Hochschule für Technik Stuttgart
Fakultät Vermessung, Informatik und Mathematik
Schellingstraße 24
D-70174 Stuttgart
T +49 (0)711 8926 2626
F +49 (0)711 8926 2553
www.hft-stuttgart.de
sekretariat.vim@hft-stuttgart.de

Hochschule für Technik Stuttgart

Master Software Technology

Syllabus

Status September 2023

Syllabus

This document contains the current module handbook for the Master Software Technology. The order of the modules corresponds to the study and examination regulations. For the compulsory elective modules (6 and 11), the modules available for selection are described in the respective subsections.

In total, the program comprises the following modules:

Explanation of the following module descriptions

The modules listed under prerequisites are of a content-related nature, not of a formal nature. Deviations from this are explicitly mentioned in the individual modules. The admission requirements for the Master's degree program in Software Technology are as follows

- Programming (knowledge of at least one object-oriented programming language)
- Software Engineering
- Database Systems
- Data Structures and Algorithms
- Distributed Systems

These content requirements are ensured (see §5 of the admission and selection regulations).

1 Concepts of Programming Languages

Course	Master Software Technology
Name of Module	Concepts of Programming Languages
Abbreviation	CPL
Semester	1
Responsible	Prof. Dr. Heusch
Lecturers	Prof. Dr. Heusch, Prof. Dr. Padó
Elective / compulsory	Compulsory module
Frequency	Every year
Weekly Contact Hours	4h
Method of Teaching	Lectures with Exercises, Reading Assignments, Assignments
Student Work Load – Lectures	68h
Student Work Load – Self Studies	82 h
European Credit Transfer Points	5 ECTS Points
Necessary Previous Knowledge	Comprehensive knowledge of Java
Final Knowledge and Skills	*Knowledge and understanding* On completion the student knows the different language paradigms and concepts, especially in ■ Procedural Programming ■ Functional Programming ■ Object Oriented Programming ■ Logic Programming *Disciplinary / professional skills* On completion the student is able to select a fitting paradigm and programming language for a given problem and to rate the implications of this decision.
Index	■ The early days: FORTRAN, COBOL, PL/1 ■ Recursion, Functions and Lambda Calculus: Lisp, Scheme, Forth ■ The Algol-Languages: Algol, Pascal, PL/SQL, C ■ Object-oriented Programming: Simula, Smalltalk, C++ ■ Weakly typed languages: Perl, Python, JavaScript, Ruby ■ Programming in Logic: PROLOG

	■ Special Purpose Programming Languages: ABAP, APL, Flash
Pre-Exam Requirements	none
Method and Extent of Examination	Project Work (graded)
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ Abelson, H., J. Sussmann: Structure and Interpretations of Programming Languages. MIT-Press/ McGraw-Hill, 1996. ■ Clocksin, W., C. Mellish: Programming in Prolog, Springer, 2003. ■ Mitchell, J.: Concepts in Programming Languages. Cambridge University Press, 2001. ■ Sebesta, R.: Concepts of Programming Languages. Addison-Wesley 2003.

2 Database Systems II

Course	Master Software Technology
Name of Module	Database Systems II (Advanced Topics in Databases)
Abbreviation	DAB
Semester	1
Responsible	Prof. Koch
Lecturers	Prof. Koch, Prof. Dr. Kramer
Elective / compulsory	Compulsory module
Frequency	Every year
Weekly Contact Hours	4h (3h Lectures + 1h Exercises/Presentations)
Method of Teaching	Lecture with theoretical and practical exercises; independent group project with presentation
Student Work Load – Lectures	68h
Student Work Load – Self Studies	112h
European Credit Transfer Points	6 ECTS Points
Necessary Previous Knowledge	Data structures/algorithms; Bachelor level understanding of file systems, computer architecture, and databases; Entity Relationship Modeling; basic knowledge of the relational model and SQL

