Technische Universität
Kaiserslautern

o1o1seda o1o1oo

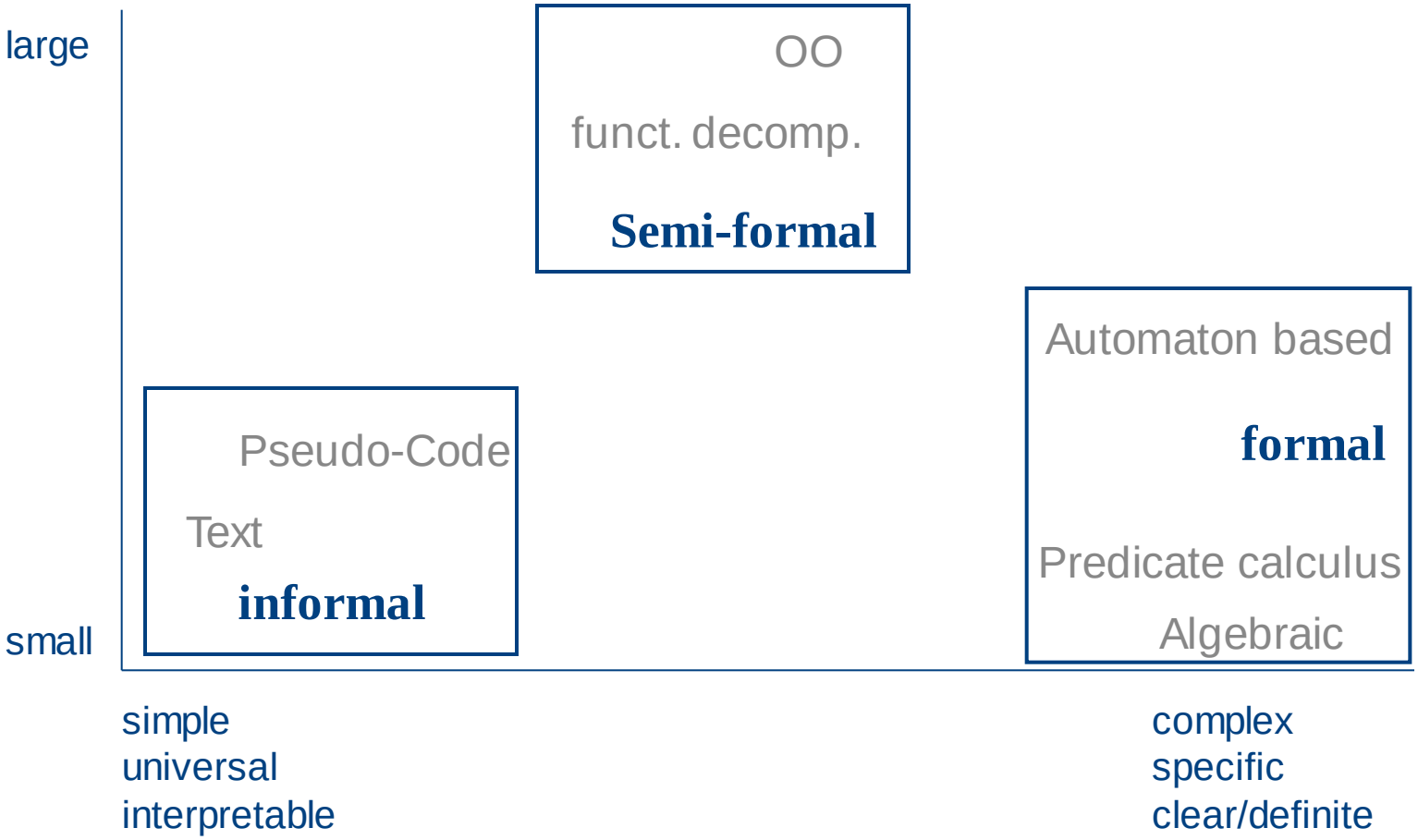
software engineering dependability

# Software Quality Assurance
# Overview

# Content

- Situation Analysis
- Consequences
- Classification of Test Methods
- Test Methods

# Construction Methods



# Construction Methods: Prognosis

- Text will be eclipsed increasingly
- Object-oriented development methods (OOA, OOD, OOP) will establish increasingly due to their excellent properties with regard to the mastery of large software systems
  - The standard for OOA and OOD is UML presently
  - The standards in programming are C++ and Java
- In some applications functional decomposion techniques (e.g. SA) will be preserved
- Formal techniques will remain confined to specific application areas

