Università
della
Svizzera
italiana

**Faculty
of
Informatics**

Master of Science in Software & Data Engineering

2017/18

Software and Data Engineering

Software plays a pivotal role in almost all aspects of our life, including transportation, communication, economy, and healthcare. We put trust in software to accomplish complex and vital tasks for us, such as managing our finances, sharing our family and friends' memories, diagnosing diseases, flying airplanes or driving cars. The complexity of these tasks, while becoming transparent to us, does not go away: It is distilled into the software our civilization depends on. Indeed, we are already in the era of ultra-large-scale software systems, composed by millions of code components interacting among them. In such a scenario, software cannot be understood without its data and data becomes valuable only thanks to the software analyzing it. In other words, software engineering aims at managing the complexity of software, keeping it under control. Data engineering focuses instead on how to collect, store, and process huge amounts of data, that can be analyzed to gather insights and support decision making activities. The master features courses taught by world's leading researchers of the Software Institute at the USI Faculty of Informatics.

Awarded Degree
Master of Science in Software and Data Engineering.
Application Deadline
April 30th / June 30th depending on the nationality of the applicant.
Tuition fees per semester
Residents CHF 2'000.– / international CHF 4'000.–
Duration
4 semesters (2 years) - 120 ECTS
Scholarships
Fondazione per le Facoltà di Lugano
10 study grants for Faculty of Informatics, covers first year of tuition, renewable according to grade
Contacts/information
www.msde.usi.ch
studyadvisor@usi.ch

Goals and contents

The study programme is compounded of four modules: Software Engineering, Data Engineering, Electives, and Master thesis. The Software Engineering module embraces 36 ECTS and provides students with a deep knowledge of state-of-the-art techniques. Topics related to this module are software design, software quality and testing, software architecture, software performance, and software analytics. The Data Engineering module includes three courses (18 ECTS) teaching students techniques and tools to design and model data (1st semester), to convert data into information (2nd semester), and to transform information into knowledge useful to support decision making activities (3rd semester). The topics studied in the Software and the Data Engineering modules are continuously integrated through the whole course of study. This is done by devoting 18 ECTS to deal with both Software and Data Engineering with project based learning. The Electives module includes 12 ECTS, that the student can freely select from a given list of courses offered at the USI Faculty of Informatics based on his/her personal preference. Finally, the remaining 36 ECTS are dedicated to the MSc thesis. Students will use the 6 ECTS of the 3rd semester to visit the research groups of the Software Institute of the Faculty of Informatics and to prepare a thesis proposal. Then, they will work full time on the thesis in the 4th semester in the research group of their choice.

Language

This programme is entirely held in English. Applicants who are not native English speaker or whose first degree was not taught in English, must supply an internationally recognised certificate to demonstrate a C1 level on the Common European Framework of Reference for language learning (CEFR).

Student profile and admission requirements

Students without a strong background in software engineering may be required to attend some additional courses.

Career opportunities

Data is the new natural resource to be mined and exploited using software. Data analytics software provides actionable insights at the basis of continuous improvement and decision making processes. Such insights can be found by exploring large quantities of data, by asking the right questions and knowing how to reliably and efficiently find the appropriate answers. Students graduating in this Master will be highly specialized software and data engineers, able to fully understand and manage the complexity of modern software systems and of the sea of data surrounding them. Mastering how to effectively use software to deal with the data deluge is a key capability for any organization undergoing digital transformation efforts. Also, the demand for software and data engineers is currently very high and it is expected to grow even more in the near future. Besides the expected high employability in industry, graduated students will also represent the perfect candidate for pursuing a PhD degree at USI, by working in one of the research groups of the Software Institute.

