Specialization in Technology: Informatics (major) | Technik-Schwerpunkt: Informatik Vertiefungsmodule (major)

Module Description

CIT3330002: IT Security 2 | IT-Sicherheit 2

Version of module description: Gültig ab summerterm 2023

Module Level: Bachelor/Master	**Language:** German	**Duration:** one semester	**Frequency:** summer semester
Credits:* 5	**Total Hours:** 150	**Self-study Hours:** 90	**Contact Hours:** 60

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The examination is performed in the form of a 90-minute exam. The exam questions test whether the candidate has acquired a subset of the skills in the list below.

List of Skills: The student
- masters basic and advanced concepts and techniques to secure systems and individual artifacts,
- can apply learned techniques correctly,
- is able to identify security problems using concrete examples,
- knows the advantages and disadvantages of the most important classical and advanced security concepts.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:
IN0042 IT Security, IN0010 Introduction to Computer Networking and Distributed Systems, IN0009 Basic Principles: Operating Systems and System Software

Content:
The module treats selected types in depth and discusses special topics of IT security. Current concepts and solutions in the field of digital identity, such as smart cards, physically unclonable functions (PUF), SSI and token-based authentication in distributed systems are dealt with in depth. In the area of application security, selected issues such as the security of Instant Messenger services are discussed. In the area of system security, the module is dedicated to advanced concepts such as Trusted Computing, Trusted Execution Environments and Confidential

Computing. It also discusses selected aspects of hardware-security. The module deals with the current and developing state of the security of wireless and mobile communication architectures (e.g. 5G) and with concepts for ad-hoc security in networked IoT devices (e.g. BluetoothLE). The module will also give an insight into the methodological development and evaluation of secure systems (security engineering).

Intended Learning Outcomes:
After participating in the module, students are able to evaluate complex security solutions. They are able to analyze the security level of applications. They will be able apply basic and advanced security concepts in a correct manner to solve security problems. They understand the causes of security problems in networks but also in mobile and embedded systems. They master the most important concepts for their protection and can also apply advanced approaches. They are able to identify possible security weaknesses in designs or protocols and to develop solutions based on the methods and concepts they have learned.

Teaching and Learning Methods:
The module consists of a lecture and an accompanying exercise. In the lecture, the teaching content is conveyed and the students are encouraged to study the literature and to deal with the content of the topics. In the exercises, concrete questions are discussed and examples are worked on, in groups. With 6-8 exercise sheets, the students are guided to work through the topics of the lecture and to deepen the material of the lecture using concrete tasks.

Media:
Slides

Reading List:
- IT-Sicherheit: Konzepte, Verfahren, Protokolle, Claudia Eckert, 11. Auflage, De Gruyter, 2023. (in german)
- William Stallings, Lawrie Brown: Computer Security: Principles and Practice, 2018
- Understanding Cryptography, C. Paar und J. Pelzl, Springer

Responsible for Module:
Eckert, Claudia; Prof. Dr.

Courses (Type of course, Weekly hours per semester), Instructor:
IT Sicherheit 2 (CIT3330002) (Vorlesung mit integrierten Übungen, 4 SWS)
Eckert C [L], Eckert C, Kilger F, von Tschirschnitz M
For further information in this module, please click campus.tum.de or here.

Module Description

CIT4230000: Strategic IT Management | Strategic IT Management

Version of module description: Gültig ab winterterm 2022/23

Module Level: Master	**Language:** English	**Duration:** one semester	**Frequency:** winter semester
Credits:* 4	**Total Hours:** 120	**Self-study Hours:** 75	**Contact Hours:** 45

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The examination will be in the form of a written exam (90 minutes) in which the students' ability to describe, evaluate and apply models, methods, key figures, visualizations and tools of strategic IT management to given problems in a context-dependent manner will be tested. Successful participation in a voluntary case study or a workshop throughout the exercise can be included as a bonus in the assessment of the exam. The exact regulations for this will be announced in time at the beginning of the module.

Repeat Examination:

(Recommended) Prerequisites:
Bachelor in Informatics or Information Systems, specifically:
- Fundamentals of Business Information Systems
- Databases
- Software Engineering
- Business Process Management

Content:
1. IT Strategy and IT Management Frameworks
2. IT Governance and IT Organisation
3. Enterprise Architecture Management
3.1 Schools of EAM
3.2 Foundations of EAM
3.3 Capability Based Planning
3.4 Agile EAM and EAM Patterns
4. Large-scale Agile Software Development
4.1 Foundations and Frameworks of LSAD

4.2 Selected Topics of   LSAD
5. Case Studies

Intended Learning Outcomes:
After participating in the module, students will be able to understand the key challenges, concepts, methods and decision areas of strategic IT management and their interactions. Furthermore, students will understand the relationship between strategic IT management, enterprise architecture management, scaled agile software development and IT governance. Moreover, after participating in the module, students have in-depth knowledge of patterns and frameworks of enterprise architecture management and scaled agile software development in the areas of models, methods, metrics and visualizations, among others.

Teaching and Learning Methods:
With the help of a slide presentation, the lecture introduces the fundamental concepts of strategic IT management. Furthermore, the understanding of the basic concepts of strategic IT management is deepened in the lectures with the help of appropriate tasks and examples. Through exercises during the lecture and the work on a separate case study, special modeling techniques from enterprise architecture management and from scaled agile software development are explained and practiced. In addition, self-study assignments are provided.

Media:
Slide presentation, teamwork

Reading List:
Hanschke, I. (2013). Strategisches Management der IT-Landschaft: Ein praktischer Leitfaden fu#r das Enterprise Architecture Management. Carl Hanser Verlag GmbH Co KG.

Kaplan, J. D. (2005). Strategic IT portfolio management: governing enterprise transformation. PRTM.

Buckl, S., Ernst, A. M., Matthes, F., Ramacher, R., & Schweda, C. M. (2009, September). Using enterprise architecture management patterns to complement TOGAF. In 2009 IEEE International Enterprise Distributed Object Computing Conference (pp. 34-41). IEEE.
Buckl, S., Ernst, J., Lankes, A. M., Matthes (2008).
Enterprise architecture management pattern catalog (version 1.0, february 2008)

Sandkuhl, K., Fill, H. G., Hoppenbrouwers, S., Krogstie, J., Matthes, F., Opdahl, A., ... & Winter, R. (2018). From expert discipline to common practice: a vision and research agenda for extending the reach of enterprise modeling. Business & Information Systems Engineering, 60(1), 69-80. Chicago

Uludag, O#., Kleehaus, M., Caprano, C., & Matthes, F. (2018, October). Identifying and structuring challenges in large-scale agile development based on a structured literature review. In 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC) (pp. 191-197). IEEE.

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. Journal of Systems and Software, 119, 87-108.
Paasivaara, M., Behm, B., Lassenius, C., & Hallikainen, M. (2018). Large-scale agile transformation at Ericsson: a case study. Empirical Software Engineering, 23(5), 2550-2596.
Chicago

Responsible for Module:
Matthes, Florian; Prof. Dr. rer. nat.

Courses (Type of course, Weekly hours per semester), Instructor:
Strategisches IT-Management (CIT4230000) (Vorlesung, 3 SWS)
Matthes F [L], Matthes F, Tobisch F
For further information in this module, please click campus.tum.de or here.

Module Description

ED140016: Computational Flow Stability and Transition | Computational Flow Stability and Transition

Version of module description: Gültig ab summerterm 2025

Module Level: Bachelor/Master	**Language:** English	**Duration:** one semester	**Frequency:** winter/summer semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 105	**Contact Hours:** 75

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The examination in this module consists in (25%) the completion of weekly exercises. Students have to prove their knowledge of linear stability tools, the relevant cases of application to derive physical understanding from them and their ability to use state-of-the-art libraries
The examination in this module consists in (50%) the oral presentation of a research article. Students have to prove their understanding of the application of the linear stability methods in research publication.
The examination in this module consists in (25%) the completion of a final project. The students have to show their ability to apply linear stability tools implemented in state-of-the-art libraries to a model problem.