# Situation Analysis of Software Development in Practice

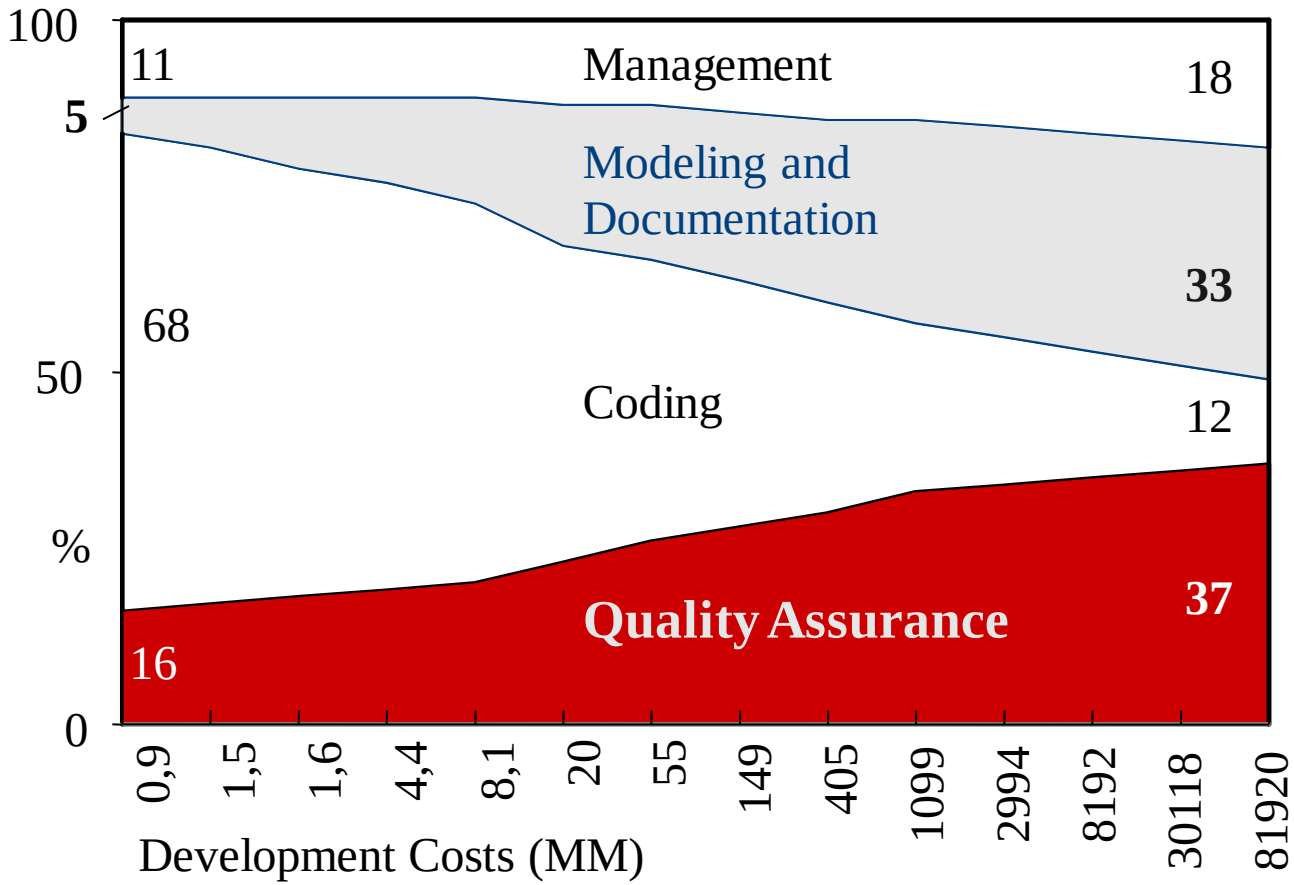
- Question: Who ensures that the construction steps are perfectly done?
- Answer: Nobody!
- Consequence: The software development is not completed with the implementation of the code. Often extensive tests are necessary.
- Typical approach:
  - Ensure that the development processes are suitable ➔ quality management
  - Ensure that the construction steps provided the desired results ➔ quality assurance (can also be done more or less formally)

# Situation Analysis of Software Development in Practice: Increasing Quality Requirements



- For 50% of the failures in the industrial sector software faults are responsible
- According to Cusumano the located defects have developed in 1000 lines of source code as follows:
  - 1977: on average 7- 20 defects
  - 1994: on average 0,2 - 0,05 defects
  - In 17 years the defect rate could be lowered about 100 fold
- Increasing burdens
  - Application software is often used 20 years or longer
  - As the application environment of this application software changes permanently this software also has to be adapted constantly. These permanent adaptation processes often cause two-thirds of all software costs.

# Situation Analysis of Software Development in Practice




According to data from: Jones C., *Applied software measurement*, New York: McGraw-Hill 1991

# Situation Analysis of Software Development in Practice



Design methods:

