Module Specification

Computing Complexity

Version: 2021-22, v1.0, 29 Jun 2021

Contents

Part 1: Information

Module title: Computing Complexity

Module code: UBLLU1-15-M

Level: Level 7

For implementation from: 2021-22

UWE credit rating: 15

ECTS credit rating: 7.5

Faculty: Faculty of Environment & Technology

Department: FET Dept of Architecture & Built Environ

Partner institutions: None

Delivery locations: Frenchay Campus

Field: Architecture and the Built Environment

Module type: Project

Pre-requisites: None

Excluded combinations: None

Co-requisites: None

Continuing professional development: No

Professional, statutory or regulatory body requirements: None

Part 2: Description

Overview: The module is designed for complete novices in computational design platforms as one of 2 Bootcamp-based modules. Computing Complexity is both an introduction and revision of computational design fundamentals. The Bootcamp fast-paced module is to prepare for the urban scale design exploration .For the Autumn cohort intake, this module must be the introductory module, on which everything is built. The module focuses on coding and scripting to help the students' mindset be

agile between recursive and linear methodologies. In tandem, the students will investigate and understand the concept of complex systems as part of the History and Theory thread woven throughout the programme The module is designed to be standalone for other ABE MSc programme elective courses and CPD students.

The Bootcamp fast-paced module is to prepare for the urban scale design exploration. A series of fast-paced lab-based tutorials introducing computational methods and cutting-edge programming and scripting languages to ramp up the cohort's knowledge foundation. Tracking these technical learnings' understanding will be evaluated individually as a response to a primer brief.

In tandem, the students will investigate and understand the concept of complex systems as part of the History and Theory thread woven throughout the programme. Over two out of the four weeks, the theoretical component is introduced as a tandem strand to the taught skills. The students become familiar with how historical computational design research and practice learn from and design for the natural environment's emergent phenomenon. The two weeks start with a lecture and seminar and given a small brief accompanied by theoretical readings. The brief encourages the student to explore the emergent design through a non-digital/manual logical exercise underpinned by seminal theory. To tracking the understanding and the development of critical thinking, in groups of two, the students are encouraged to research the topic further and provide a visual presentation in the form of a design research poster, at the end of the two weeks.

The tuition targets computational macroscale design methods, such as community, urban, and global problem-solving interventions. Macroscale design virtual abstractions currently focus on simulation, generative, and mixed reality methodologies. These techniques are underpinned by complexity and cognitive theories. The fabrication strand will focus on rapid prototyping abstractions targeting small-scale modelling, using additive and subtractive fabrication methods, and microcontroller actuated embedded intelligence.
These technical skills and theoretical understandings begin in this module and continue in the UBLLV1-15-M Urban Sentience module, along with the design's transference from scripting to fabrication.

Features: The fast-paced learning in this module has two tandem tracks:
1) individual technical learning
The individual authored contribution considers each students' understandings of the computer lab-based tuition of computational methods and the ability to tap into the open-source community to find solutions agilely.
2)group work focussed on theory.
The required teamwork submission is evaluated as a single output to appraise the students' interpretation of the field's theoretical relationship. The group work is an icebreaker for the cohort, enabling them to collectively climb the learning curve and introduce them to this interdisciplinary field's collaborative nature.

Educational aims: Challenge the linear design process of architectural modelling by beginning with the programming methodologies and modes of development.

Critically evaluate concepts and evidence from a range of sources or design research and practice underpinning complexity theories.

Systematic understand of the fundamentals of current programming methods (syntax and logic) and its interrelationship with the biological and mathematical models.

Understand and explore small-scale digital fabrication methods and the use of prototyping machinery.

Works creatively and effectively within a team, supports or is proactive in leadership, negotiates in a professional context and manages conflict and creative differences. Proactively seeks to resolve conflict.

Outline syllabus: The module runs as a short, intense skills lab-based module with a theoretical thread addressing emergence and complex systems.

As an introductory course, there will be a general lecture on the theoretical and historical overview of the topic, focusing on complex systems and their application in academic and practice of design. Closer to the end of the term, one-to-one clinical hours will be scheduled to ensure students are supported as they climb the learning curve.

This module's simulation strand is delivered in a series of intense day-long and half day-long sessions in a computer lab. This learning's Bootcamp nature aims to ramp up and bring up the cohort's programming knowledge base to fundamental skills. The programme used in this module is defined by the tutors teaching the module, and the current practices. The programming topics covered are:

Coding best practices (Syntax and logic)
Fundamentals of programming
Programming current algorthims simulating and solving complex systems

The fabrication strand is delivered similarly to the simulation strand, namely, Bootcamp delivery. The tuition focuses on current modes of physical computing and embedded intelligence within a macroscale design context.

