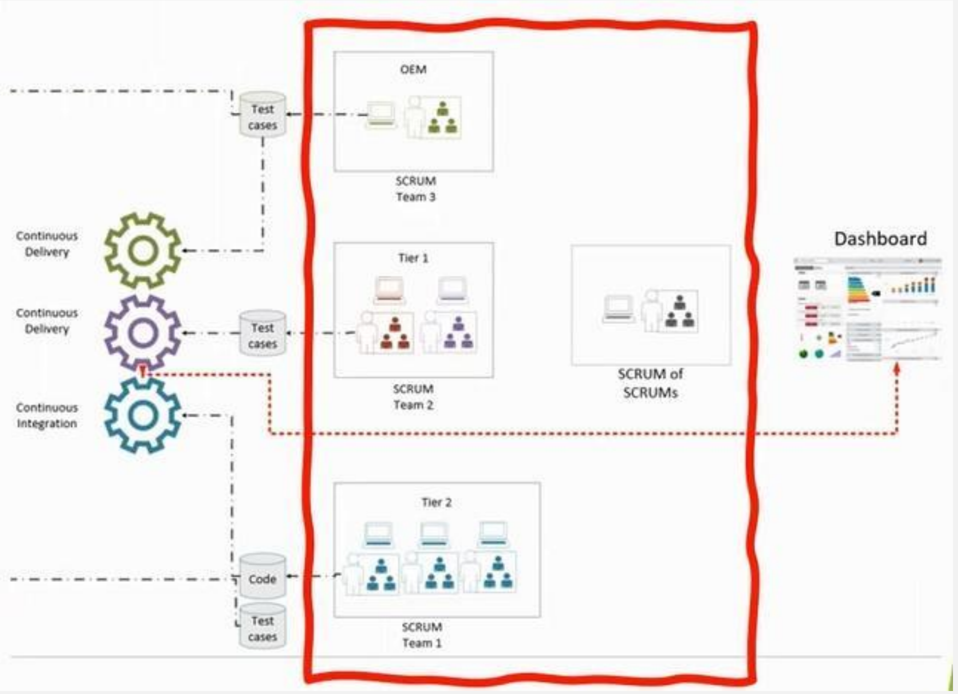
- **Data Misuse, Privacy, and Security Threat**
    - Software-driven vehicles collect large amounts of data: driving speed, routes, search history, music playlists, location, and even credit card information.
    - Cyber attacks can compromise this data, leading to privacy concerns and consumer mistrust.
    - i.e., 25 Tesla cars were remotely hacked after exploiting the security vulnerability of TeslaMate, a famous Tesla data logging tool. Therefore, electric vehicle manufacturing needs to be more vigilant towards enhancing software security.

- **Complex Infrastructure and Design**
    - Modern EVs rely on sensors and embedded devices to sense their surroundings.
    - EV manufacturers require complex testing environments to ensure performance and reliability.
    - Managing these complex infrastructures increases time and cost of product delivery.
    - Inadequate infrastructure for frequent software updates can introduce bugs and errors.

DEVOPS AS SOLUTION



Evolution of Automotive SW Development



DevOps Schematic for Automotive SW Development

ELECTRICAL VEHICLE (EV) IN AUTOMOTIVE AREA

- **Data Misuse, Privacy, and Security Threat**
  - **DevSecOps integrates security into DevOps by continuously monitoring and detecting potential threats.**

- **Complex Infrastructure and Design**
  - **Cloud technology integrated with DevOps helps build, maintain, and test applications efficiently.**

ELECTRICAL VEHICLE (EV) IN AUTOMOTIVE AREA

- **Continuous Deployment**
  - EV software requirements evolve during and after product release.
  - New regulations, like road safety laws, must be integrated into existing systems quickly.
  - DevOps supports agile development, allowing EV companies to update systems with minimal downtime.
  - Enables faster tracking and implementation of changes at every development phase.