Repeat Examination:
Next semester / End of Semester

(Recommended) Prerequisites:
Python programmation ; Stabilit theory ; Fluid dynamics ; Hydrodynamics

Content:
The course offers core knowledge on computational methods to solve large linear systems stemming from global operator-based analysis (stability, transient growth, sensitivity, resolvent) of physical problems, such as fluid dynamics. The course systematically progresses from the foundational derivation of the basic iterative algorithms to the use of state-of-the-art, high performance, Python librairies implementing these methods. Throughout the course, special emphasis is placed on identifying the physical knowledge that can be drawn from each the operator-based tool and the relevant cases of application.
The following content will be addressed:
1. General principles of linear satbility analysis and example cases in fluid dynamics

2. Introduction to non-normality, its measure, and transient temporal growth
3. Introduction to the pseudo-spectrum of an operator and the methods to compute it. Link to the transient growth and pseudo-resonance.
4. Sensitivity of the spectrum of an operator and the methods for eigenvalues tracking
5. Introduction to sparse matrices and sparse operations
6. Iterative methods for solving large linear systems and eigenvalue problems (Richardson, Arnoldi, Lanczos, conjugate gradient) and spectral transformations
7. Use of PETSc/SLEPc libraries
8. Itroduction to resolvent analysis and the action of an operator

By the course‘s completion, students are expected to have a comprehensive understanding of iterative computational methods applicable to global operator-based analysis and the capability to apply it to a variety of problems.

More info on the course's webpage: https://www.epc.ed.tum.de/tfd/lehre/computational-methods-for-operator-based-analysis/

Intended Learning Outcomes:
After successful participation in the module, students are able to
- know the linear stability tools
- recognize the relevant cases of application of linear stability tools to derive physical understanding from them
- use dedicated iterative methods for large linear systems
- provide a critical analysis on the use of iterative methods for large linear systems in research publications
- use state-of-the-art libraries implementing iterative methods for large linear systems
- apply iterative methods for large linear systems on a model problem.

Teaching and Learning Methods:
The learning is weekly organized around a lecture and a lab session. The theoretical knowledge is delivered during the lecture, where active learning is fostered with Kahoot-type questions and small coding tasks. The theoretical knowledge is applied during the lab session to implement the methods and solve a model problem in Python. The coding tasks addressed during the lecture serve as a basis for the lab session.
The students are confronted with the application of computational methods for operator-based analysis to research problems by presenting a research article selected from a pool of pre-selected articles.
During the last third of the semester, the students will carry out a project in groups of two. The project assesses the student's capability to apply the methods learned to model problems.

Media:
Slides and ineractive exercises. Programming with Python.

Reading List:
P. J. Schmid. Nonmodal Stability Theory. Annu. Rev. Fluid Mech., 39:129–62, 2007.

Peter J. Schmid and Dan S. Henningson. Stability and Transition in Shear Flows. Number 142 in Applied Mathematical Sciences. Springer, New York Berlin Heidelberg, 2001. ISBN 978-1-4612-6564-1 978-0-387-98985-3. doi: 10.1007/978-1-4613-0185-1.
L.N. Trefethen and M. Embree. Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and
Operators. Princeton University Press, 2005.

Responsible for Module:
Polifke, Wolfgang; Prof. Dr.

Courses (Type of course, Weekly hours per semester), Instructor:
Seminar on Computational Methods for Operator-Based Analysis (Seminar, 2 SWS)
Polifke W [L], Brokof P, Varillon G

Computational Flow Stability and Transition (Vorlesung, 2 SWS)
Polifke W [L], Brokof P, Varillon G

Exercises on Computational Methods for Operator-Based Analysis (Übung, 2 SWS)
Polifke W [L], Brokof P, Varillon G
For further information in this module, please click campus.tum.de or here.

Module Description

ED160003: Applied Data Analytics and Machine Learning in Python | Applied Data Analytics and Machine Learning in Python [ADAM]

Version of module description: Gültig ab winterterm 2022/23

Module Level: Master	**Language:** English	**Duration:** one semester	**Frequency:** winter semester
Credits:* 4	**Total Hours:** 120	**Self-study Hours:** 60	**Contact Hours:** 60

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The module examination takes place in the form of exercises. The comprehensive course in-cludes a 20-minute written entrance test (carried out at the beginning of the practical course, un-graded (course work), without any aids) and programming tasks (course work, 5 task blocks) and a written final test (examination, 30 min, no aids admitted). All the tests must be passed.
The entrance test can be held either in attendance or online to check the prerequisites (e.g., a basic understanding of the theoretical principles of Machine Learning, especially of Reinforcement Learning, from an application-oriented point of view). For the entrance test, a question bank and related information material on Machine Learning, Reinforcement Learning, Python, and control processes modeling are provided in advance. Students prepare the answers themselves using the information material. The final questions on the entrance test are randomly selected from the provided question bank.
Every task to be solved must be submitted and accepted by the supervisors to track the learning progress. Only one attempt to rework is allowed per task. The criteria of successful acceptance are formal and syntactical correctness, and the contribution of independently developed solu-tions.
On the last day of the practical course, a 30-minute final exam takes place. The final exam con-sists of short questions, programming tasks, sketches/diagrams, and modeling. It is graded and in written form, either in attendance or as an online test. No aid is permitted. The final grade of the practical course is made up of 100% of the final exam grade.

Repeat Examination:

(Recommended) Prerequisites:
Recommended, but not mandatory, is the successful participation in the lectures „Grundlagen der modernen Informationstechnik“, „Industrial Automation 2“, „Intelligent Systems and Machine Learning for Production Processes“ and „Höhere Mathematik“

Content:
The practical course "Applied Data Analytics and Machine Learning in Python" aims to impart basic skills and abilities in the application-oriented handling of machine learning methods, data mining and Python programming. For this purpose, basic Python knowledge of syntax and common Python frameworks from data analysis and machine learning will be taught or refreshed firstly. It is followed by an introduction to the basic concepts, methods, and algorithms of machine learning and modeling. The acquired background knowledge is then transferred to application-oriented problems from control engineering, process control and quality monitoring by conceptualizing, modeling and implementing solutions for various complex use cases.

Intended Learning Outcomes:
After participating in the practical course, students will be able to
• understand the basic theoretical principles of machine learning, specifically reinforcement learning and unsupervised learning, from an application-oriented perspective.
• use Python and common Python frameworks for data analysis and machine learning, and to design and implement suitable pipelines for data analysis independently. Students will learn to read and understand Python-specific online documentation on open source code, analyze the code behind it, and apply common Python frameworks (e.g., Tensorflow, Keras, scikit-learn). Computational thinking will be strengthened.
• understand complex control problems, Big Data, and Machine Learning problems in an appropriate representation form (flowchart, state diagrams) and implement them in Py-thon.
• model and develop a control task for an automation system as a reinforcement learning task for a given problem.
• assess and critically evaluate the use, effort, and challenges of reinforcement learning for decision making and optimization of automation engineering tasks.

Teaching and Learning Methods:
The module takes the form of an introductory theory course and a practical course. In the theory part, lectures and presentations convey the theoretical background knowledge, allowing the students to work independently. The basic understanding of these theoretical principles is essential in order to solve the subsequent tasks in the practical course. For the subsequent practical blocks, the students will receive an exercise script with tasks of increasing difficulty, which need to be solved either alone or in a small group. For this purpose, demo use cases are considered, for which the taught development steps from problem description to system control are implemented based on reinforcement learning methods. The solutions are documented by each group and discussed in mutual exchange with another group. Subsequently, each group presents the results to the supervisors for final acceptance. By working independently on the tasks and use cases of varying complexity, students actively deal with the problems posed, recognize and overcome typical hurdles in modeling and programming and through hardware limitations, and choose their own pace as far as possible when working on them. In this way, students can learn to assess the effort and benefits of reinforcement learning methods. Students learn to present solutions in a comprehensible way during the review loops, communicate open problems, deal with alternative solutions, and defend their approaches in a scientific discussion.

Media:
Presentations, discussion sessions, active participation, practical exercises with task lists.

Reading List:
Sutton, R.S. & Barto, A.G., 2018. Reinforcement learning: An introduction, MIT press.

Responsible for Module:
Vogel-Heuser, Birgit; Prof. Dr.-Ing.

Courses (Type of course, Weekly hours per semester), Instructor:
Applied Data Analytics and Machine Learning in Python (Praktikum, 4 SWS)
Vogel-Heuser B, Krüger M, Zhang M
For further information in this module, please click campus.tum.de or here.

