

SOFTWARE ENGINEERING FOR CRITICAL SYSTEMS (SE-O-04)

AIMS:

Critical Systems (CSs) are those whose operation poses a risk to human life, health, economy or environment. Typically, CSs are large and complex industrial systems or products which have been constructed through the effort of multi-disciplinary teams. The engineering and assessment of modern CSs is thus complex, multi-disciplinary task, often involving mechanical, structural, and electronic and software engineers and psychologists. The skill of economists is needed; sociologists also have a role in addressing the interaction between complex systems and teams of operators.

This module aims to introduce and critically analyse CSs. Requirements for the engineering of CSs will be introduced and the role of formal approaches in the life cycle of CSs will be explored.

LEARNING OUTCOMES:

Upon successful completion of this module, the student will be able to:

- critically evaluate the current taxonomies of CSs including international standards;
- critically evaluate the use of formal methods in the life cycle of CSs;
- appreciate the essential issues of time-critical systems both at the specification and design stages, including scheduling techniques;
- critically evaluate the use of temporal logic for the engineering and re-engineering of CSs.

SYLLABUS CONTENT:

- Classification and analysis: examples, analysis, taxonomy, standardization efforts (UK, Europe, USA).
- Time-critical systems: what are they; technical issues (design and architecture, specification, scheduling, reliability and dependability).
- Role of formal approaches: software in CSs (and real-time); goals and objectives; assurance; formal approaches in CSs life cycle; examples of formal approaches (model-based, logic-based, process algebras, refinement).
- Refinement, abstraction, and evolution: rationale and model (why, what is needed, computational model); interval temporal logic (syntax, semantics, tool (e.g. Tempura: executable subset); refinement calculus (the temporal agent model, algebraic laws, example); abstraction calculus (abstraction rules, example); evolution (guided evolution, example, tool (e.g. Ana Tempura)).

PREREQUISITES: None

RECOMMENDED ASSESMENT: Coursework and unseen paper