Final Knowledge and Skills	*Knowledge and understanding* On completion the student has a deeper understanding of DBMS functionality and in particular of modern system approaches. He or she has practical experience with at least one relational database system and insight into current database research issues. *Disciplinary / professional skills* On completion the student is able to evaluate strengths and weaknesses of database and transaction processing systems and to make informed decisions about different situations of database usage in practical projects within enterprise contexts.
Index	■ Review of principles of relational databases, advanced features of SQL, the MySQL DBMS ■ Database programming (ODBC, SQL/CLI, JDBC, Embedded SQL, Dynamic SQL, SQLJ) ■ Transaction management: review of basic properties, distributed and nested transactions, sagas, 2 phase and 3 phase commit protocol, long transactions, architecture and functionality of transaction processing systems ■ Recovery: logging, checkpointing, savepointing, recovery after software and hardware failures, backup methods ■ Concurrency control: 2 phase locking, isolation levels, timestamp and optimistic protocols ■ Distributed databases: data fragmentation, replication, and allocation techniques; distributed recovery and concurrency control ■ Mobile databases: architecture, data replication, transaction processing, performance ■ Object-oriented and object-relational databases, comparison to relational systems
Pre-Exam Requirements	Successful group project
Method and Extent of Examination	Written examination, 120 minutes
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ Bernstein, P., E. Newcomer: Principles of Transaction Processing for the System Professional. Morgan Kaufmann, 1997. ■ Cattell, R.G.G.: Object Data Management, Addison-Wesley, 1994. ■ Ceri, S., G. Pelagatti: Distributed Databases, Principles and Systems. McGraw-Hill, 1984. ■ Connolly, T.M., C.E. Begg, A.D. Strachan: Database Systems, A Practical Approach to Design, Implementation and Management. Addison-Wesley, 2001. ■ Date, C.J.: An Introduction to Database Systems. Addison Wesley, 1999. ■ Elmasri, R., S. Navathe: Fundamentals of Database Systems. Addison Wesley 2004. ■ Gray, J., A. Reuter: Transaction Processing, Concepts and Techniques. Morgan Kaufmann, 1993.

	■ Ozsu, M.T., P. Valduriez: Principles of Distributed Database Systems. Prentice Hall, 1999. ■ Stonebraker, S., D. Moore, P. Brown: Object-Relational DBMSs. Morgan Kaufmann, 1998. ■ Various international research papers (distributed in class)

3 Software Project Management II

Course	Master Software Technology
Name of Module	Software Project Management II (Advanced Topics in Software Project Management)
Abbreviation	SPM
Semester	1
Responsible	Prof. Dr. Kramer
Lecturers	Prof. Dr. Deininger, Prof. Dr. Höß, Prof. Dr. Kramer
Elective / compulsory	Compulsory module
Frequency	Every year
Weekly Contact Hours	4h (3h Lectures + 1h Exercises/Presentations)
Method of Teaching	Lecture with Exercises, Reading Assignments, Students' presentations, applying project management software
Student Work Load – Lectures	68h
Student Work Load – Self Studies	112h
European Credit Transfer Points	6 ECTS Points
Necessary Previous Knowledge	■ Software Project Management (Bachelor Level) ■ Experience in (small) software projects, either at the university or in industry
Final Knowledge and Skills	*Knowledge and understanding* On completion the student knows different estimation methods for estimating efforts and costs of software projects. He or she understands the underlying principles of project management software. He or she is well aware of test methods, risk management and maturity models, their usage and their benefits. *Disciplinary / professional skills* On completion the student is able to organize projects of different kinds and sizes and to plan and to control projects using project plans. He or she

	is able to select and to use appropriate cost estimation methods and project management software in practical projects. He or she is able to apply methods for risk management and to use maturity models for improving processes.
Index	■ Brief recap of software project management basics (e.g. work break down structure, project organization) ■ Methods for planning and controlling projects ■ Estimation methods: efforts, costs ■ Network planning techniques ■ Project management software ■ Testing methods ■ Risk management ■ Maturity models
Pre-Exam Requirements	Presentation in class
Method and Extent of Examination	Oral Examination, 20 minutes
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ Ahren, D.M., A. Clouse, R. Turner: CMMI Distilled 2nd Edition. Addison Wesley, 2003. ■ Futrell, R.T., D.F. Shafer, L.I. Shafer: Quality Software Project Management. Software Quality Institute Series. Prentice Hall, 2002. ■ IEEE Guide to the Software Engineering Body of Knowledge (SWEBOK); IEEE, 2004. ■ PMI Standards Committee: A Guide to the Project Management Body of Knowledge (PMBOK); Project Management Institute, 3rd edition, 2004. ■ subject specific additional literature, project management software

4 Software Architecture

Course	Master Software Technology, Digital Processes
Name of Module	Software Architecture
Abbreviation	SWA
Semester	1
Responsible	Prof. Dr. Deininger
Lecturers	Prof. Dr. Deininger, Prof. Dr. Wanner, Prof. Dr. Speiser
Elective / compulsory	Compulsory module
Frequency	Every year

Weekly Contact Hours	4h (3h Lectures + 1h Exercises)
Method of Teaching	Lecture with theoretical and practical exercises
Student Work Load – Lectures	68h
Student Work Load – Self Studies	112h
European Credit Transfer Points	6 ECTS Points
Necessary Previous Knowledge	Software Engineering, Object Oriented Software Implementation
Final Knowledge and Skills	*Knowledge and understanding* On completion the student knows the different interrelationships between requirements and design and architectural choices. He or she knows the principles of software design and the different design views and knows how a system design affects the testability of a system. *Disciplinary / professional skills* On completion the student is able to develop different design views and select fitting patterns for certain problems and draw from architectural choices. He or she is able to select and use appropriate modeling techniques. He or she can rate the consequences of certain design decisions Students can earn the official iSAQB certificate at the end of the semester.
Index	*Based on iSAQB Certified Professional for Software Architecture Foundation Level (https://www.isaqb.org/):* ■ Basic Concepts ■ Roles, Responsibilities, and Activities of Architects ■ Deriving Quality Goals and Design Constraints ■ Design Principles ■ Patterns ■ Additional design considerations ■ Design Approaches and Methods ■ Documentation and Communication ■ Software Architecture Evaluation ■ Domain Driven Design ■ QA-Architecture
Pre-Exam Requirements	Assignments
Method and Extent of Examination	Written examination, 120 minutes
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ Bass, L., P. Clements, R. Kazman: Software Architecture in Practice, 3nd edition, Addison-Wesley Professional, 2012.