The historical, theoretical and critical understanding of complexity will be delivered as lectures and seminars over two weeks. The lectures and readings focus on various complex system theories. The students are invited to read and distil seminal texts to be discussed in the seminar sessions. In a small group of 2 to 3, the team is invited to translate an algorithm in an analogue (non-computational) exercise.

Part 3: Teaching and learning methods

Teaching and learning methods: The fast-paced learning in this module is delivered on two tracks:

1. Lab-based practical skills:

The practical lectures, exercises, and primer project are designed to facilitate competency acquisition through applied and indirect learning, building knowledge by introducing the new subjects and gained coding and scripting skills.

2. Lecture-based, seminar discourse, and self-directed study:

This track enables students to support their independent learning by exploring more

profound computation design issues and receiving feedback. Students are exposed to the long track of computational design accumulated during the past two decades and encouraged them to build on that body of work. The introduction of analogue logic exercises and the invitation to abstract the process into visual representation allows students with non-design and design background to understand and communicate complex information.

For both methods, studies conducted outside of contact hours is essential to complete the assigned work successfully. Students will be expected to come prepared for the module sessions with work-in-process and seminal readings. Feedback will be in the form of direct verbal and/or written.

Marking criteria and assessment format will be indicated on the project brief will be accessible to the students at the beginning of each project. Students' work will also be exposed to critical peer evaluation through discussion.

Presentations by the students will enable peer learning and help students develop the skills and capabilities to analyse problems, negotiate, make decisions, and present solutions to problems collaboratively.

Module Learning outcomes:

MO1 Interpret complex systems theory and critically reflect on the interrelationship with the academic and practice of computation design processes.

MO2 Identify and apply the fundamentals of programming and fabrication methods to resolve macroscale design, in response to a primer brief

MO3 Produce an artefact (simulation/fabrication) applying researched knowledge of complex system algorithms in design processes.

MO4 Flexibly and creatively select appropriate computational methods by proactively undertaking substantial investigation and accessing resources towards a design research process.

MO5 Present an interpretation of key complex systems theories in a graphical and verbal illustration at a high level of abstraction, arguing from competing perspectives.

Hours to be allocated: 150

Contact hours:

Independent study/self-guided study = 120 hours

Computer-based activities = 20 hours

Total = 150

Reading list: The reading list for this module can be accessed at readinglists.uwe.ac.uk via the following link https://uwe.rl.talis.com/modules/ubllu1-15-m.html

Part 4: Assessment

Assessment strategy: The assessment strategy adopted by this module involves a mix of practical skills assessment, and a verbal and graphical presentation to reflect on key complexity theories applied in design. The practical assessment is designed to evaluate students' fundamental practical coding skills and apply them to generate solutions in a project. The students are expected to design and present a visual presentation (poster or PowerPoint) responding to the theoretical strand brief. The students must appreciate the criticality of communicating complex data and logic visually.

Formative assessments: will be evaluated in one-to-one tutorials.
Summative assessments: will be evaluated in both 1) verbal/graphical presentation (controlled condition evaluations) for the theoretical teamwork project, and 2) portfolio documentation of the student's design demonstrating their ability to define inputs and select an algorithm towards finding a solution as an output, for the individual practical project.

Resit Strategy: The portfolio will include the revised individual work to the same brief,

to evaluate students' fundamental practical coding skills and application. It will also include an individual experiment with a reduced scope to acknowledge the change form group work to an individual resit. This revised experiment will include the theoretical component as part of the design process. The student will also be asked to reflect on what has been lost, gained, and learned by working individually or in a team.

Assessment criteria will be made available to the students, along with each assignment brief.
Feedback: there will be peer and tutor feedback throughout the module critiques. The students will be invited to provide self-assessment. Written feedback on completion of the projects.

Assessment components:

Final Project - Component A (First Sit)
Description: Individual Project
Weighting: 50 %
Final assessment: No
Group work: No
Learning outcomes tested: MO2, MO3, MO4

Presentation - Component A (First Sit)
Description: Seminar and workshop group work
Weighting: 50 %
Final assessment: Yes
Group work: Yes
Learning outcomes tested: MO1, MO4, MO5

Portfolio - Component A (Resit)
Description: Portfolio: requires and updated failed component and merging independent practical work and theoretical investigation, the portfolio should include a reflection of losses, gains, and learnings from individual and teamwork, and presented coherently.

Weighting: 100 %

Final assessment: Yes

Group work: No

Learning outcomes tested: MO1, MO2, MO3, MO4, MO5

Part 5: Contributes towards

This module contributes towards the following programmes of study:

Computational Architecture [Sep][FT][Frenchay][1yr] MSc 2021-22

Computational Architecture [Sep][FT][Frenchay][1yr] MSc 2021-22

Computational Architecture [Sep][PT][Frenchay][2yrs] MSc 2021-22

Computational Architecture [Sep][PT][Frenchay][2yrs] MSc 2021-22