**Module Description   
Computational Design and Fabrication 1**

**Module level:**  
Master

**Language:**  
English

**Module duration:**  
one semester

**Occurrence:**  
Winter Semester

**Credits\*:**  
6

**Total number  
of hours:**  
180

**Self-study  
hours:**  
120

**Contact  
hours:**  
60

\* The number of credits can vary depending on the corresponding SPO version. The valid number is always indicated on the Transcript of Records or the Performance Record.

**Description of achievement and assessment methods:**  
Students' comprehension and practical application skills in the course are assessed through the completion and submission of design and programming assignments. These assignments are integral to demonstrating the student's ability to understand the presented topics and apply the concepts discussed in the module in a hands-on manner. The module's grading criteria are based on the quality of each student's individual submissions.

**Possibility of re-taking:**  
In the next semester: No  
At the end of the semester: No

**(Recommended) requirements:**  
Basic knowledge of programming (e.g., in Python) and/or Rhino Grasshopper (software) is recommended but not necessary.

**Contents:**  
The course bridges fundamental principles of geometric design and computing and leverages these insights to develop new algorithms and tools for 3D shape generation, simulation, structural design, and digital fabrication in architecture and engineering. Through a blend of frontal lectures and hands-on tutorials, students explore and learn innovative computational design solutions, emphasizing advanced fabrication and construction techniques across scales. This interdisciplinary exploration draws upon the close interrelation between various scientific disciplines, including mathematics, computer science, structural engineering, and architecture.

The course curriculum is structured to provide a comprehensive understanding of key topics:

* Introduction to computational design: Foundational concepts of structural and fabrication-aware design, setting the stage for integrating practical and theoretical knowledge.
* Computational geometry fundamentals: Covers essential data structures, data exchange formats, and algorithms that form the backbone of effective computational design methods.
* Advanced geometry processing: Introduction to form-finding methods and geometry processing, enhancing students' algorithmic thinking and programming skills for practical applications in design and fabrication.

**Study goals:**  
After attending the course, students will be able to:

* understand and apply the basic principles of parametric and algorithmic design,
* develop basic versions of key algorithms in architectural geometry, structural design, and digital fabrication,
* use common CAD tools as interfaces to self-implemented solutions,
* understand the scope and relevance of computational methods for architectural design research and practice,
* critically assess academic literature in the field to evaluate scientific rigour and relevance,
* integrate considerations of sustainability into design processes, reflecting on ecological, social, and economic factors.

**Teaching and learning methods:**  
This module is structured around a sequence of frontal lectures and tutorials that introduce various aspects of computational design with a special focus on structural design, shape optimization, and digital fabrication. Each session focuses on a specific topic, accompanied by programming exercises designed to reinforce learning objectives. The course provides a solid foundation for students to navigate the complexities of contemporary design processes using computational tools. These foundational principles are further reinforced through hands-on assignments, ensuring practical application and deeper understanding.

The teaching method is based on the following three pillars:

* Theoretical foundations: In a series of sessions, students are introduced to the theoretical foundations of computational design, gaining insights into fundamental concepts and methodologies.
* Practical applications: Students translate theoretical knowledge into practice through programming exercises, where they engage directly with computational tools to implement and experiment with various design techniques.
* Toolchain exploration: A curated sequence of computational design tools is introduced and demonstrated using real-world examples. Students leverage these tools to complete assignments, gaining proficiency in software utilisation while improving their design skills.

**Media formats:**  
Media-supported presentations (PowerPoint, videos, etc.) for the frontal lectures and a programming software platform for the tutorials.

**Literature:**  
References will be provided at the beginning of the semester

**Responsible for the module:**  
Dörfler K

**Lecturer:**  
D'Acunto P, Dörfler K, Hojjat M, Goldbach A