	■ Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: Pattern-Oriented Software Architecture: A System of Patterns, John Wiley & Sons, 1996. ■ Clements, P., F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, J. Stafford: Documenting Software Architectures, 2nd edition, Addison-Wesley, 2010. ■ Evans, E.: Domain-Driven-Design, Addison- Wesley, 2008. ■ Fowler, M.: Patterns of Enterprise Application Architecture; Addison-Wesley, 2014. ■ Gamma, E., R. Helm, R. Johnson, J. Vlissides: Design Patterns: Elements of Reusable OO Software. Addison-Wesley, 1997. ■ Martin, R.C.: Clean Architecture: A Craftsman's Guide to Software Structure and Design: A Craftsman's Guide to Software Structure and Design Addison-Wesley, 2018 ■ Meyer, B.: Object-Oriented Software Construction. Prentice Hall, 1997. ■ Shaw, M., P. Clements: The Golden Age of Software Architecture, IEEE SOFTWARE, March/April 2006, 31-39.

5 Intercultural Training

Course	Master Software Technology
Name of Module	Intercultural Training
Abbreviation	ICT
Semester	1 and 2

Anmerkung: Intercultural Training 1 und 2 wird zu Beginn des ersten und zweiten Semesters als Blockveranstaltung durchgeführt und dient im Wesentlichen zur Integration der neuen Studenten und insbesondere der Quereinsteiger in die bestehende Gruppe. Aus diesem Grund überschneiden sich auch die Inhalte der beiden Module.

Course	Master Software Technology
Name of Module	Intercultural Training (Part 1)
Abbreviation	ICT1
Semester	1
Responsible	Course Director Software Technology
Lecturers	*External Lecturer*
Elective / compulsory	Compulsory module
Frequency	Every year
Weekly Contact Hours	1h (held as a Block-course)

Method of Teaching	Simulated Activities, Case Studies, Critical Incidents, Film Analyses, Role Playing, Lecture Input, Group Discussions, Reports.
Student Work Load – Lectures	12h
Student Work Load – Self Studies	18h
European Credit Transfer Points	1 ECTS Point
Necessary Previous Knowledge	none
Final Knowledge and Skills	*Knowledge and understanding* On completion the student knows the theoretical bases of intercultural discipline and the rationale behind intercultural learning. He or she has a deeper understanding of communications, decision making, and cultural differences. Also, he or she knows better about their classmates and their cultural background. *Disciplinary / professional skills* On completion the student is able to sense cultural differences and adapt his communication and decisions to multi-cultural workplace.
Index	■ Theoretical Bases: What is Culture? Understanding Concepts of Interculturality, Multiculturality, Diversity, Cultural Programming, Cultural Perception. ■ Working with Hofstede´s Dimensions of Culture: Individualist vs. Collectivist Culture; Dealing with Power and Hierarchy; Monolinear and Polylinear Culture; Gender and Culture. ■ Skills and Processes: Perceiving; Communicating; Managing Cultural Conflict; Coping with Diversity
Pre-Exam Requirements	none
Method and Extent of Examination	Individual Participation, Group Input, Reporting, Project Submission (ungraded)
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	none

Course	Master Software Technology
Name of Module	Intercultural Training (Part 2)
Abbreviation	ICT2
Semester	2
Responsible	Course Director Software Technology

Lecturers	Prof. Dr. Melinda Madew
Elective / compulsory	Compulsory module
Frequency	Every year
Weekly Contact Hours	1h (held as a Block-course)
Method of Teaching	Simulated Activities, Case Studies, Critical Incidents, Film Analyses, Role Playing, Lecture Input, Group Discussions, Reports.
Student Work Load – Lectures	30h
Student Work Load – Self Studies	none
European Credit Transfer Points	1 ECTS Point
Necessary Previous Knowledge	none
Final Knowledge and Skills	*Knowledge and understanding* On completion new students know the theoretical bases of intercultural discipline and the rationale behind intercultural learning. He or she has a deeper understanding of communications, decision making, and cultural differences. All new students know better about their classmates and their cultural background. *Disciplinary / professional skills* On completion the student is able to sense cultural differences and adapt his communication and decisions to multi-cultural workplace.
Index	■ Integration of Newcomers ■ Wrap-up for Newcomers: Theoretical Bases, Working with Hofstede's Dimensions of Culture, Skills and Processes ■ Further Skills and Processes: Perceiving; Communicating; Managing Cultural Conflict; Coping with Diversity
Pre-Exam Requirements	none
Method and Extent of Examination	Individual Participation, Group Input, Reporting, Project Submission (ungraded)
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	none