ELECTRICAL VEHICLE EXAMPLE DEVOPS

- **Tesla:**
  - Utilizes DevOps to deliver microservices using Docker containers and Kubernetes orchestration.
  - Implements Over the Air (OTA) wireless software updates using DevOps.
  - Enables quick bug fixes and frequent delivery of new features through DevOps practices.

- **Jaguar (Land Rover):**
  - Adopted DevOps to improve throughput of the infotainment system.
  - Built a server with continuous integration and testing for software integration.
  - Automated build and deployment process using Docker containers.
  - Reduced build-to-delivery time from 6 weeks to 1 hour.
  - Delivered the same infotainment system for nine different vehicles.

- **Mercedes (Connected Cars Platform team):**
  - Created an automated and secure DevOps pipeline based on containers and Kubernetes.
  - Implemented an automated CI/CD pipeline governing development, testing, and deployment.
  - Reduced software release time from months to weeks.

CHALLENGES IN USING SOFTWARE ENGINEERING TECHNIQUES IN AN INTERDISCIPLINARY CONTEXT

Topic	Problem	Example	Solution
(A) Complexity of Integration	The integration of CI/CD techniques in hardware-intensive environments (e.g., automotive or robotics projects) is complex.	In autonomous vehicles or production facilities, software updates must be seamlessly integrated into existing systems without endangering the machines.	Use simulation environments and specialized testing equipment to test the impact of software changes on hardware.
(B) Legacy Systems and Old Systems	Old software systems (legacy systems) are difficult to adapt to new agile or CI/CD processes.	In financial technology or the automotive industry, there is decades-old software, the replacement or integration of which into new structures is risky.	Gradual modernization and use of microservices to ensure compatibility with older systems.

TECHNICAL CHALLENGES IN THE APPLICATION OF SOFTWARE ENGINEERING TECHNIQUES IN AN INTERDISCIPLINARY CONTEXT

Topic	Problem	Example	Solution
(A) Communication Problems Between Disciplines	Different disciplines (e.g., software developers and mechanical engineers) often speak different "languages," which complicates communication and understanding.	Software developers think in abstract algorithms and data structures, while engineers solve practical, mechanical problems. This leads to misunderstandings when discussing requirements and solutions.	Use common, understandable documentation standards and hold regular alignment meetings.
(B) Different Priorities and Work Styles	Engineers and software developers often have different approaches and priorities. Engineers work with fixed plans, while software developers are more flexible.	A mechanical engineer needs precise, predictable plans for construction, while a software developer can respond more flexibly to changes.	Agile methods help bridge this gap by creating small, testable milestones that both sides can understand and track.

CHALLENGES IN IMPLEMENTING SOFTWARE ENGINEERING TECHNIQUES IN AN INTERDISCIPLINARY CONTEXT

Topic	Problem	Example	Solution
(A) Scaling Agile Methods in Large Teams	Agile methods work well in small teams, but scaling in large interdisciplinary projects (e.g., in the automotive or automation industry) can be challenging.	A large project that involves software and hardware engineers, designers, and product specialists must be well-coordinated. Too many interfaces complicate collaboration.	Scaled Agile Framework (SAFe) or LeSS (Large Scale Scrum) provide opportunities to scale agile methods to large teams by creating clear structures and processes.
(B) Technical Debt from Fast Iterations	In fast, iterative development cycles, technical debt can accumulate because short-term solutions are implemented to progress faster. This debt must be "repaid" later through extensive refactoring.	In an autonomous vehicle, a provisional algorithm for sensor processing is introduced, which must be optimized in later versions.	Plan regular refactoring sprints to reduce technical debt and maintain long-term code sustainability

