ELEC5618: SOFTWARE QUALITY ENGINEERING

LECTURE NOTES

WEEK 1: AN INTRODUCTION TO SOFTWARE QUALITY ENGINEERING

Software Quality Engineering

- A process that ensures that developed software meets and complies with defined or standardised quality specifications
- An ongoing process within the software development lifecycle that routinely checks the developed software to ensure it meets desired quality measures

WEEK 2: FUNDAMENTALS OF SOFTWARE QUALITY ASSURANCE

Software Quality

- The degree to which a system, component, or process meets specified requirements
- Conformance to explicitly stated functional and performance requirements, explicitly documented development standards and implicit characteristics that are expected of all professionally developed software

Software Quality Assurance

Requirements	Design	Implementation	Operation
Completeness is hard to achieve (complexity)	For reliability, manufacturability, maintainability	Limited success with statistical process control	Fixing found bugs

Software Quality Control: fault or failure detection through static or dynamic testing of artefacts

- Outputs: acceptance decisions, rework, process adjustments
- Tools: Pareto analysis, statistical sampling, Six Sigma, quality control chards

Software Quality Models & Characteristics

Factor-Criteria-Metrics-Model	ISO/IEC 9126
Factors (to specify): they describe the external view of the software, as viewed by the users **Criteria (to build):** they describe the internal view of the software, as seen by the developer **Metrics (to control):** they are defined and used to provide a scale and method for measurement	**Functionality:** are the required functions available? **Reliability:** how reliable is the software? **Usability:** is the software easy to use? **Efficiency:** how efficient is the software? **Maintainability:** how easy is it to modify? **Portability:** how easy is it to transfer environments?

Software Quality Assurance Approaches

Manual	Automatic
Techniques: writing unit test with Junit, code review + low false positive rate + defects of requirement - difficult and expensive - hard to reuse	**Techniques:** source code analysis, run-time analysis + find implementation defects + easier to reuse - potential false positive - cannot fine requirements defects