6 Elective Module 1

Course	Master Software Technology

Name of Module	Elective Module 1
Abbreviation	EL1
Semester	1

6.1 Computational Intelligence

Course	Master Software Technology
Name of Module	Computational Intelligence
Abbreviation	CI
Semester	1
Responsible	Prof. Dr. Jörg Homberger
Lecturers	Prof. Dr. Jörg Homberger, Prof. Dr. Ulrike Padó
Elective / compulsory	Elective module
Frequency	Every year
Weekly Contact Hours	4h (3h Lectures + 1h Exercises/Presentations)
Method of Teaching	Lecture with theoretical and practical exercises; independent group project with presentation
Student Work Load – Lectures	68 h
Student Work Load – Self Studies	112 h
European Credit Transfer Points	6 ECTS Points
Necessary Previous Knowledge	Basic principles of data structures and algorithms
Final Knowledge and Skills	*Knowledge and understanding* On completion the student understands algorithms for complex optimization and learning problems. He or she knows about application areas of these methods like Advanced Planning Systems. Moreover, the students learn to program and optimize the hyperparameter of machine learning systems. *Disciplinary / professional skills* On completion the student is able to select and implement an appropriate algorithm for a given problem.
Index	■ Evolutionary Algorithms ■ Neural Networks and Deep Learning ■ Support Vector Machines

	■ Automated Machine Learning
Pre-Exam Requirements	none
Method and Extent of Examination	Project Work (graded)
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ Aggarwal, C.C.: Neural Networks and Deep Learning. Springer. 2018. ■ Eiben, A.E., J.E. Smith: Introduction to Evolutionary Computing. Springer, Berlin, 2003. ■ Gridin, I.: Automated Deep Learning Using Neural Network Intelligence. Springer. 2022. ■ Homberger, J.: A parallel genetic algorithm for the multi-level unconstrained lot-sizing problem. *INFORMS Journal on Computing,* 2008. ■ Kruse, R., Mostaghim, S., Borgelt, C., Braune, C., Steinbrecher, M.: Computational Intelligence. Springer. 2022. ■ Lam, H.K., Ling, S.H., Nguyen (eds.): Computational Intelligence and its Applications. Evolutionary Computation, Fuzzy Logic, Neural Network and Support Vector Machine Techniques. World Scientific. 2012. ■ Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT. 2002. ■ Schwefel, H.-P.: Evolution and Optimum Seeking. Wiley, New York, 1995.

6.2 IT-Security

Course	Master Software Technology
Name of Module	IT-Security
Abbreviation	ITS
Semester	1
Responsible	Prof. Dr. Jan Seedorf
Lecturers	Prof. Dr. Jan Seedorf, *External Lecturer*
Elective / compulsory	Elective module
Frequency	Every year
Weekly Contact Hours	4h (3h Lectures + 1h Exercises/Presentations)
Method of Teaching	Lecture with theoretical and practical exercises; independent group project with presentation
Student Work Load – Lectures	68 h
Student Work Load – Self Studies	112 h

European Credit Transfer Points	6 ECTS Points
Necessary Previous Knowledge	■ Background in Computer Science and Information Technology ■ Basic understanding of the issues of IT-Security ■ Programming experience
Final Knowledge and Skills	After successful participation in the course, students are able to critically discuss common mechanisms and protocols for increasing the IT security of today's systems. Students have a broad knowledge of IT security, data protection, and privacy on the Internet after completing the course. Students are familiar with modern IT protection concepts from the areas of cryptography, identity management, web, and mobile security, and secured programming practices in SDLC. They are able to recognize attack vectors in IT systems and develop countermeasures.
Index	■ Basic principles of IT-Security ■ Attack vectors ■ Security Models ■ Network Security ■ Web and Mobile Security ■ Distributed System Security ■ Decentralized System Security (Block Chain, SSI) ■ Authentication and Access Control Mechanisms (OIDC, OAuth 2.0, MFA) ■ Cryptographic Solutions (Message Digests (SHA, MD5), digital signatures (DSA), certificates and PKIs) ■ Cloud Computing ■ Privacy and Data Protection ■ Security in Software Development Life Cycle
Pre-Exam Requirements	none
Method and Extent of Examination	Project Work (graded)
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ J. Buchmann: Introduction to Cryptography. Springer, 2000 ■ D. Gollmann: Computer Security. Wiley & Sons, 2011 ■ W. Stallings: Cryptography and Network Security - Principles and Practice. Prentice-Hall, 2003 ■ S. Garfinkel und G. Spafford: Practical Unix & Internet Security, O'Reilly & Associates ■ Henk C.A. van Tilborg, "Encyclopedia of Cryptography and Security", ISBN-13: 978-0387234731 ■ C. Adams / S. Lloyd, "Understanding Public-Key Infrastructure", ISBN 1-57870-166-X ■ Alex Preukschat, Drummond Reed, “Self-Sovereign Identity”, ISBN 9781617296598

