System architecture - elective package

System architecture (This flyer will be updated soon)	
Offered by	Department of Mathematics and Computer Science
Language	English
Primarily interesting for	All students, but most relevant for students with background in Mechanical Engineering, Physics, Web Science Some basic understanding of Propositional logic (e.g. like in 2IT60 Logic and set theory). Mastery of elementary programming techniques (e.g. like in 2IP90 or 2IS60).
Prerequisites	Required courses: - Recommended courses: -
Contact person	Dr. P.J.L. Cuijpers (P.J.L.Cuijpers@tue.nl)

Content and composition

This package brings together three courses that center around the interplay of computer and software. The Computer Systems course explains the working of a computer as a device consisting of several virtual layers, stacked on top of each other. The Operating Systems course surveys the services that are provided by the collective of these layers and what concepts are central to the understanding of system as a whole. In the Design-Based Learning project software for a controller, on the one hand, needs to be written and a physical machine, on the other hand, needs to be constructed that together comprise a so‐called cyber-physical system.

Course code	Course name	Level classification
2IC30	Computer systems	-
2IO75	DBL embedded systems	-
2INC0	Operating systems	-

Course description

Computer systems (2IC30)

Introduction to the layered structure and workings of general-purpose machines that can execute computer programs, spanning the conceptual gap between simple switches on the one hand and the abstract machine that constitutes a "computer" on the other. Topics addressed in the course include combinatorial and sequential circuits, number representation and arithmetic, basic computer building blocks, instruction sets, machine and assembly language. Practical work involves the use of propositional logic for the design of simple combinatorial circuits, the guidance of state machines for the construction of sequential circuits, the writing of and the translation of Java algorithms into assembly language.

DBL embedded systems (2IO75)

TThis is a project about the design and construction of embedded systems. In addition the development of professional skills, like SCRUM, efficient reporting, and giving engaging presentations, is an important aspect of this project. Each group of approx. 6 participants specifies, designs, models, and generates code from models, and actually builds a machine that, under control of a small computer, performs a self-invented task, such as sorting objects. The machines work together in the sense that they are all receiving objects from a shared belt. Students are encouraged but not obliged to explore the possibility to build machines that take on the responsibility to communicate adequately with one another to coordinate fair processing of objects. The project is concluded with a presentation and demonstration of the machine, during which the teacher likes to poke pencils into machines to see if they are also robust against unexpected internal failures.

System architecture - elective package

For facilitating group work each group will be assigned a room in MetaForum for two time slots including slot B. The second time slot is to be chosen by the student when enrolling for this DBL in OASE

Operating systems (2INC0)
This course is an introduction to the field of computer operating systems. The course contents are fairly traditional: processes, threads, process scheduling, atomicity and synchronization, deadlock, memory management, virtual memory and file systems. The March 2014 textbook by Silberschatz, Galvin and Gagne entitled Operating System Concepts (already in its 8th edition) will be used, that comes with the standard theory as well as with useful exercises. Examples taken from the book and also from other literature will be discussed, often referring to Linux and Windows.

In many cases we will also use POSIX, the portable operating systems interface. We will mainly use the C programming language when considering concrete code, but occasionally use an ad-hoc Pascal or Java-like language to express algorithms.