Die Modulbeschreibung sollte direkt über diesen Link in HISinOne eingepflegt werden.

Module code	**Module title**		**Category**
MAIE1050	Software Quality Engineering & DevOps		MA
	Degree program	MA Software Engineering	
	Faculty	Building Services Engineering and Computer Science	

Module coordinator	Prof. Dr. Volker Herwig
Module type	Mandatory module
Frequency	1x annually in SuSe
Recommended semester	1. semester
Credit (ECTS-Points)	5
Academic Assessment Method	Exam PZ = Examination requirement (N: graded) PZ (N), K90)
Teaching language	English
Admission requirements for this Module	none
Module duration	1 Semester
Required Registration	Students enrolled in the above-mentioned degree program/standard semester will be registered automatically upon re-enrollment; all other participants, please refer to the information below. none

	Course	**Lecturer**	**Type**	**Group Size (max.)**	**Number of Groups**	**Contact hours per week (SWS)**	**Workload (in h)**	
							Face-to-face	**Self-study**
1	Software Quality Engineering & DevOps	Extern	Seminar	30	1	4	60	65
2	Titel der Lehrveranstaltung.	Dozent*in	Wählen Sie ein Element aus.		Wählen Sie ein Element aus.			
3	Titel der Lehrveranstaltung.	Dozent*in	Wählen Sie ein Element aus.		Wählen Sie ein Element aus.			
4			Wählen Sie ein Element aus.					
5	Titel der Lehrveranstaltung.	Dozent*in	Wählen Sie ein		Wählen Sie ein			

			Element aus.		Element aus.			
						Sum	**4,0**	**60**	**65**
						Total Workload for Module	**125**	

Learning Objectives / Learning outcomes	1. Explain and contrast **Quality Assurance (QA)**, **Quality Engineering (QE)**, **Development–Operations (DevOps)**, and **Site Reliability Engineering (SRE)** practices, and justify *shift-left* and *shift-right* quality approaches across the **Software Development Life Cycle (SDLC)**. 2. Design a system-specific quality strategy grounded in recognized quality models (e.g., **International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) 25010**), deriving measurable, testable quality attributes. 3. Define and implement a test strategy (test pyramid) with automated unit, component/contract, integration, and end-to-end tests, including coverage and mutation-testing targets. 4. Build a **Continuous Integration / Continuous Delivery (CI/CD)** pipeline with quality gates (linting, static analysis, dependency scans and a **Software Bill of Materials (SBOM)**), artifact versioning, environment promotion, and safe release patterns (blue-green, canary, rollback). 5. Apply **Infrastructure as Code (IaC)** to provision reproducible, ephemeral test/staging environments (e.g., containerized and orchestrated) and embed them into the delivery pipeline. 6. Instrument applications for observability (logs, metrics, traces), define **Service Level Indicators (SLIs)** and **Service Level Objectives (SLOs)** with error budgets, and configure alerting for actionable incident response. 7. Plan and execute non-functional testing (performance, load/stress, reliability/chaos experiments) and analyze results to tune architecture and runtime configurations. 8. Integrate **Development, Security, and Operations (DevSecOps)** practices—threat modeling, **Static Application Security Testing (SAST)**, **Dynamic Application Security Testing (DAST)**, **Software Composition Analysis (SCA)**, secrets management, and supply-chain security—and evaluate residual risk and compliance impacts. 9. Use risk-based testing and applicable compliance requirements to prioritize quality activities and justify trade-offs among cost, time, scope, and quality. 10. Apply modern version-control and branching strategies (e.g., trunk-based, short-lived branches) with robust code review and traceability to support continuous delivery. 11. Define, track, and interpret delivery and quality metrics (e.g., lead time, deployment frequency, change-failure rate, **Mean Time To Restore (MTTR)**, defect trends) to drive continuous improvement. 12. Document and communicate the quality architecture, pipeline design, and assurance evidence, and defend decisions to technical and non-technical stakeholders.
Contents	• The module integrates Quality Engineering with Development–Operations (DevOps) to ensure continuous, measurable software quality from code to production. Students design a quality strategy, implement automated tests, and build Continuous Integration / Continuous Delivery (CI/CD) pipelines with quality gates. They instrument services for observability and apply

	Development, Security, and Operations (DevSecOps) practices, using delivery and reliability metrics for continuous improvement. Core topics: • Quality models (e.g., ISO/IEC 25010), quality attributes, acceptance criteria • Risk-based test strategy; test pyramid (unit, component/contract, integration, end-to-end) • Code quality: linting, static analysis, dependency management, Software Bill of Materials (SBOM) • CI/CD pipelines: stages, quality gates, artifact versioning, promotion, canary/blue-green/rollback • Environments & Infrastructure as Code (IaC); containers/orchestration; secrets handling • Observability: logs, metrics, traces; Service Level Indicators (SLIs)/Service Level Objectives (SLOs), alerting, runbooks • Non-functional testing: performance, reliability/chaos; tuning based on findings • DevSecOps: threat modeling, Static/Dynamic Application Security Testing (SAST/DAST), Software Composition Analysis (SCA), supply-chain security
Literature	• Humble, J., & Farley, D. (2010). *Continuous Delivery*. Addison-Wesley. • Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The Science of DevOps*. IT Revolution. • Kim, G., Humble, J., Debois, P., & Willis, J. (2021). *The DevOps Handbook* (2nd ed.). IT Revolution. • Meszaros, G. (2007). *xUnit Test Patterns*. Addison-Wesley. • Beck, K. (2002). *Test-Driven Development: By Example*. Addison-Wesley. • Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (Eds.). (2016). *Site Reliability Engineering*. O’Reilly. • Hidalgo, A. (2022). *Implementing Service Level Objectives*. O’Reilly. • Majors, C., Fong-Jones, L., & Miranda, G. (2022). *Observability Engineering*. O’Reilly. • Vehent, J. (2018). *Securing DevOps*. Manning. • ISO/IEC. (2011). *ISO/IEC 25010:2011 Systems and software quality models*. ISO. •