6.3 Additional Elective Module 1

Course	Master Software Technology
Name of Module	Additional Elective Module 1
Abbreviation	ADD1
Semester	1
Responsible	Course Director of Software Technology
Lecturers	*depending on actual topic*
Elective / compulsory	Elective module
Frequency	Every year
Weekly Contact Hours	4h
Method of Teaching	
Student Work Load – Lectures	68h
Student Work Load – Self Studies	112h
European Credit Transfer Points	6 ECTS Points
Method of Teaching	*depending on actual topic*
Necessary Previous Knowledge	*depending on actual topic*
Final Knowledge and Skills	*depending on actual topic*
Index	*depending on actual topic*
Method and Extent of Examination	*to be defined by the examination board*
Pre-Exam Requirements	*to be defined by the examination board*
Media	Moodle, *depending on actual topic*
Recommended Literature (Excerpt)	*depending on actual topic*

7 Modern Distributed Systems

Course	Master Software Technology

Name of Module	Modern Distributed Systems
Abbreviation	MDS
Semester	2
Responsible	Prof. Dr. Wanner
Lecturers	Prof. Dr. Wanner, *External Lecturer*
Elective / compulsory	Compulsory module
Frequency	Every year
Weekly Contact Hours	4h (2h Lectures + 2h Exercises/Lab Sessions)
Method of Teaching	Lectures with Exercises, Lab Work (Weekly Assignments)
Student Work Load – Lectures	68h
Student Work Load – Self Studies	112h
European Credit Transfer Points	6 ECTS Points
Necessary Previous Knowledge	■ Object Oriented Programming ■ Distributed Systems
Final Knowledge and Skills	*Knowledge and understanding* On completion, students will understand the principles and benefits of cloud computing, microservices architecture and containerization with appropriate communication and persistence. *Disciplinary/Professional Skills* On completion, students will be able to design and develop cloud-native applications using microservices architecture. They will be able to effectively select and utilize cloud service models and providers, employ CI/CD pipelines. Students will have containerization skills using Docker. They can design and use RESTful APIs and integrate different components in distributed systems. Be able to utilize persistence in distributed systems.
Index	■ Introduction to Cloud Computing ■ Cloud-Native Development ■ Dev Tools for Cloud-Native Development ■ Containerization with Docker ■ RESTful architectures and RESTful APIs ■ Introduction to the Spring Boot framework ■ Object-Relational Mapping (ORM) and JPA
Pre-Exam Requirements	Assignments

Method and Extent of Examination	Written examination, 120 minutes
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ Bauer, C., & King, G.: Java Persistence with Hibernate. Manning Publications, 2015. ■ Biehl, M.: RESTful API Design, Independent Publishing, 2016 ■ Gilbert, J.: Cloud Native Development Patterns and Best Practices. Packt Publishing, 2018. ■ Morris, K.: Infrastructure as Code: Managing Servers in the Cloud. O'Reilly Media, 2016. ■ Reznik, P., & Dobson, J.: Cloud Native Transformation: Practical Patterns for Innovation. O'Reilly Media, 2019. ■ Turnbull, J.: The Docker Book: Containerization is the New Virtualization. James Turnbull, 2014. ■ Walls, C. Spring Boot in Action. Manning Publications, 2016.

8 Software Project

Course	Master Software Technology
Name of Module	Software Project
Abbreviation	SOP
Semester	2
Responsible	Course Director Software Technology
Lecturers	All professors of computer science, *External Lecturers*
Elective / compulsory	Compulsory module
Frequency	Every year
Weekly Contact Hours	4h (4h Integrated Lecture/Exercises/Presentations)
Method of Teaching	Project Work, Lecture, Presentations
Student Work Load – Supervision	68h
Student Work Load – Work and Presentation	172h
European Credit Transfer Points	8 ECTS Points
Necessary Previous Knowledge	Software Engineering, Software Project Management (Bachelor Level)