Module Description

IN0010: Introduction to Computer Networking and Distributed Systems | Grundlagen: Rechnernetze und Verteilte Systeme

Version of module description: Gültig ab winterterm 2011/12

Module Level: Bachelor	**Language:** German	**Duration:** one semester	**Frequency:** summer semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 105	**Contact Hours:** 75

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The exam takes the form of a 90 minutes written test. Comprehension questions and calculation tasks allow to assess acquaintance with the technologies and methods of computer networks and distributed systems, and the understanding obtained by implementation of protocol mechanisms. Calculation tasks also allow to assess the ability to determine the performance of selected computer networks and distributed applications.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:
IN0001 Introduction to Informatics 1, IN0003 Introduction to Informatics 2 - since WiSe 2018/19 Functional Programming and Verification, IN0002 Fundamentals of Programming (Exercises & Laboratory)

Content:
- Computer networking
++ Overview: Computer networking and the Internet
+++ components (router, switches, clients, server)
+++ design (topology, routing, packets vs. virtual circuits)
+++ layered system structure (OSI and Internet)
+++ historical background
++ concepts used by multiple layers (covered within the appropriate layers):
+++ Addressing
+++ Error detection
+++ Coding and modulation
+++ Media access protocols

+++ Flow control
+++ Connection management
+++ Packet vs. virtual circuit switching
++ Layers:
+++ Application layer: application protocols and applications
++++ Tasks and interface
++++ Examples: HTTP, DNS, SMTP (Mail), Peer-to-Peer protocols
+++ Transport layer
++++ Tasks and interface
++++ Examples: TCP and UDP
+++ Network layer
++++ Tasks and interface
++++ Routing: link state vs. distance vector protocols
++++ Addressing: IP Addresses
++++ Examples: IP, Routing in the Internet
+++ Link layer
++++ Tasks and interface
++++ Examples: Ethernet, Wireless LAN
+++ Physical layer
++++ Tasks and interface
++++ Examples
- Distributed systems:
++ Middleware, e.g. RPC
++ Web Services
- General tasks:
++ Network management
++ IT security
+++ Basics of cryptography
+++ Authentication, privacy, integrity
+++ Protocols with security mechanisms, e.g.: IPsec, PGP, Kerberos, SSL, SSH, ...
+++ Firewalls, intrusion detection

Content of the Exercises:
The exercises cover comprehension questions and calculation tasks and target determination of performance of protocols and mechanisms of specific layers (Physical Layer, Data Link Layer, Network Layer, Transport Layer). Programming exercises address implementation of specific protocol mechanisms.

Intended Learning Outcomes:
After successful completion of the module, participants understand the key concepts of technologies and methods of computer networks and distributed systems and are able to use key layered network architecture protocols to explain what protocol mechanisms are used in each layer and how they work. They understand the architecture of distributed applications like the World Wide Web based on Internet protocols, and the architecture of computer networks.

Participants can determine the performance of selected networks and distributed applications, and can implement specific protocol mechanisms.

Teaching and Learning Methods:
The interactive lecture with slide presentations, animations, demonstrations and life programming presents the basic knowledge of computer networks and distributed systems and explains them using examples. Quizzes help students to recognize whether they have understood the basic concepts and essential contexts. Homework enables students to deepen their knowledge in self-study. Accompanying exercises deepen the understanding of the contents of the lecture by means of suitable tasks and show the application of the various methods on the basis of manageable problems. The presentation of the own solution in the accompanying exercise improves the communication skills and allows to compare the own learning progress with that of other students. Programming tasks allow computer-aided deepening and application of conceptual knowledge to practical problems.

Media:
Lecture slides, exercise sheets, demonstrations

Reading List:
Literature is specified at the web presence of the course and in the lecture slides.

Standard publications are among others:
1. James F. Kurose, Keith W. Ross
Computernetzwerke
Pearson Studium; 5. aktualisierte Auflage, 2012
2. Andrew S. Tanenbaum / Prof. David J. Wetherall
Computernetzwerke
Pearson Studium, 5. aktualisierte Auflage, 2012

Responsible for Module:
Carle, Georg; Prof. Dr.-Ing.

Courses (Type of course, Weekly hours per semester), Instructor:
Übungen zu Grundlagen: Rechnernetze und Verteilte Systeme (IN0010), Mo, Di (Übung, 2 SWS)
Carle G [L], Dietze C, Sosnowski M, Günther S (Simon M)

Grundlagen: Rechnernetze und Verteilte Systeme (IN0010) (Vorlesung, 3 SWS)
Günther S [L], Günther S, Carle G
For further information in this module, please click campus.tum.de or here.

Module Description

IN2031: Application and Implementation of Database Systems | Einsatz und Realisierung von Datenbanksystemen

Version of module description: Gültig ab summerterm 2025

Module Level: Bachelor/Master	**Language:** German	**Duration:** one semester	**Frequency:** summer semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 105	**Contact Hours:** 75

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The academic assessment will be done by a written exam of 90 minutes. No aids are permitted. Assignments checking knowledge verify the familiarity with components of modern database systems; programming assignments verify the ability to implement and critically evaluate advanced algorithms and data structures of the database components; small scenarios with defined architectures and applications have to be set up with the methods learnt which verifies the ability to develop precise partial solutions.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:
IN0008 Fundamentals of Databases, IN0007 Fundamentals of Algorithms and Data Structures

Content:
- Error Handling
- Concurrency Control
- Security and Privacy
- Object Oriented und Object Relational Database Concepts
- Deductive Databases
- Distributed Databases
- Business Intelligence: Data Warehousing, Data Mining
- Main Memory Database Systems
- XML and Database Systems
- Database Concepts in the Context of Big Data Applications
- Performance Evaluation

Intended Learning Outcomes:
Students command the components of modern database systems in detail, they know how to implement and evaluate the underlying algorithms and data structures, and are able to develop them further under different constraints.

Teaching and Learning Methods:
Lecture, tutorial, problems for individual study

Media:
Lecture with animated slides

Reading List:
- Alfons Kemper, André Eickler: Datenbanksysteme. Eine Einführung. 10., aktualisierte und erweiterte Auflage, Oldenbourg Verlag, 2015
- A. Kemper, M. Wimmer: Übungsbuch: Datenbanksysteme. 3. Auflage Oldenbourg Verlag, 2012
- A. Silberschatz, H. F. Korth, S. Sudarshan: Database System Concepts. Sixth Edition, McGraw-Hill, 2010
- T. Härder, E. Rahm: Datenbanksysteme - Konzepte und Techniken der Implementierung. 2. Auflage, Springer Verlag, 2001
- J. Gray, A. Reuter: Transaction Processing: Concepts and Techniques. Morgan Kaufmann, 1993

Responsible for Module:
Kemper, Alfons; Prof. Dr.

Courses (Type of course, Weekly hours per semester), Instructor:
Übungen zu Einsatz und Realisierung von Datenbanksystemen (IN2031) (Übung, 2 SWS)
Kemper A [L], Kemper A, Götz T, Reif M, Rey A

Einsatz und Realisierung von Datenbanksystemen (IN2031) (Vorlesung, 3 SWS)
Kemper A [L], Kemper A, Götz T, Reif M, Rey A
For further information in this module, please click campus.tum.de or here.

Module Description

IN2040: Virtual Machines | Virtuelle Maschinen

Version of module description: Gültig ab winterterm 2011/12

Module Level: Bachelor/Master	**Language:** English	**Duration:** one semester	**Frequency:** summer semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 105	**Contact Hours:** 75

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The assessment is by means of a written exam of 90 minutes. Individual assignments ask to apply the learnt translation schemes to small example programs. By that, the exam assesses how well the student is acquainted with various programming constructs and whether she or he is able to translate these into machine code. Further assignments reflect on the concept of virtual machines itself by proposing extra language concepts for which translation schemes should be provided. The successful completion of homework asignments may contribute to the grade as a bonus. The exact details for this are announced timely at the begin of the lecture.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:
At least rudimentary knowledge of different programming languages.

Content:
While trying to produce code for a programming language like Prolog, one quickly realizes that one would like to use certain instructions during the translation which are not already available on concrete machines. On the other hand, instruction sets of modern computers are changing so quickly that it doesn't seem useful for the compiler to depend on some arbitrarily chosen instructions. Such a dependance would mean that in a few years one would feel obliged to rewrite the compiler anew.

With the implementation of the first Pascal compilers, one already arrived at the idea of first generating code for a slightly idealized machine, each of whose instructions then only need to be implemented on different target machines. Translation of modern programming languages like Prolog, Haskell or Java are also based on this principle. On one hand this facilitates portability of the compiler. On the other hand this also simplifies the translation itself since one can choose a

suitable instruction set according to the programming language to be translated. In particular, we consider:
- the translation of C;
- the translation of a functional language;
- the translation of Prolog;
- the translation of a concurrent dialect of C.

Intended Learning Outcomes:
Participants are acquainted with virtual machines for imperative, functional, logical and object-oriented programming languages. They know the principles by which various programming language concepts are translated into sequences of machine code. For sections of programs, they are able to generate code of some virtual machine, and they are able to apply the learnt principles to provide new translation schemes for given language constructs on their own.

Teaching and Learning Methods:
By means of a presentation, either by slides or whiteboard, the lecture presents schemata for the translation of various language constructs and illustrates these by means of small examples. Accompanying assignments for individual study deepen the understanding of the concepts explained in the lecture, and train students to apply the learnt schemata for the translation and to develop new schemata for selected language constructs.

