Program of the course on Component-Based Software Design

Teachers:	Giorgio Buttazzo, Marco Di Natale, Enrico Bini, Emanuele Ruffaldi
Semester:	Second
Credits:	9
Description:	In most application domains related to embedded systems (especially in consumer electronics, automotive, avionics, and multimedia systems), software size is growing exponentially, since more and more functions, previously developed in hardware, are now being implemented in software, and also new functions are continuously being added in order to lead the market and satisfy customers' requirements. Moreover, complex systems are organized into interacting components (independently developed by different teams), tested, and later integrated to compose the complete system. The major problem caused by such a high software complexity is a lack of predictability, mainly due to the interference generated by several concurrent activities sharing common resources, especially on multicore platforms. Even worse, software modules that are certified to be correct in isolation, could not behave as expected when integrated together in the whole system.
Objectives:	The objective of the course is to present a set of methodologies for designing, analyzing, and implementing complex embedded software as a number of interacting components, using precise interface operations. The course analyzes the main sources of interference in an embedded system and presents techniques to reduce and analyze them. Particular examples will be provided for multicore and heterogeneous platforms. Standards for component-based development will also be presented.
Topics:	1. Introduction to component-based design: motivations, sample applications, terminology, definitions, and major problems. 2. Interference analysis and techniques to reduce the interference. Resource reservations servers. Schedulability analysis of single components. Hierarchical component-based systems. Resource sharing protocols for hierarchical systems. 3. Parallelisms programming. The OpenMP standard and examples. Exploiting GPU accelerators: CUDA and OpenCL; Higher level tools: C++ based libraries and code generation techniques. 4. Task scheduling and allocation on multiprocessor platforms. Task models for analyzing parallel programs. Global and partitioned approaches. Response time analysis for task graphs. 5. Processor abstraction and interface. Efficient algorithms for the design of the interface. Multiprocessor abstractions. Applications models. Application partitioning and resource allocation. 6. Virtualization: general concepts, machine/processor virtualization, full vs. para- and hardware-assisted virtualization. A brief overview of the Xen virtualization approach. 7. Standards for component-based development. The AUTOSAR standard for automotive systems. 8. UML components, hierarchical state machines and automatic code generation.
Prior skills	Real-Time and Distributed Systems, Design of Embedded Systems.
Exam:	Software project and written test.
Material:	Notes from the teachers.