Final Knowledge and Skills	*Knowledge and understanding* On completion the student has knowledge and practical experience of software engineering while developing software in an industry-like project with real customers pertaining to software design, version control, documentation, testing, maintenance and software quality assurance. He or she has an understanding and experience of the difficulties of team management and troubleshooting (due to the size of the project team). *Disciplinary / professional skills* On completion the student is able to cope with typical situations arising in software projects by applying previously learned methods and team management.
Index	An industrial-like software project is supervised by faculty teachers who are usually playing the role of a customer. The software project team (6 – 10 students) participates in all stages of the software process from requirements engineering to roll-out. Beneath technical problem-solving methods of management, leadership, planning, communication and cooperation have to be employed.
Pre-Exam Requirements	Intermediate project presentation
Method and Extent of Examination	Project work, documentation and presentation (graded)
Media	Moodle, Collaboration Tools
Recommended Literature (Excerpt)	■ Beck, K., M. Fowler: Planning extreme programming. Addison-Wesley, Boston, 2001 ■ Dustin, E., J. Rashka, J. Paul: Automated Software Testing: Introduction, Management, and Performance. Addison Wesley Publishing Company, 1999. ■ Haug, M., E.W. Olsen, G. Cuevas, S. Rementeria (Eds.): Managing the Change: Software Configuration and Change Management. Software Best Practice 2, Springer, 2001. ■ Kruchten, P.: The Rational Unified Process: An Introduction (2nd Edition). Addison-Wesley, 2000. ■ Robertson, S. J. Robertson: Mastering the Requirements Process. Addison Wesley, 1999.

9 Software Verification and Validation

Course	Master Software Technology
Name of Module	Software Verification and Validation
Abbreviation	SVV
Semester	2
Responsible	Prof. Dr. Heusch

Lecturers	Prof. Dr. Heusch, *External Lecturer*
Method of Teaching	Lecture with theoretical and practical exercises.
Elective / compulsory	Compulsory module
Frequency	Every year
Weekly Contact Hours	2h (1h Lectures + 1h Exercises/Presentations)
Method of Teaching	Lectures with Exercises, Reading Assignments, Assignments
Student Work Load – Lectures	34h
Student Work Load – Self Studies	56h
European Credit Transfer Points	3 ECTS Points
Necessary Previous Knowledge	Software Engineering basics of Testing and Quality Assurance
Final Knowledge and Skills	*Knowledge and understanding* On completion the student knows about advanced techniques of software validation by using stochastic models and the formal methods for software verification based on first order logic and formal specification. *Disciplinary / professional skills* On completion the student is able to select an appropriate stochastic model and validate software. He or she can apply formal specification for program proofing and use Eiffel for creating provably verifiable software.
Index	■ Stochastic models of program behavior ■ First order logic and formal specification ■ Static analysis and program transformations ■ Use of model checking and deductive techniques ■ Using Eiffel for provably verifiable software
Pre-Exam Requirements	Assignments
Method and Extent of Examination	Written examination, 90 minutes
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ Apt, K.R., E.-R. Olderog: Verification of Sequential and Concurrent Programs, Second Edition, Springer, 1997. ■ Meyer, B.: Object Oriented Software Construction. Prentice-Hall, 1997 ■ Wordsworth, J. B.: Software Development with Z - A Practical Approach to Formal Methods in Software Engineering. Addison-Wesley, 1992. ■ Zeller, A.: Why Programs fail: A guide to systematic debugging. dpunkt, 2005.

10 Software Engineering II

Course	Master Software Technology
Name of Module	Software Engineering II (Advanced Topics in Software Engineering)
Abbreviation	SWE
Semester	1
Responsible	Prof Dr. Wanner
Lecturers	Prof. Dr. Deininger, Prof. Dr. Wanner, Prof. Dr. Speiser
Elective / compulsory	Compulsory module
Frequency	Every year
Weekly Contact Hours	4h (3h Lectures + 1h Exercises/Presentations)
Method of Teaching	Lectures with Exercises, Reading Assignments, Assignments
Student Work Load – Lectures	68h
Student Work Load – Self Studies	112h
European Credit Transfer Points	6 ECTS Points
Necessary Previous Knowledge	■ Programming experience ■ Software Engineering (Bachelor Level)
Final Knowledge and Skills	*Knowledge and understanding* Upon completion, students will have a deeper understanding of the software development process. He or she will know state-of-the-art approaches to developing and operating software systems and advanced programming techniques. *Disciplinary / professional skills* On completion, students will be able to select and apply an appropriate approach to development and operation. Students will be able to perform extensive testing and apply advanced programming techniques to real-world problems.
Index	■ Architecture Principles ■ DevOps ■ Model-Driven SW Development ■ Large-Scale SW Development ■ Test-Driven Development and test techniques ■ Advanced programming techiques: Reflective Programming, Parallel Programming, Reactive Programming, Event Driven Programming