CHALLENGES IN SCALING OF SOFTWARE ENGINEERING TECHNIQUES IN AN INTERDISCIPLINARY CONTEXT

SECURITY AND COMPLIANCE CHALLENGES

Topic	Problem	Example	Solution
Safety Requirements in Critical Sectors	In sectors such as medical technology or the automotive industry, software changes must adhere to strict safety and compliance regulations.	In autonomous vehicle software, every code change must be thoroughly assessed for its impact on vehicle safety. Faulty software can lead to serious accidents.	Introduce Safety-Critical Development Guidelines along with rigorous testing and validation processes (e.g., ISO 26262 for the automotive industry).
Data Privacy and Compliance	When developing software that processes personal data, strict data protection regulations such as GDPR must be followed.	Software for hospitals that manages patient data must ensure all data is encrypted and stored in a privacy-compliant manner.	Implement Privacy by Design: Data protection requirements should be integrated from the beginning of the development process to avoid future issues.
Interdisciplinary Collaboration for Compliance	Collaboration across multiple disciplines (e.g., software engineers, legal teams, and domain experts) is necessary to ensure compliance with various regulatory frameworks.	In a project that involves both medical devices and healthcare data processing, compliance needs to meet both healthcare data regulations and device safety standards.	Foster cross-disciplinary teams to integrate compliance workflows, ensuring regulatory requirements are considered throughout software development lifecycles.
Security in Embedded Systems	Embedded systems in sectors such as aerospace or automotive must meet strict real-time safety and security requirements.	The software managing aircraft control systems must respond to threats or malfunctions in real-time without human intervention.	Employ real-time security protocols and testing, using standards such as DO-178C (avionics software) to ensure both safety and compliance.

REVOLUTIONIZING INDUSTRIES THROUGH ADVANCED SOFTWARE ENGINEERING TECHNIQUES

REVOLUTIONIZING INDUSTRIES THROUGH ADVANCED SOFTWARE ENGINEERING TECHNIQUES

Key Techniques	Description	Benefits	Industries Transformed
Artificial Intelligence (AI)	AI mimics human intelligence to perform complex tasks autonomously, generating valuable insights through data analysis and machine learning algorithms.	• Automates repetitive tasks, reduces human error. • Enhances decision-making with data-driven insights. • Increases operational efficiency.	• Healthcare • Finance • Manufacturing
Big Data Analytics	Big Data enables businesses to manage and analyze massive datasets, uncovering hidden patterns and trends to drive informed decision-making and optimize strategies.	• Real-time analytics for faster decisions. • Improves customer understanding and personalized experiences. • Optimizes operations	• E-commerce • Marketing • Healthcare
Agile Development	Agile enables iterative and flexible software development, emphasizing collaboration, adaptability, and rapid delivery of solutions.	• Frequent customer feedback improves solutions. • Minimizes risks and adapts to challenges. • Faster time to market and continuous innovation.	• Software development • IT services • Project management
DevOps	DevOps integrates development and operations teams, fostering collaboration and automating the software delivery pipeline for faster, more reliable updates.	• Enables continuous integration and deployment. • Reduces downtime with automated testing and monitoring. • Improves software quality and delivery performance.	• Software development • Online services

SOFTWARE ENGINEERING TECHNIQUES IN AN INTERDISCIPLINARY CONTEXT SUMMARY & CONCLUSION

Existing Challenges

1. Varied interpretations and understandings across disciplines can hinder collaboration.
2. Misaligned goals and communication gaps are common when teams with different expertise work together