Contacts

USI Università della Svizzera italiana
Study Advisory Service
+41 58 666 4795
studyadvisor@usi.ch

Study programme

First semester		Introduction to Software & Data Engineering	3.0
		Software Design & Modeling	6.0
		Engineering of Domain Specific Languages	3.0
		Programming Styles	3.0
		Design 101	3.0
		Data Design & Modeling	6.0
	Electives 6.0	Software Engineering	6.0
		Advanced Networking	6.0
Second semester		Software Quality & Testing	6.0
		Software Architecture	6.0
		Visual Analytics Atelier	6.0
		Information Modeling & Analysis	6.0
	Electives 6.0	Compilers	6.0
		Information Security	6.0
		Physical Computing	6.0
		Robotics	6.0
Third semester		Software Analytics	6.0
		Software Performance	6.0
		Knowledge Analysis & Management	6.0
		Hacking Project	6.0
		Software & Data Engineering Seminar	6.0
Fourth semester		Master Thesis	30.0

Please be aware that slight changes in the study programme may occur.

First semester

Introduction to Software & Data Engineering

Software lives in a world of data. Data is the new natural resource to be mined and exploited using software. Data analytics software provides actionable insights at the basis of continuous improvement and decision making processes. Such insights can be found by exploring large quantities of data, by asking the right questions and knowing how to reliably and efficiently find the appropriate answers. This course provides an overview of the Master programme in Software and Data engineering, introducing the basic concepts that tie together the rest of the lectures.

Software Design & Modeling

The focus of this course is on object-oriented design. The course starts with an introduction to object-oriented programming. Java is used as the reference programming language. Then, in the first part of the course students will (i) learn how to assess (both manually and automatically) the design quality of object-oriented systems, and (ii) get a deep understanding of design patterns. The course takes a hands-on, learning-by-doing approach with assignments performed on open source systems. For example, students will be required to run a "design critique" of well-known open source systems, identifying design flaws and refactoring them. In the second part of the course, students will apply the acquired expertise in the context of a group project.

Engineering of Domain Specific Languages

Domain-Specific Languages (DSLs) are programming languages that are tailored to a particular application domain. This essential focus of DSLs allows to more concise and understandable programmes, with a tight stakeholder integration, and when done properly, to improved code quality.
The course will feature the current state-of-the-art on both the design and implementation of DSLs. The course will emphasize the different design dimensions of DSLs, the involved language paradigms and features, and the drawbacks with respect to simple API modeling. As a student, you will also learn the difference between internal and external DSLs. In parallel, you will learn how to effectively implement a DSL from scratch, using modern language workbenches and programming languages that enable the construction of expressive and effective DSLs.

Programming Styles

You are a good programmer. You are fluent in multiple programming languages. But can you program in style? This course is a journey through the landscape of programming languages and idioms. You will discover a multitude of dramatically different programming styles. A style arises from a set of constraints you impose on your code. Styles are independent of a given programming language. You can use many different programming languages to write code in a given style. In this course you won't just hear about and discuss various styles. You will practice those styles. You will look at programming as an art, and like a budding artist – be it a painter, a writer, or a musician – you will practice producing works in multiple artistic styles.

Design 101

This class teaches students a series of universal design principles that can be used for a number of tasks, such as enhancing the way a design is perceived, help people learning from a design, increase the usability and appeal of a design, and ultimately make better design decisions to create new and ameliorate existing designs. The principles are then put to practice in a series of contexts, such as graph design, table design, slide design, presentation design, etc.

Data Design & Modeling

Data design and modeling provides the foundation for representing and storing large amounts of structured, semistructured and unstructured data. Data can be persistent or volatile, it can be processed in batches or in continuous streams. Students become familiar with the CRUD primitives (create, read, update, delete) and the ACID/BASE transactional properties of existing SQL/NOSQL data management technologies. Dealing with large amounts of data requires to determine suitable sharding and replication strategies and understand to trade-off availability against consistency. Data quality and provenance are critical aspects to be considered to determine whether data is worthy of trust.

Electives

Software Engineering

Software engineering is the discipline of building software in a methodical way to ensure that the product satisfies its users' needs, is correct (or, more generally, dependable) and maintainable. The course teaches the students how to organize software development projects, how to analyze and specify software requirements, and how to verify software. The course will focus on the use of formal models and methods in software development. 1. Software lifecycle models. Project planning and management. Cost estimation. Standards. Maturity models. 2. Requirements elicitation and specification. 3. Notations and models for formal specification: state machines and Statecharts, Petri nets, declarative descriptions (Alloy). 4. Verification: testing, formal programme verification, model checking. The course will be based on lectures and exercise sessions. The students will also be given assignments, which will be presented and discussed in class.