Pre-Exam Requirements	Assignments
Method and Extent of Examination	Written examination, 120 minutes
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ Beck, Kent: Test-Driven Development by Example, Addison-Wesley, 2002. ■ Bloch, Joshua: Effective Java, Addison-Wesley, 3rd Edition, 2017 ■ Cohn, Mike: Succeeding with agile: Software Development using Scrum, Addison Wesley, 2010. ■ Czarnecki, Krzysztof, Ulrich W. Eisenecker: Generative Programming, Addison-Wesley, 2000. ■ Fowler, Martin: Refactoring, Addison Wesley. Boston, 2001. ■ Humble, Jez, David Farley: Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, Addison-Wesley, 2010 ■ Martin, Robert. C.: Agile Software Development. Prentice Hall, 2003. ■ Martin, Robert C.: Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hal, 2008. ■ Palmer, Stephen R.: A Practical Guide to Feature-Driven Development. Prentice Hall, 2002. ■ Utting, Mark, Bruno Legeard: Practical Model-Based Testing. Elsevier, Morgan Kaufmann, 2014 Vocke, Ham: The Practical Test Pyramid. https://martinfowler.com/articles/practical-test-pyramid.html, 2018

11 Elective Module 2

Course	Master Software Technology
Name of Module	Elective Module 2
Abbreviation	EL2
Semester	2

11.1 Business Process Technology

Course	Master Software Technology
Name of Module	Business Process Technology
Abbreviation	BPT
Semester	2

Responsible	Prof. Dr. Höß
Lecturers	Prof. Dr. Höß, Prof. Dr. Kramer, Prof. Dr. Lückemeyer
Elective / compulsory	Elective module
Frequency	Every year
Weekly Contact Hours	4h (3h Lectures + 1h Exercises/Presentations)
Method of Teaching	Lecture with theoretical and practical exercises
Student Work Load – Lectures	68h
Student Work Load – Self Studies	112h
European Credit Transfer Points	6 ECTS Points
Necessary Previous Knowledge	Programming, Middleware Technology, Basic internet technology (XML, DTD, XML Schema, Namespaces, XPath, XSL, XSLT, DOM)
Final Knowledge and Skills	*Knowledge and understanding* On completion the student knows how to model inter- and intra-organizational business processes and technical workflows. He or she understands the concepts of orchestration of services and the choreography between services. He or she knows the necessary concepts and technologies for service-enabling existing legacy applications. *Disciplinary / professional skills* On completion the student is able to select and use modeling techniques for business processes and technical workflows and apply these techniques in a practical project. He or she can rate the technological and organizational implications when an IT landscape in a company has to be transformed into a process-oriented SOA.
Index	■ Guided self-study for recapitulating basic internet technologies (XML, DTD, XML Schema, Namespaces XSLT, DOM) ■ Business Process Management (BPM) ■ Business Process Modeling with BPMN ■ Service-oriented Architectures and Web Services (SOAP, WSDL, …) ■ Orchestration of services with BPEL ■ Design and implementation of exemplary business processes which span across several IT systems ■ Case studies and practical examples
Pre-Exam Requirements	Assignments, System exercises and presentations
Method and Extent of Examination	Written examination, 90 minutes
Media	Blackboard, Powerpoint, Computer Demonstration, moodle

Recommended Literature (Excerpt)	■ Allweyer, T.: BPMN 2.0 – Introduction to the Standard for Business Process Modeling. Books on Demand, 2010. ■ Cummins, F.: Building the Agile Enterprise: With SOA, BPM and MBM. Morgan Kaufmann, 2009. ■ Krafzig, D., K. Banke, D. Slama: Enterprise SOA: Service Oriented Architecture Best Practices. Prentice Hall, 2005. ■ Margolis, B.: SOA – Concepts, BPEL and SCA. MC Press, 2007. ■ Weske, M.: Business Process Management – Concepts, Languages, Architectures. Springer, 2007. ■ Weerawarana, S., F. Curbera, F. Leymann, T. Storey et al.: Web Services Platform Architecture. Prentice Hall, 2005. ■ Current research papers and online material on BPM, SOA, BPMN, BPEL and other topics

11.2 Business Intelligence

Course	Master Software Technology
Name of Module	Business Intelligence
Abbreviation	DWH
Semester	2
Responsible	Prof. Koch
Lecturers	Prof. Koch
Elective / compulsory	Elective module
Frequency	Every year
Weekly Contact Hours	4h (3h Lectures + 1h Exercises/Presentations)
Method of Teaching	Lecture with theoretical and practical exercises; independent group project with presentation
Student Work Load – Lectures	68h
Student Work Load – Self Studies	112h
European Credit Transfer Points	6 ECTS Points
Necessary Previous Knowledge	Database theory (especially normal forms, relational algebra, design procedures), relational systems, SQL, Middleware Technology, Bachelor-level mathematics
Final Knowledge and Skills	*Knowledge and understanding*