Media:
Slide show, blackboard, possibly online programming and/or animations

Reading List:
Seidl, wilhelm: Compiler Design. Virtual Machines. Springer, 2010

Responsible for Module:
Seidl, Helmut; Prof. Dr.

Courses (Type of course, Weekly hours per semester), Instructor:
Virtuelle Maschinen (IN2040) (Vorlesung mit integrierten Übungen, 5 SWS)
Kocal A, Petter M, Seidl H
For further information in this module, please click campus.tum.de or here.

Module Description

IN2067: Robotics | Robotics

Version of module description: Gültig ab summerterm 2022

Module Level: Master	**Language:** English	**Duration:** one semester	**Frequency:** winter semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 105	**Contact Hours:** 75

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
In a 90-minute written exam, participants must create a mathematical model of a kinematic chain of a given manipulator, determine the relationship between the required forces and torques in the actuator and the dynamic state of the robot, and design a stable PID controller for an exemplary task design that is described in the problem.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:
- Vector algebra
- Differential calculus
- Basic knowledge of physics (Newton's Law, etc.)

Content:
On the one hand, this module presents a method how a mechanical system can be converted into a mathematical system of equations for analysis of motion. In a second step, the parameterization of the control of a manipulator is derived from the mathematical analysis of the above equation in order to compensate for any errors and deviations along a given trajectory.

The following topics are defined in the lecture and discussed using practical examples:
- Coordinate systems (Denavit-Hartenberg convention)
- Forward kinematics (relationship: joint rotation to manipulator motion)
- Inverse kinematics (relationship: manipulator motion to joint rotations)
- Newton-Euler/Lagrange analysis of the dynamic state in the joints
- Dynamic modeling of the manipulator (mathematical model (MVG) for motion analysis)
- PID control of position and force

Intended Learning Outcomes:
The participants should be able to create a mathematical model of a mechanical system using force/torque analysis (Newton-Euler approach) or by energy analysis (Lagrange method), which relates the drive torques in the joints to motion parameters of the manipulator.
They should also be able to explain the meaning and mathematical relationship to the above model for the control parameters of a PID controller for a robotic system and determine their optimal values for a position and force controller.

Teaching and Learning Methods:
The course content is presented to the students in a lecture and deepened in an interactive discussion. There are also recorded lectures from previous years that can be used for self-study.
The individual learning is supported by tutorials, which are to be solved independently by the students and then their solutions are presented in a 2-hour exercise.
Practical examples from the industry on the presented topics will also be shown and guest lectures from the industry will be organised.

Media:
Blackboard, slides, videos and online examples

Reading List:
Introduction to Robotics Mechanics and Control John
J, Craig, Prentice Hall. ISBN 0-13-123629-6

Responsible for Module:
Burschka, Darius; Prof. Dr.-Ing.

Courses (Type of course, Weekly hours per semester), Instructor:
Robotik (IN2067) (Vorlesung mit integrierten Übungen, 5 SWS)
Burschka D
For further information in this module, please click campus.tum.de or here.

Module Description

IN2073: Cloud Computing | Cloud Computing

Version of module description: Gültig ab summerterm 2024

Module Level: Bachelor/Master	**Language:** English	**Duration:** one semester	**Frequency:** summer semester
Credits:* 4	**Total Hours:** 120	**Self-study Hours:** 75	**Contact Hours:** 45

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The exam takes the form of an written 60 minutes test without aids. Questions allow to asses acquaintance with the concepts of Cloud and Grid Computing. Questions describing usage scenarios and asking for the evaluation of the learned techniques in these scenarios are used to assess the ability to apply the learned techniques. In a discussion, their ability to solve research question is assessed.

Repeat Examination:
Next semester

(Recommended) Prerequisites:
Knowledge in computer architectures and distributed systems would be helpful.

Content:
The lecture starts with an introduction and a presentation of the base technologies for Cloud and Grid computing. The layered architecture of Grids and the base services are presented. Cloud Computing is then introduced and the different models SaaS, PaaS, IaaS. The list of base services is extended for Cloud Computing. The lecture also covers a discussion of legal issues.

Intended Learning Outcomes:
The students know the goals of Cloud and Grid computing. They can present application scenarios in different domains. They are familiar with the fundamental techniques in the areas security, application development and resource management. They can identify the differences and similarities between Cloud and Grid computing and distributed systems. They are able to participate in Cloud and Grid-related research projects.

Teaching and Learning Methods:
The concepts of Grid and Cloud Computing are introduced in the lecture. In the exercises, the student work on assignments that allow them to train the development of Cloud applications. References to current literature allow the students to deepen their understanding of the concepts.

Media:
Slides, Script, Exercise Sheets, Prepared Code Snippets.

Reading List:
- Berman, F., Fox, G., Hey, A. (ed.): Grid Computing-Making the Global Infrastructure a Reality, Wiley, Chichester 2003 (collection of 43 contributions, Grids and applications)
- Di Martino et.al. Engineering the Grid, American Scientific Publishers, 2004, (collection of 34 contributions to application and technology of grids)
- Furht, B., Escalante, A.: Handbook of Cloud Computing, Springer 2010
- Chorafas, D.: Cloud Computing Strategies, CRC Press 2011

Responsible for Module:
Gerndt, Hans Michael; Prof. Dr.

Courses (Type of course, Weekly hours per semester), Instructor:
Cloud Computing (IN2073) (Vorlesung, 2 SWS)
Gerndt H, Nunez Araya I

Übung zu Cloud Computing (IN2073) (Übung, 1 SWS)
Gerndt H, Nunez Araya I
For further information in this module, please click campus.tum.de or here.

Module Description

IN2076: Advanced Computer Architecture | Advanced Computer Architecture

Version of module description: Gültig ab summerterm 2024

Module Level: Bachelor/Master	**Language:** English	**Duration:** one semester	**Frequency:** winter semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 120	**Contact Hours:** 60

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The exam takes the form of written 90 minutes examwithout aids. Questions allow to assess acquaintance with the concepts of Computer Architecture. Questions describing scenarios for the interaction of programs with certain architectures will assess the student's ability to evaluate architectural components and to apply the obtained knowledge.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:
IN0004 Introduction to Computer Organization and Technology - Computer Architecture

Content:
After an introduction to the goals and the learning outcomes of the module, cross cutting aspects for all advanced architectures are presented. This section covers performance, availability, reliability, fault tolerance, parallelism, memory hierarchy and virtualization. After a recap of the computer architecture basics, the module covers the major types of parallelism and the respective architectures. For instruction level parallelism advanced concepts of the instruction pipeline are discussed as well as superscalar and VLIW processors. This part also covers advanced techniques for the memory hierarchy and compiler support for instruction level parallelism. The next architecture class, data parallel systems, covers vector units in standard processors, array computers, GPGPUs and vector supercomputers. The section presents also the programming interfaces and discusses their interaction with the architectures. Shared memory systems supporting thread level parallelism are discussed next. First the general concepts coherence, memory consistency and synchronization are covered. Then their implementation in uniform and non-uniform memory architectures is presented, ranging from standard multicore systems to large-scale shared memory systems. The last presented architecture class covers distributed

memory systems supporting process-level parallelism. This section presents high performance communication networks and design alternatives for network interfaces, manycore processors and massively parallel systems. Parallel file systems are discussed as they are important for all these systems. The module closes with optional presentations about energy efficiency, parallel applications, parallel programming, performance evaluation and non-conventional architectures.

Intended Learning Outcomes:
At the end of the module students know and understand the architecture of current processors as well as of entire IT systems. They can evaluate and assess different designs. The students understand the interaction of architecture and compiler technology. They understand the different classes of parallel architectures and can evaluate their advantages and disadvantages for certain applications.

Teaching and Learning Methods:
The module consists of a four hour lecture. The students need 90 hours to learn the presented concepts, and to understand and extend the presented examples. They need to come up with own examples to deepen their knowledge and should compare the learned concepts with presentations in the recommended text books.

Media:
Slides, mindmaps, script

Reading List:
- Hennessy, Patterson: Computer Architecture - A quantitative Approach.
- Andrew Tanenbaum: Structured Computer Organization
- David E. Culler et.al.: Parallel Computer Architecture: A Hardware / Software Approach
- Antonio Gonzales et.al.: Processor Microarchitecture: An Implementation Perspective

Responsible for Module:
Gerndt, Hans Michael; Prof. Dr.

Courses (Type of course, Weekly hours per semester), Instructor:
Rechnerarchitektur (IN2076) (Vorlesung, 4 SWS)
Gerndt H [L], Gerndt H
For further information in this module, please click campus.tum.de or here.