Positive Implementations

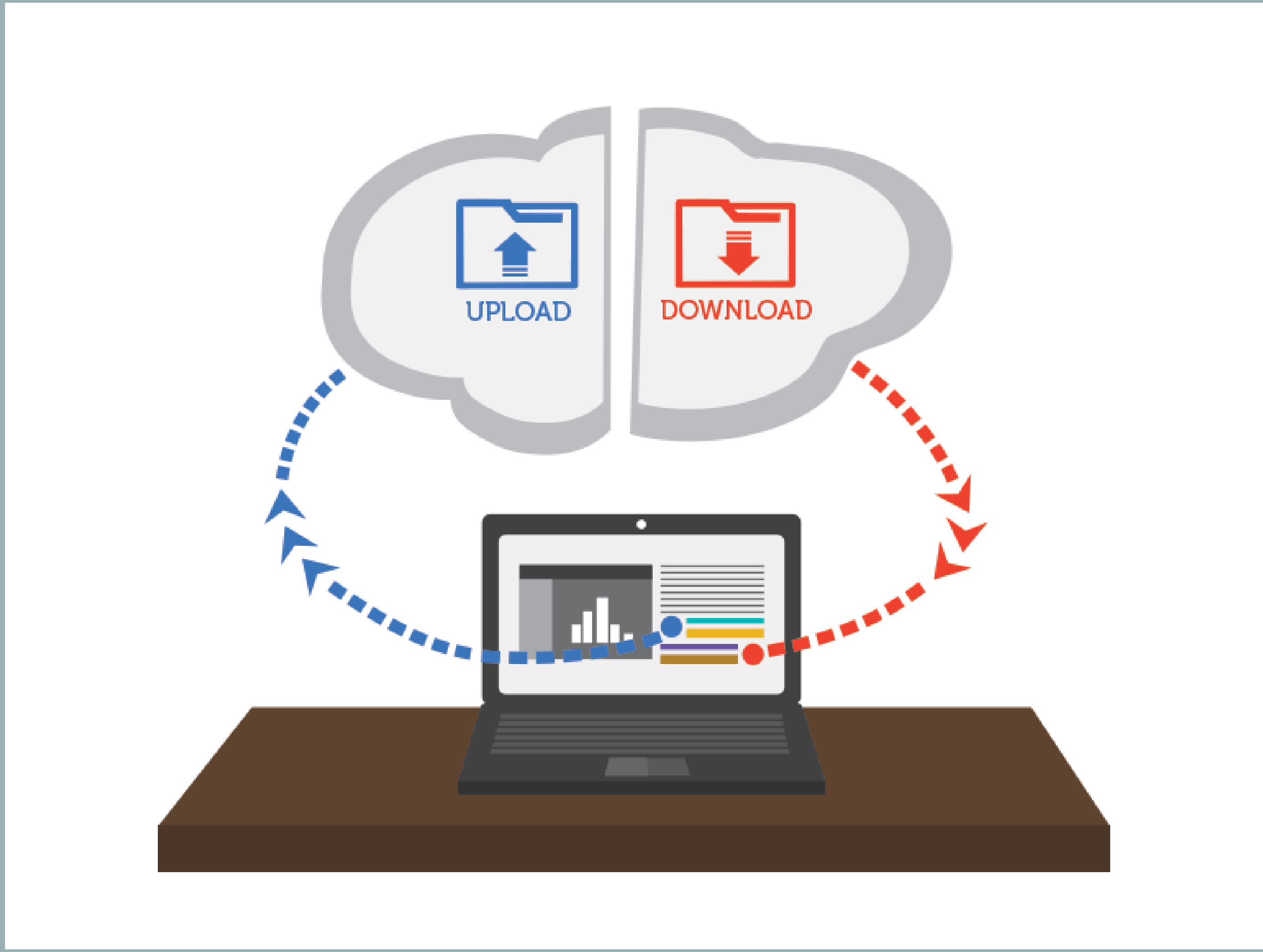
1. Agile and DevOps have been successfully adopted in automation projects, offering flexibility and continuous integration.
2. These methodologies help bridge gaps between software development and other fields like engineering, research, or management

Challenges of Applying Software Engineering Techniques

1. Applying software engineering techniques in an interdisciplinary context requires overcoming differences in terminology, workflow expectations, and priorities between teams.
2. Managing these differences is essential for success.

LET'S END WITH A SIMPLE QUESTION

What means upload and download?



QUESTIONS?

PRACTICAL IMPLEMENTATION OF SOFTWARE ENGINEERING TECHNIQUES IN AN INTERDISCIPLINARY CONTEXT

Presentation Download

Contact information

mail@marcuszinn.de

LinkedIn: Marcus Zinn