	On completion the student has a deeper understanding of goals and functionality of data warehouse systems. He or she has practical experience with a data warehouse system and insight into current business intelligence research issues. *Disciplinary / professional skills* On completion the student is able to evaluate strengths and weaknesses of data warehouse systems, to build a data warehouse system, and to make informed decisions about different situations of data warehouse usage in practical projects within enterprise contexts.
Index	■ Purposes and application areas for data warehouses, case studies, comparison to database systems and transaction processing systems ■ Reference model for data warehouses, data acquisition, monitoring, extraction, transformation, loading, data marts versus data warehouse, data warehouse bus architecture ■ Data analysis: OLAP, data mining (statistical methods, regression, value prediction, decision trees, association discovery, a priori method, neural networks, visualization). ■ System architectures with middleware, web based architectures ■ Multidimensional models and algebra ■ Conceptual and physical modeling: multidimensional entity relationship model, schema evolution, star join schemas, snow flaking, array structures, performance optimization (materialized views, efficient indexing techniques) ■ Implementation of data warehouses with different DBMS types, ROLAP, MOLAP, HOLAP; OLAP extensions of SQL
Pre-Exam Requirements	Successful seminar paper and presentation
Method and Extent of Examination	Written examination, 90 minutes
Media	Blackboard, Powerpoint, Computer Demonstration, moodle
Recommended Literature (Excerpt)	■ Adamson, C., M. Venerable: Data Warehouse Design Solutions. Wiley, 1998. ■ Bauer, A., H. Günzel: Data Warehouse Systeme - Architektur, Entwicklung, Anwendung. dpunkt Verlag, 2008. ■ Kimball, R.: The Data Warehouse Toolkit - Practical Techniques for Building Dimensional Data Warehouses. Wiley, 1996. ■ Kimball, R., L. Reeves, M. Ross, W. Thornthwaite: The Data Warehouse Lifecycle Toolkit - Expert Methods for Designing, Developing, and Deploying Data Warehouses. Wiley, 1998. ■ Inmon, W.H.: Building the Data Warehouse. Wiley, 1996. ■ Humphries, M., M.W. Hawkins, M.C. Dy: Data Warehousing - Architecture and Implementation. Prentice Hall, 1999. ■ Course material, additional up-to-date articles available online in the Moodle System

11.3 Additional Elective Module 2

Course	Master Software Technology
Name of Module	Additional Elective Module 1
Abbreviation	ADD2
Semester	2
Responsible	Course Director of Software Technology
Lecturers	*depending on actual topic*
Elective / compulsory	Elective module
Frequency	Every year
Weekly Contact Hours	4h
Method of Teaching	
Student Work Load – Lectures	68h
Student Work Load – Self Studies	112h
European Credit Transfer Points	6 ECTS Points
Method of Teaching	*depending on actual topic*
Necessary Previous Knowledge	*depending on actual topic*
Final Knowledge and Skills	*depending on actual topic*
Index	*depending on actual topic*
Method and Extent of Examination	*to be defined by the examination board*
Pre-Exam Requirements	*to be defined by the examination board*
Media	Moodle, *depending on actual topic*
Recommended Literature (Excerpt)	*depending on actual topic*

12 Master Thesis

Course	Master Software Technology

Name of Module	Master Thesis
Abbreviation	MT
Semester	3
Responsible	Supervising professor
Lecturers	-
Elective / compulsory	Compulsory module
Frequency	Every semester
Weekly Contact Hours	-
Method of Teaching	Lecture with theoretical and practical exercises; independent group project with presentation
Student Work Load	900h Total
European Credit Transfer Points	30 ECTS Points
Necessary Previous Knowledge	Modules of the previous semesters (at least 40 ECTS); the thesis topic has to be relevant to the taught modules of the master course.
Final Knowledge and Skills	On completion the student shows that he or she can solve in a predefined period a problem of his or her domain independently using scientific methods. This includes ■ sustained independent work of high quality fulfilling an agreed specification, ■ performing a critical review of research literature in the field of information technology, ■ analysis, synthesis and creative application of what has been learned in previous courses, ■ creation of a detailed and coherent report, in which the thesis work is presented in the context of the problem domain, with solutions proposed or implemented, justified and a critical appraisal of the work done.
Index	Depends on the actual domain specific topics; typically the thesis consists of one or more of the following activities: ■ Production of a detailed specification or design for a software system, or the implementation of one. Usually this includes a critical evaluation of the requirements and in the assessment of alternative tools, methods and solutions that could be employed and a conclusion which justifies the particular choices made. ■ Evaluation of some existing tools or technique or software system. Usually this includes the development and application of criteria in performing such an assessment. ■ Gathering of empirical evidence by directly testing existing tools or software system, and/or by seeking information from those who use (or would use in the case of a system to be developed) the system about aspects of its use. Usually this includes a justification of the

	approach taken in obtaining such evidence and in supporting the conclusions that can be drawn (or not drawn) from it.
Pre-Exam Requirements	none
Method and Extent of Examination	Written report, abstract, oral presentation
Media	*depending on actual topic*
Recommended Literature (Excerpt)	*depending on actual topic*