Module Description

IN2101: Network Security | Network Security

Version of module description: Gültig ab winterterm 2011/12

Module Level: Bachelor/Master	**Language:** English	**Duration:** one semester	**Frequency:** winter semester
Credits:* 5	**Total Hours:** 150	**Self-study Hours:** 90	**Contact Hours:** 60

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The examination will take the form of a 75-minute examination.
Questions of comprehension and arithmetic tasks check the familiarity with the technologies and methods of cryptographic procedures and protocols and mechanisms for network security covered in the module.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:
IN0009 Basic Principles: Operating Systems and System Software, IN0010 Introduction to computer networking and distributed systems

Content:
The course provides an introduction to the field of network security. Starting with possible threats and attack scenarios, requirements for providing specific security services are derived. After introducing the basic concepts of security mechanisms, the integration of security mechanisms into network architectures and network protocols are discussed. Security vulnerabilities of existing network architectures are also discussed.
As a basis for the realization of security mechanisms, cryptographic algorithms (in particular symmetric cryptography, public key cryptography and cryptographic hash functions) are presented. Afterwards, the basics and methods for security protocols for authentication, authorization, access control, message integrity, confidentiality and non-repudiation are discussed. Subsequent sections present specific security mechanisms, in particular of the TCP/IP protocol family. The standard examples include PKI, Kerberos, IPSec, and TLS, Firewall-architectures and Intrusion Detection Systems.

Intended Learning Outcomes:
Participants understand security goals for the Internet and the components in which communication protocols are implemented. They understand the possibilities available to attackers in the network. They understand the protection offered by cryptographic and network security mechanisms, and have the knowledge to apply network security protocols and implement architectures that can achieve specific security goals.

Teaching and Learning Methods:
Lecture for content transfer, as well as tasks for self-study in order to deepen the subject, as well as programming challenges to test and apply the learned knowledge.

Media:
Lecture slides, whiteboard, exercise sheets, demos

Reading List:
- R. Bless, S. Mink, E.-O. Blaß, M. Conrad, H.-J. Hof, K. Kutzner, M. Schöller: "Sichere Netzwerkkommunikation", Springer, 2005, ISBN: 3-540-21845-9
- Niels Ferguson, B. Schneier: ?Practical Cryptography?, Wiley, 1st edition, March 2003.
- G. Schäfer. Netzsicherheit ? Algorithmische Grundlagen und Protokolle. Soft cover, 422 pages, dpunkt.verlag, 2003.

Additional references to articles and other resources are given in the slides.

Responsible for Module:
Carle, Georg; Prof. Dr.-Ing.

Courses (Type of course, Weekly hours per semester), Instructor:
Netzsicherheit (IN2101) (Vorlesung mit integrierten Übungen, 4 SWS)
Carle G [L], Carle G, Kinkelin H, von Seck R, Rezabek F, Kempf M, Sattler P, Steger L
For further information in this module, please click campus.tum.de or here.

Module Description

IN2222: Cognitive Systems | Cognitive Systems

Version of module description: Gültig ab winterterm 2011/12

Module Level: Master	**Language:** English	**Duration:** one semester	**Frequency:** summer semester
Credits:* 5	**Total Hours:** 150	**Self-study Hours:** 90	**Contact Hours:** 60

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The exam takes the form of a 75 minutes written test. Questions allow to assess acquaintance with basic concepts of the perception - cognition - action loop in biological as well as in technical cognitive systems.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:

Content:
Basic concepts (perception - cognition - action), transfer of cognitive skills (sensors, algorithms, behaviour patterns), study of prototypes of existing systems available at TUM.

Intended Learning Outcomes:
Upon successful completion of the module, students understand fundamental principles of human cognition and are able to transfer to technical systems. They are able to apply these mathematical and algorithmic methods to obtain cognitive qualities in technical systems, especially in the area of robotics. These methods include architectures for cognitive systems, knowledge representation and reasoning, handling of uncertain knowledge, perception, planning, and action execution.

Teaching and Learning Methods:
Basic terms and concepts of the subject area are introduced in the lecture with presentation slides and discussed on the basis of examples. During the exercise, the acquired knowledge will be deepened by studying and discussing the contents of conference papers and professional articles. In addition, relevant tools for the implementation of the concepts presented will be presented interactively and applied to typical tasks.

Media:
Slide show, blackboard, online programming experiments, animations

Reading List:
- VERNON, David. Artificial cognitive systems: A primer. MIT Press, 2014.
- WILSON, Robert Andrew; KEIL, Frank C. (Hg.). The MIT encyclopedia of the cognitive sciences. MIT press, 2001.
- GERSTNER, Wulfram; KISTLER, Werner M. Spiking neuron models: Single neurons, populations, plasticity. Cambridge University Press, 2002.

Responsible for Module:
Knoll, Alois Christian; Prof. Dr.-Ing. habil.

Courses (Type of course, Weekly hours per semester), Instructor:
Kognitive Systeme (IN2222) (Vorlesung mit integrierten Übungen, 4 SWS)
Knoll A [L], Bing Z, Knoll A, Petrovic N, Purschke N, Schamschurko A
For further information in this module, please click campus.tum.de or here.

Module Description

IN2309: Advanced Topics of Software Engineering | Advanced Topics of Software Engineering

Version of module description: Gültig ab winterterm 2014/15

Module Level: Master	**Language:** German/English	**Duration:** one semester	**Frequency:** winter semester
Credits:* 8	**Total Hours:** 240	**Self-study Hours:** 150	**Contact Hours:** 90

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
In the written exam (100 minutes) students should prove to be able to identify a given problem and find solutions within limited time. The examination will completely cover the content of the lectures. The anwers will require own formulations. In addition, some modeling may be required.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:
Introduction to Software Engineering (IN0006)

Content:
Developing large software systems necessitates tightly intertwining and synchronizing both the activities in the development process and the generated artifacts. Central success criteria include
- Iterative elicitation and implementation of the requirements;
- Architecture as the backbone of development; and
- Mastering quality in its different facets.
This class on Advanced Software Engineering demonstrates how requirements engineering, specification, architecture, detailed design, coding, software quality, and quality assurance interweave. Specifically, we discuss how non-functional requirements are reflected in the architecture; which tradeoffs exist with functional requirements as well as external and internal quality attributes; and how these requirements can be formulated and verified by tests.
We will consider:
1. Requirements Engineering: Techniques for elicitation, analysis, prioritization, specification, validation of functional and non-functional requirements; anti-requirements.

2. Software Architecture: Principles, views&styles, architecture documentation, patterns, frameworks, reference architectures, product lines, analysis of an architecture s quality and of trade-offs.
3. Software Quality: Internal and external quality attributes, among others: maintainability, testability, understandability and performance, security, availability; software metrics
4. Quality Assurance: Assessment, prioritization, conflict resolution and reviews of requirements; assessments of architectures w.r.t. internal and external quality attributes; tests and reviews for functional and non-functional requirements in the code; fault models.
5. Influence of the development process.

Intended Learning Outcomes:
At the end of the class, students understand the central goals and methods of requirements engineering. They know all relevant software quality attributes. They know how to elicit, specify and manage respective requirements; how they are reflected in architectures and, conversely, how architecture influences these attributes; how requirements, architectures and code can be assessed w.r.t. these attributes; and what the influence of the development process is. They can apply this knowledge in smaller projects in practice.

Teaching and Learning Methods:
This module comprises lectures and accompanying tutorials. The contents of the lectures will be taught by talks and presentations. Students will be encouraged to study literature and to get involved with the topics in depth. In the tutorials, concrete problems will be solved - partially in teamwork - and selected examples will be discussed.

Media:
Lecture with slides

Reading List:
McConnell, Code Complete: A Practical Handbook of Software Construction, 2nd edition, Microsoft, 2004
Summerville, Software Engineering 9, Prentice Hall, 2010
Brooks, The Mythical Man Month, Addison-Wesley Longman, 1995
Rombach, Endres: A Handbook of Software and Systems Engineering, Addispn Wesley, 2003
Bass et al., Software Architecture in Practice, Addison Wesley, 3rd edition, Addison Wesley, 2012
Clements et al., Documenting Software Architectures, 2nd edition, Addison Wesley, 2010
Clements et al., Evaluating Software Architectures, Addison Wesley, 2001
Reussner, Hasselbring, Handbuch der Software-Architektur (in German), 2006
Jackson, Problem Frames, ACM Press, 2000
Sommerville, Sawyer: Requirements Engineering: A Good Practice Guide, John Wiley, 1997
Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, John Wiley, 2009
Goucher, Riley, Beautiful Testing, OReilly, 2009
Wagner, Software Product Quality Control, Springer, 2013

More references will be provided in class.