Advanced Networking

This course covers advanced topics in computer networks, with a blend of theoretical and practical topics. On the theoretical side, the syllabus will cover mathematical foundations of networking, including discussions of queuing theory, control theory, information theory, and optimization. On the practical side, the syllabus will cover concepts and designs related to modern network architectures and technologies (e.g., data-center networks, software-defined networks), protocols (e.g., SPDY, HTTP/2, IPSec), and services (e.g., Zookeeper, DHTs). Students will gain hands-on experience with topics discussed in class through a series of exercises using network simulators and emulators.

Second semester

Software Quality & Testing

Avoiding failures in software systems by construction is impossible. This course is about methodologies, techniques and tools to check the quality of software systems, identify and remove faults before software deployment to reduce the possibility of runtime failures. Students will see the many facets of the problem and will learn methodologies, approaches and techniques to check the quality of complex software systems. Students will see the different approaches to testing and analysis and will understand the interplay of testing and analysis within the software development process.

Software Architecture

Architecture is not only necessary as the global blueprint to manage the complexity of large software systems, but should also be seen as the focus of the main design decisions influencing the quality attributes (modularity, maintainability, extensibility, portability, interoperability, reuse, performance, scalability, elasticity) of the resulting system. This class teaches the students to structure complex software systems using components and connectors while keeping track of the rationale behind their design decisions.

Visual Analytics Atelier

Real world problems more often than not have a fuzzy nature: they are not well defined problems with well defined data where one can just apply a statistical model and/or a machine learning technique. In reality, there is a clear objective and part of the solution is to define the problem by understanding and refining the underlying data. Solving these problems is an iterative process during which the data is integrated and explored multiple times from different angles, refining the model and understanding of it. Data exploration exploits a combination of interactive visualization and querying, as a means to understand the "shape" of the data, break down its complexity, and ask questions/test hypotheses. In this course we will see a number of data exploration and visualization techniques, and how to use them to iteratively solve a given problem by understanding data. The course will also cover how to explore ultra large datasets, which tools and techniques exist, and what are the challenges and constraints.

Information Modeling & Analysis

While the world provides an endless source of data, it does not come with a pre-defined model to understand it. Turning data into information requires to ask the right question and know how to find an answer. This course teaches students to turn data into information by using advanced modeling and analytics techniques based on machine learning. Like programming, teaching a machine to learn requires to design models selecting the most appropriate representation techniques. Unlike programming, teaching a machine to learn requires to select the right amount of training data and evaluate the quality of the predicted outcome. As they work through concrete application case studies, students will also gain a good understanding of the limitations and the assumptions behind each data mining technique so that they can be properly and correctly applied.

Electives

Compilers

This course studies the construction of optimizing compilers, focusing on the "backend" of the compiler: techniques for generating efficient code on modern architectures. We will cover dataflow analysis, programme optimization, and code generation across basic blocks, procedures, and complete programmes. We will look some of the key challenges for modern compilers and runtime systems: optimization of object-oriented languages, dynamic compilation, garbage collection, dependence analysis, and loop transformations. The bulk of your course grade will come from programming assignments that implement programme analysis, intermediate representations, and optimizations.

Information Security

This class exposes students to the fundamental concepts of cryptography, network security, and computer security. The growing importance of networks and distributed systems, and their use to support safety-critical applications, has made information security a central issue for systems today. The class centers on two main parts: security foundations (which includes security terminology, core cryptograhic principles, and secure protocols) and applied security (which discusses network security, computer security, software security, and web security). Students learn to critically assess the security properties of a system and make informed decisions about implementing secure processes. Most classes feature in-class labs where students are asked to implement a cryptographic primitive or secure protocol, or attack a vulnerable system.