Responsible for Module:
Pretschner, Alexander; Prof. Dr.

Courses (Type of course, Weekly hours per semester), Instructor:
Zentralübung zu Advanced Topics of Software Engineering (IN2309, IN2126) (Übung, 2 SWS)
Weber I [L], Stiehle F, Weber I

Advanced Topics of Software Engineering (IN2309, IN2126) (Vorlesung, 4 SWS)
Weber I [L], Stiehle F, Weber I
For further information in this module, please click campus.tum.de or here.

Module Description

IN2346: Introduction to Deep Learning | Introduction to Deep Learning

Version of module description: Gültig ab summerterm 2018

Module Level: Master	**Language:** English	**Duration:** one semester	**Frequency:** summer semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 120	**Contact Hours:** 60

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
- Written test of 90 minutes at the end of the course.
- After each practical session, the students will have to provide the written working code to the teaching assistant for evaluation. The students will be awarded a bonus in case they successfully complete all practical assignments.

The exam takes the form of a written test. Questions allow to assess acquaintance with the basic concepts and algorithms of deep learning concepts, in particular how to train neural networks. Students demonstrate the ability to design, train, and optimize neural network architectures, and how to apply the learning frameworks to real-world problems (e.g., in computer vision). An important aspect for the student is to understand the basic theory behind the training process, which is mainly coupled with optimization strategies involving backprop and SGD. Students can use networks in order to solve classification and regression tasks (partly motivated by visual data).

Repeat Examination:
Next semester

(Recommended) Prerequisites:
Programming knowledge is expected. At least one programming language should be known, preferably Python.

MA0902 Analysis for Informatics
MA0901 Linear Algebra for Informatics

Content:
- Introduction to the history of Deep Learning and its applications.
- Machine learning basics 1: linear classification, maximum likelihood
- Machine learning basics 2: logistic regression, perceptron

- Introduction to neural networks and their optimization
- Stochastic Gradient Descent (SGD) and Back-propagation
- Training Neural Networks Part 1:
regularization, activation functions, weight initialization, gradient flow, batch normalization, hyperparameter optimization
- Training Neural Networks Part 2: parameter updates, ensembles, dropout
- Convolutional Neural Networks, ConvLayers, Pooling, etc.
- Applications of CNNs: e.g., object detection (from MNIST to ImageNet), visualizing CNN (DeepDream)
- Overview and introduction to Recurrent networks and LSTMs
- Recent developments in deep learning in the community
- Overview of research and introduction to advanced deep learning lectures.

Intended Learning Outcomes:
Upon completion of this module, students will have acquired theoretical concepts behind neural networks, and in particular Convolutional Neural Networks, as well as experience on solving practical real-world problems with deep learning. They will be able to solve tasks such as digit recognition or image classification.

Teaching and Learning Methods:
The lectures will provide extensive theoretical aspects of neural networks and in particular deep learning architectures; e.g., used in the field of Computer Vision.
The practical sessions will be key, students shall get familiar with Deep Learning through hours of training and testing. They will get familiar with frameworks like PyTorch, so that by the end of the course they are capable of solving practical real-world problems with Deep Learning.

Media:
Projector, blackboard, PC

Reading List:
- Slides given during the course
- www.deeplearningbook.org

Responsible for Module:
Nießner, Matthias; Prof. Dr.-Ing.

Courses (Type of course, Weekly hours per semester), Instructor:
Introduction to Deep Learning (IN2346) (Vorlesung mit integrierten Übungen, 4 SWS)
Cremers D [L], Cremers D, Gladkova M (Chui P), Hofherr F (Qian S, Xia Y)
For further information in this module, please click campus.tum.de or here.

Module Description

IN2359: Blockchain-based Systems Engineering | Blockchain-based Systems Engineering

Version of module description: Gültig ab summerterm 2019

Module Level: Master	**Language:** English	**Duration:** one semester	**Frequency:** summer semester
Credits:* 5	**Total Hours:** 150	**Self-study Hours:** 90	**Contact Hours:** 60

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
Type: Written exam
The assessment is by means of a written exam of 90 minutes. Students are required to be able to answer questions regarding the contents of the lecture without further aids. Answering the questions requires partly own formulations, partly own calculations.
Amount of Work:
Comprehensive knowledge about the contents of the lecture and the exercises has to be gained. The completion of voluntary homework assignments is recommended for the successful passing.

Repeat Examination:

(Recommended) Prerequisites:
• IN0002: Fundamentals of Programming
• IN0006: Introduction to Software Engineering
• IN0009: Basic Principles: Operating Systems and System Software

Content:
Blockchain technology and, in general, distributed ledger technology (DLT) provide the technical foundation for the development and usage of innovative, decentralized distributed systems. In this lecture, we analyze the characteristics of these technologies. Additionally, students should be empowered to analyze and develop Blockchain-based solutions. Following contents are going to be covered:
• Cryptographic basics
• Peer to peer-networks
• Data structure and setup of Blockchain
• Consensus mechanisms

• Smart contracts & smart contract Engineering
• Use cases of digital ledger technologies
• Alternative DLT approaches
• Risks, challenges, and limitations of the technology
• Trends and developments in Blockchain

Intended Learning Outcomes:
After the successful participation in this module, the students are able to analyze Blockchain-based application systems. Further, they are able to create these systems for given use cases and to select appropriate technology. They understand the technological foundations such that they are enabled to comprehend and assess alternative distributed ledger technologies.

Teaching and Learning Methods:
Lecture, central exercise

Media:
Presentation with digital slides

Reading List:
Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press

Responsible for Module:
Matthes, Florian; Prof. Dr. rer. nat.

Courses (Type of course, Weekly hours per semester), Instructor:
Blockchain-based Systems Engineering (IN2359) (Vorlesung mit integrierten Übungen, 4 SWS)
Matthes F [L], Matthes F, Gebele J, Öz B
For further information in this module, please click campus.tum.de or here.

Module Description

IN2406: Fundamentals of Artificial Intelligence | Fundamentals of Artificial Intelligence

Version of module description: Gültig ab summerterm 2024

Module Level: Bachelor/Master	**Language:** English	**Duration:** one semester	**Frequency:** winter semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 105	**Contact Hours:** 75

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
Written exam at the end of the semester lasting 90min. The questions will cover most of the learned material and are typically shorter than the problems solved in the exercise, but similar in difficulty. All questions will be aligned with the learning outcomes. The correct answers to these questions may involve mathematical derivations and calculations. As an incentive to create artificial intelligence oneself, we provide programming challenges: if students solve a required number of programming challenges, they obtain a 0.3 grade bonus for their exam. During the exam, students are allowed to use a pen (color should not be red or green and pencils are not allowed), a non-programmable calculator, and a formula sheet. Whether this formula sheet is provided or can be written by the students will be announced before each exam.

Repeat Examination:
End of Semester

(Recommended) Prerequisites:
Previous attendance of
- IN0007 Fundamentals of Algorithms and Data Structures
- IN0015 Discrete Structures
- IN0018 Discrete Probability Theory
is beneficial. However, all content is taught from ground up and the listed lectures are not essential. Students who have not attended these lectures will have to invest additional time.

Content:
- Task environments and the structure of intelligent agents.
- Solving problems by searching: breadth-first search, uniform-cost search, depth-first search, depth-limited search, iterative deepening search, greedy best-first search, A* search.

- Constraint satisfaction problems: defining constraint satisfaction problems, backtracking search for constraint satisfaction problems, heuristics for backtracking search, interleaving search and inference, the structure of constraint satisfaction problems.
- Logical agents: propositional logic, propositional theorem proving, syntax and semantics of first-order logic, using first-order logic, knowledge engineering in first-order logic, reducing first-order inference to propositional inference, unification and lifting, forward chaining, backward chaining, resolution.
- Bayesian networks: acting under uncertainty, basics of probability theory, Bayesian networks, inference in Bayesian networks, approximate inference in Bayesian networks.
- Hidden Markov models: time and uncertainty, inference in hidden Markov models (filtering, prediction, smoothing, most likely explanation), approximate inference in hidden Markov models.
- Rational decisions: introduction to utility theory, utility functions, decision networks, the value of information, Markov decision processes, value iteration, policy iteration, partially observable Markov decision processes.
- Learning: types of learning, supervised learning, learning decision trees, reinforcement learning.
- Introduction to robotics: robot hardware, robotic perception, path planning, planning uncertain movements, control of movements, application domains.

Intended Learning Outcomes:
After attending the module, you are able to create artificial intelligence on a basic level using search techniques, logics, probability theory and decision theory. Your learned abilities will be the foundation for more advanced topics in artificial intelligence. In particular, you will acquire the following skills:
- You can analyze problems of artificial intelligence and judge how difficult it is to solve them.
- You can recall the basic concepts of intelligent agents and know possible task environments.
- You can formalize, apply, and understand search problems.
- You understand the difference between constraint satisfaction and classical search problems as well as apply and evaluate various constraint satisfaction approaches.
- You can critically assess the advantages and disadvantages of logics in artificial intelligence.
- You can formalize problems using propositional and first-order logic.
- You can apply automatic reasoning techniques in propositional and first-order logic.
- You understand the advantages and disadvantages of probabilistic and logic-based reasoning.
- You can apply and critically asses methods for probabilistic reasoning with Bayesian networks and Hidden Markov Models.
- You understand and know how to compute rational decisions.
- You have a basic understanding on how a machine learns.
- You know the basic areas and concepts in robotics.

Teaching and Learning Methods:
The module consists of a lecture and exercise classes. The content of the lecture is presented via slides, which are completed during the lecture using the blackboard and/or an electronic writing pad. Students are encouraged to additionally study the relevant literature. In the exercise classes, the learned content is applied to practical examples to consolidate the content of the lecture. Students should ideally have tried to solve the problems before they attend the exercise. To

encourage more participation, students are regularly asked questions or encouraged to participate in online polls. As an incentive to create artificial intelligence oneself, we provide programming challenges: if students solve a required number of programming challenges, they obtain a 0.3 grade bonus for their exam.

Media:
Slides, blackboard, electronic writing pad, exercise sheets;

Reading List:
- P. Norvig and S. Russell: Artificial Intelligence: A Modern Approach,
Prentice Hall, 4th edition. (English version)
- P. Norvig and S. Russell: Künstliche Intelligenz: Ein moderner Ansatz,
Pearson Studium, 4. Auflage. (German version)
- W. Ertel: Grundkurs Künstliche Intelligenz: Eine praxisorientierte
Einführung, Springer, 4. Auflage.
- P. Zöller-Greer: Künstliche Intelligenz: Grundlagen und
Anwendungen, composia, 2. Auflage.
- D. L. Poole and A. K. Mackworth: Artificial Intelligence: Foundations of
Computational Agents, Cambridge University Press.
- P. C. Jackson Jr: Introduction to Artificial Intelligence, Dover
Publications.

Responsible for Module:
Althoff, Matthias; Prof. Dr.-Ing.

Courses (Type of course, Weekly hours per semester), Instructor:
Fundamentals of Artificial Intelligence (IN2406) (Vorlesung mit integrierten Übungen, 5 SWS)
Althoff M [L], Althoff M, Lercher F, Lützow L, Mair S, Thumm J
For further information in this module, please click campus.tum.de or here.

Module Description

IN8030: Informatics Advanced Practical Courses for Management | Informatik Master-Praktika für Management

Version of module description: Gültig ab winterterm 2019/20

Module Level: Master	**Language:** German/English	**Duration:** one semester	**Frequency:** winter/summer semester
Credits:* 10	**Total Hours:** 300	**Self-study Hours:** 210	**Contact Hours:** 90

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
Type of Assessment: project work

Different phases of a software project (especially definition, design, development, implementation, documentation, test) along a specific application in an advanced area of expertise are worked on by the participants in teams. Single teams may work only on a subset of all the phases. Current state of the art application specific methods and systems are applied. The obtained results are documented in written form and orally presented, if applicable. It is announced in advance how the single activities will be weighted to calculate the module grade.

Repeat Examination:
Next semester

(Recommended) Prerequisites:
Basic knowledge of the specific subject area may be necessary (elective course).

Content:
- Implementation of a software application or subapplication in teams
- Application specific methods and systems according to the current state of the art
- Techniques for documentation and presentation of results or intermediate results in application development

This module is offered by many lecturers and deals with different contents according to their focus in research and teaching (e.g., data bases, compiler construction, information systems, networks, group ware, graphics, robotics, image processing).

Intended Learning Outcomes:
Participants command the development of an application in an advanced area of expertise, coming from the area of the respective chair (e.g., data bases, information systems, networks, group ware, graphics, robotics, image processing) using a methodologically clean approach. They are able to use application specific methods and systems that meet the current state of the technology. In teams they work in a goal oriented way. The participants have the competence to document their approach and present the results.

Teaching and Learning Methods:
The participants apply rigorous software engineering in an advanced project in small teams according to a specificattion and with tight schedule (design, implementation, test). Partial results of the team work are communicated in presentations. The individual phases of system building are to be documented.

Media:
Beamer, slides, whitebord, platform for collaborative work, software development environment, application specific tools

Reading List:
To be announced by the lecturers, domain specific

Responsible for Module:
Kemper, Alfons; Prof. Dr.

Courses (Type of course, Weekly hours per semester), Instructor:
Masterpraktikum - Digital Product Innovation and Development (IN2106, IN4381) (Praktikum, 6 SWS)
Stocco A

Master-Praktikum - Enterprise Software Engineering am Beispiel von SAP (IN2128, IN2106, IN212801) (Praktikum, 6 SWS)
Wittges H [L], Blank N, Fleischle A, Hafner A, Landler P

Master-Praktikum - Entwicklung innovativer Services am Beispiel von SAP Technologien (IN2128, IN2106, IN212802) (Praktikum, 6 SWS)
Wittges H [L], Wittges H, Haug K, Wolf N, Eckhard F
For further information in this module, please click campus.tum.de or here.

Module Description

IN8031: Informatics Advanced Seminar Courses for Management | Informatik Master-Seminare für Management

Version of module description: Gültig ab winterterm 2019/20

Module Level: Master	**Language:** German/English	**Duration:** one semester	**Frequency:** winter/summer semester
Credits:* 5	**Total Hours:** 150	**Self-study Hours:** 120	**Contact Hours:** 30

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
Participants have the necessary methodological and interdisciplinary skills to independently write a scientific essay about an advanced topic in computer science, as well as to present and discuss its content. The students can work with scientific literature (i.e. search, categorize, prioritize, cite, ...). They master the required presentation and discussion techniques.

Repeat Examination:
Next semester

(Recommended) Prerequisites:
Basic knowledge of the specific subject area may be necessary (elective course).

Content:
- Independent assessment of a scientifically demanding theme
- Preparation of a term paper with a section on related work
- Presentation and discussion of scientific results

This module is offered by all faculty members. They select suitable topics from their research area and support the students in learning the technical and scientific skills.

Intended Learning Outcomes:
The participants acquire the necessary methodological and interdisciplinary skills for dealing with a scientifically demanding topic in Informatics. They know how to produce and present a scientific seminar paper. They can analyze and evaluate the results by contrasting them. They can work with scientific literature (search, evaluation, citation) and are able to present the constituent aspects of their topic in both a written paper and a presentation in front of the group.

Teaching and Learning Methods:
Participants independently assess a demanding scientific topic, prepare a presentation and discuss their findings.
The accompanying term paper summarizes the main concepts of the subject and provides an overview over related work.

Media:
Beamer, slides, whiteboard, report, possibly tool presentation and/or animations

Reading List:
Scientific publications in the respective area

Responsible for Module:
Kemper, Alfons; Prof. Dr.

Courses (Type of course, Weekly hours per semester), Instructor:
Bachelor/Master-Seminar - Behavioral Insights in the Age of Big Data (IN0014, IN2107, IN2396, IN4424) (Seminar, 2 SWS)
Großklags J [L], Chen M

Bachelor/Master-Seminar - Data Analytics for Cybercrime and Undesirable Online Behaviors (IN0014, IN2107, IN2396, IN4896) (Seminar, 2 SWS)
Großklags J [L], Großklags J

Bachelor/Master-Seminar - Security and Privacy Economics (IN0014, IN2107, IN2396, IN4892) (Seminar, 2 SWS)
Großklags J [L], Großklags J, Syrmoudis E

Seminar - Interdisciplinary Research Seminar on Generative AI in Sub-Saharan Africa (IN0014, IN2107, IN2396, IN45081) (Seminar, 2 SWS)
Großklags J [L], Onyekwelu B

Master-Seminar - Digital Transformation (IN2107, IN2396, IN4831) (Seminar, 2 SWS)
Krcmar H [L], Böttcher T, Kernstock P, Oswald G

Master-Seminar - Digital Transformation & Sustainability (IN2107, IN4426) (Seminar, 2 SWS)
Krcmar H [L], Böttcher T, Weber M

Master-Seminar - Digitale Transformation der Verwaltung - Chancen für Bürger, Unternehmen, Verwaltung und Politik durch moderne Informationstechnik (IN2107, IN4972) (Seminar, 2 SWS)
Krcmar H [L], Denkhaus W, Daßler L, Jäger I

Miniprojekte im strategischen IT Management (IN2107, IN4768) (Seminar, 2 SWS)
Matthes F [L], Matthes F, Tobisch F

Master-Seminar - Scientific Methods in Information Systems (IN2107, IN2396, IN4439) (Seminar, 2 SWS)
Rinderle-Ma S [L], Rinderle-Ma S, Barrientos Moreno M

Master-Seminar: Trustworthy AI for Medicine (IN2107, IN45048) (Seminar, 2 SWS)
Rückert D [L], Hölzl F, Schwethelm K
For further information in this module, please click campus.tum.de or here.