Physical Computing

Physical Computing is about integrating the real world with sensing, communication, and computation. It is about rapidly prototyping devices that can react and interact directly with their environment, rather than being accessed through a keyboard and monitor. The class introduces students to the idea of using small, programmable microcomputers to build self-contained, physical systems that help automate everyday tasks. The course exposes students to basic electronics, microcontroller programming, wireless networking (WiFi and Bluetooth), mobile interfaces (smartphones), and embedded sensing. The class centers on Arduino and ESP development boards that allow one to rapidly build reactive and/or interactive everyday items, without the need for attaching a Mac or PC to them.

Robotics

Robotics addresses the design, construction, and automatic control of mechatronical systems. The course provides a general overview of robotics, focusing on autonomous mobile robots: autonomous systems which exist and move in the physical world, can sense their environment using multiple sensors, can reason about it to issue plans, and can act on it to achieve one or multiple goals. The fundamental concepts and models necessary to achieve such a view of a robotic system will be studied: Forward and Inverse Kinematics; Proprio- and Exteroceptive Sensing; Model-based and Model-free State Estimation ; Feedback-based Control; Paradigms and Architectures for Robot Control; Localization and Mapping; Motion Planning; Navigation; Coordination and Cooperation in Multi-robots and Swarms. The course includes theory classes, hands-on classes, and homework. Students will learn how to use the ROS and the simulator Gazebo, and will apply the learned concepts through the programming of both simulated and real robots.

Third semester

Software Analytics

Students will acquire key competences needed to analyze complex software systems and improve their maintenance and evolution. The first part of the course is focused on the study of techniques related to reverse engineering, defect prediction and analysis, and the mining and analysis of structured and unstructured data in software repositories. In the second part, students will apply the acquired expertise in the context of a software analytics project. A solid basis in object-oriented programming and software engineering is required.

Software Performance

This course teaches how the various layers of a computer system interact and affect the resulting performance. It performs two cuts down the system stack: one about the 'state' and the other about the 'behavior' of a system. The discussion of 'state' investigates memory usage of applications, leak detection, garbage collection, virtual memory management, and cache performance. The discussion of 'behavior' investigates call graphs, dynamic class loading, shared libraries and dynamic linking, control flow graphs, exception handling, compiler optimizations, and branch prediction. The course uses Java virtual machines and their internal operation as a running example and teaches basic static and dynamic programme analysis techniques. Given the quantitative aspects of performance, the course introduces basic instrumentation and measurement tools, experimentation and evaluation approaches, and data analysis and visualization techniques.

Knowledge Analysis & Management

Knowledge – the tacit insights, explicit understandings, practical know-how, and theoretical know-why – is what allows people and organizations to function intelligently. Knowledge Analysis and Management is a broad field that covers the entire lifecycle of knowledge: from identifying, representing, capturing, generating, structuring and sharing valuable intellectual assets. Students will learn to represent knowledge with natural and formal languages, access and integrate knowledge sources on the Web, classify and use knowledge for decision making, transfer and disseminate knowledge within and across organizations. The course will feature examples of case studies, methods and tools that are particularly suitable for managing software-specific knowledge, e.g., pattern languages, design spaces and architectural decision meta-models.

Hacking Project

The course allows students to put into practice the knowledge they acquired both in software and data engineering. Students will design and develop a complete software product analyzing streams of data in real time for its proper functioning integrating everything they have learned in the rest of the programme.

Software & Data Engineering Seminar

Students will visit one or more of the research groups of the Software Institute of the Faculty of Informatics to define the topic of their MSc thesis. Students are expected to (i) identify the research group in which they would like to work for their thesis, (ii) define, together with the chosen Advisor, the goal of the thesis, (iii) study the state-of-the-art related to the thesis, and (iv) write a detailed thesis proposal.

Fourth semester

Master Thesis

Students are expected to carry out a research project for their Master thesis under the supervision of a Faculty member. The thesis must represent an original contribution to the field of Software & Data Engineering. The specific research project is defined as output of the Software & Data Engineering Seminar.

Università
della
Svizzera
italiana

**Faculty
of
Informatics**

**Master of Science
in Software & Data Engineering**

2017/18