Module Description

SOT86050: Advanced Computational Methods | Advanced Computational Methods [AdvCompMeth]

Version of module description: Gültig ab summerterm 2022

Module Level: Master	**Language:** English	**Duration:** one semester	**Frequency:** winter semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 120	**Contact Hours:** 60

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The class will be graded based on four graded exercises (Übungsleistung). These graded exercises will contribute equally to the final grade. The exercises are take home activities and students can use all available material to solve the tasks. Students must show that they can apply the methods learned in class (see below) to new challenges to successfully master the graded exercises.

Repeat Examination:
Next semester

(Recommended) Prerequisites:
POL20100 Programmierung für Sozialwissenschaftler/-Innen OR POL40300 Computational Methods

Content:
Numpy, Pandas, Data I/O, Data Selection, Data Manipulation, and Aggregation, Visualization; Statistics (correlation; T test, Chi2 test); Data Merging, Cleaning, and Reshaping. Web Scraping; APIs (Twitter, Reddit). Text Analysis: Preprocessing, NLP (POS, NER), Representation (vectorization); Topic Modeling. Network Analysis: Intro, Metrics (density, reciprocity, degrees, distance, connectivity,..), Centrality, simulation (diffusion dynamics). Machine Learning: Intro, Regression, Classification, Unsupervised learning (Clustering, Community Detection).

Intended Learning Outcomes:
After the class, students will be able to apply advanced computational methods to their own research projects. They will be able to collect and manipulate data. The students will then be able to utilize existing packages in the Python programming language to analyze large text corpora and network data. They will also learn how to use standard machine learning methods on their data.

Teaching and Learning Methods:
The classes combine lectures and in-class exercises. During the lecture part, new topics will be presented. In the second part of each class, the new concepts will be implemented and applied with in the Python programming language.

Media:
PowerPoint slides, exercise sheets, example code.

Reading List:
John McLevey (2022), Doing Computational Social Science: A Practical Introduction. SAGE Publications Ltd.

Responsible for Module:
Pfeffer, Jürgen; Prof. Dr. rer. soc. oec.

Courses (Type of course, Weekly hours per semester), Instructor:
[SOT86050] Advanced Computational Methods (Seminar, 4 SWS)
Sargsyan A (Ghawi R, Pfeffer J)
For further information in this module, please click campus.tum.de or here.

Module Description

SOT86069: Analyzing Text Data: From Basics to Advanced Techniques | Analyzing Text Data: From Basics to Advanced Techniques

Version of module description: Gültig ab summerterm 2025

Module Level: Master	**Language:** English	**Duration:** one semester	**Frequency:** summer semester
Credits:* 6	**Total Hours:** 180	**Self-study Hours:** 120	**Contact Hours:** 60

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The module examination consists of four graded exercise submissions, with two of them accompanied by oral presentations.
Each exercise includes a Python notebook and a dataset, requiring students to apply text analysis methods (e.g., topic modeling) on the provided data using the Python programming language. The exercises are take-home activities. Therefore, students can use the course materials to complete the tasks. To ensure a deeper understanding of the applied methods, the second and fourth assignments will include a 10-15 minute oral presentation. During these presentations, students will explain and discuss the methods they used in their work.
The grades are distributed as follows:
1st and 3rd assignments: 20 points each (no oral presentation).
2nd and 4th assignments: 30 points each (including oral presentation).
Through the exercises and presentations, students demonstrate their ability to analyze text data using Python and effectively communicate their approach and findings.

Repeat Examination:
Next semester

(Recommended) Prerequisites:
POL20100 Programmierung für Sozialwissenschaftler/-Innen OR POL40300 Computational Methods.

Content:
The module addresses the following content:
• Text Preprocessing

• Text Exploration and Descriptive Statistics
• Natural Language Preprocessing
• Part of Speech (POS) Tagging
• Named Entity Recognition (NER)
• Text Representation
• Text Classification
• Text Clustering
• Sentiment Analysis
• Topic Modeling
• Text Embeddings and Word Vectors

Intended Learning Outcomes:
Throughout this module, participants will gain a comprehensive understanding of the foundational principles and advanced techniques for analyzing text data. Upon successful completion of this module, students will be proficient in preprocessing textual data, employing various tokenization and representation methods, and applying supervised and unsupervised learning algorithms for text classification and clustering tasks. Additionally, participants will master sentiment analysis techniques, topic modeling, and text summarization methods, enabling them to extract valuable insights from large volumes of text data. Through hands-on programming exercises using Python and real-world datasets, learners will develop practical skills in text analysis, empowering them to apply their knowledge across various domains and contexts.

Teaching and Learning Methods:
The module consists of a lecture with integrated exercises. The course usually consists of a lecture part and an exercise part. During the first part, the respective topics and contents from the field of Text Analysis are presented. In the second part, these are then implemented and applied with in the Python programming language.

Media:
PowerPoint slides, exercise sheets, example code.

Reading List:
Steven Bird, Ewan Klein, and Edward Loper (2009). Natural Language Processing with Python. O'Reilly.

Responsible for Module:
Pfeffer, Jürgen; Prof. Dr. rer. soc. oec.

Courses (Type of course, Weekly hours per semester), Instructor:
(SOT86069) Analyzing Text Data - From Basics to Advanced Techniques (Vorlesung mit integrierten Übungen, 4 SWS)
Ghawi R
For further information in this module, please click campus.tum.de or here.

Module Description

SOT86603: Telling Stories with R and Data Visualizations | Telling Stories with R and Data Visualizations

Version of module description: Gültig ab summerterm 2022

Module Level: Master	**Language:** English	**Duration:** one semester	**Frequency:** winter/summer semester
Credits:* 3	**Total Hours:** 90	**Self-study Hours:** 60	**Contact Hours:** 30

Number of credits may vary according to degree program. Please see Transcript of Records.

Description of Examination Method:
The module examination consists of project work (submission of R code and the dataset) with a final presentation (10-15 minutes). In the first phase, students create their own data set, which must have at least one of the following 3 characteristics: 1) Multiple levels; 2) Original topic, subject or perspective; 3) Impressive scope (e.g., large number of observations or a long time span). Students demonstrate that they can independently present information with visuals and exhibits, and can develop clear and effective charts. Accordingly, students develop a visual story. With this, students will demonstrate that they are able to apply the functionality of ggplot2. Students will present their visual examples and submit the R code (and dataset) to ensure that their analysis is reproducible. With the presentation, students will demonstrate their ability to communicate and explain a measurable phenomenon and also showcase their understanding of the features of effective (easily understandable) graphs.

Repeat Examination:

(Recommended) Prerequisites:
Students need to be able to load datasets using R, and have previous experience analyzing data with R / RStudio. Prior knowledge of issues to related to design, style, and principles of data visualization is not required.

Content:
Topics include: Grammar of graphics; Layers (including facets), themes; Refininement of plots; 3-way cross-tabs; Heatmaps; Visualizing output from statistical models; Coefficients and uncertainty; Predicted probabilities, and interactions.

Intended Learning Outcomes:
After successfully completing the module, students will be able to create effective visualizations of social and political data, make discoveries and communicate new insights. In addition, students can apply different approaches to graphically represent different types of evidence, including macroeconomic data, summaries of statistical models, data from social media, and other types of information.

Teaching and Learning Methods:
The module consists of a seminar. The seminar consists of lectures by the lecturer and programming with rmd or qmd scripts. The practical exercises allow students to test their understanding of the concepts and code snippets taught in the introductory sessions of the module.

Media:
Lecture slides for lectures and R (qmd) scripts for tutorials

Reading List:
A detailed list of both mandatory and recommended readings and online turiorals is posted on the course syllabus. Students are encourage to get a headstart by reviewing: Villanueva, R. A. M., & Chen, Z. J. (2019). ggplot2: elegant graphics for data analysis.

Responsible for Module:
Theocharis, Ioannis; Prof. Dr. phil.

Courses (Type of course, Weekly hours per semester), Instructor:
[SOT86603] Telling Stories with R and Data Visualizations (Seminar, 2 SWS)
Zilinsky J
For further information in this module, please click campus.tum.de or here.