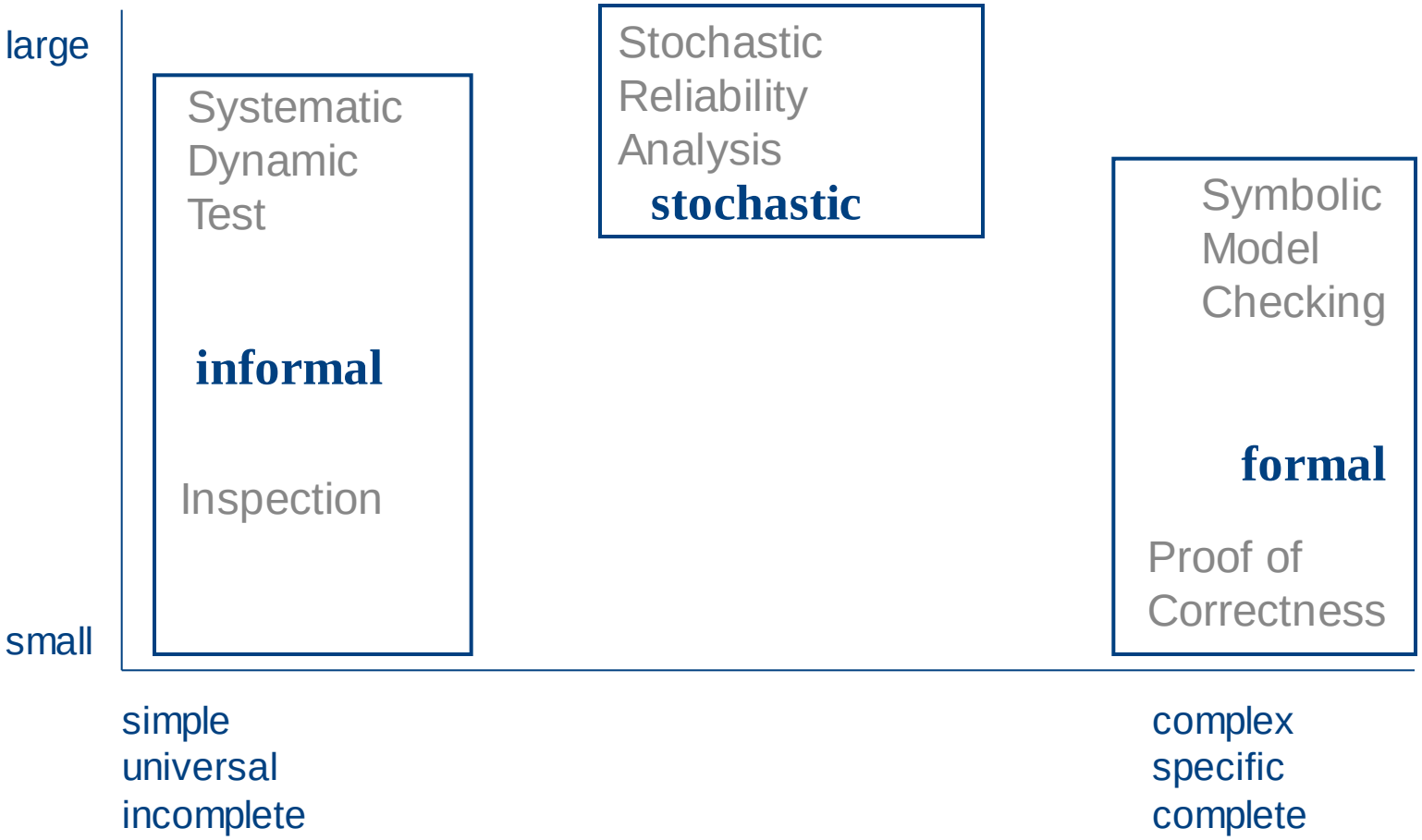
- Still widespread use of informal methods (text)
- High interest in semi-formal methods (in particular OO)
- Minor use of formal methods

- Quality management:
  - Trend towards the certification of quality management processes (ISO 9001)
  - Stage of capability maturity model-based assessment methods (e.g. SPICE)
- Quality assurance methods:
  - Informal methods are frequently applied (testing, review techniques)
  - Formal methods (proofs) often fail concerning the complexity of the software and the properties of modern programming languages
  - Stochastic methods are not widespread, but are increasingly required in critical application areas in particular

# Categories of Quality Assurance Methods

- Informal Methods:
  Methods based on plausibility which produce incomplete results
  - Testing
  - Inspection and review
- Stochastic Methods:
  Methods which produce statistically reliable, quantified results
  - Stochastic reliability analysis
- Formal Methods:
  Methods which produce formally complete results on the basis of formal specifications
  - Formal verification techniques (Proofs)

# Quality Assurance Methods



# Quality Assurance Methods: Prognosis

- Systematic informal methods are widely used and are obligatory for many application areas where they are required by appropriate standards
  - Function-oriented test planning
  - Tool supported structural testing
- Test support is essential (e.g. regression tests)
- Static analyses are additionally used
  - Inspections in early phases
  - Tool supported analyses of code in addition to the analyses performed by the compiler (in particular concerning the languages C / C++ / Java)

# Situation Analysis: General Consequences

- Mature Processes ...
  - ... are necessary, but barely offer a differentiation of competitors
  - ... operate confidence-building, but provide no further statements
- Design methods:
  - Informal methods are simple and universal, but often insufficient
  - Semi-formal methods allow the description of extensive software, but not the description of critical properties of technical software (e.g. safety).
  - Formal methods are powerful, but are often too specific
- Quality assurance methods:
  - Informal methods are indispensable, but produce no sufficient completeness (testing, inspection methods)
  - Formal methods (proofs) provide – to some degree – complete results, but often fail due to preconditions, that are not fulfilled
  - Stochastic methods generate well-defined, reliable results, but require mathematical knowledge which is often not given in practice respectively tools which are not available on the market

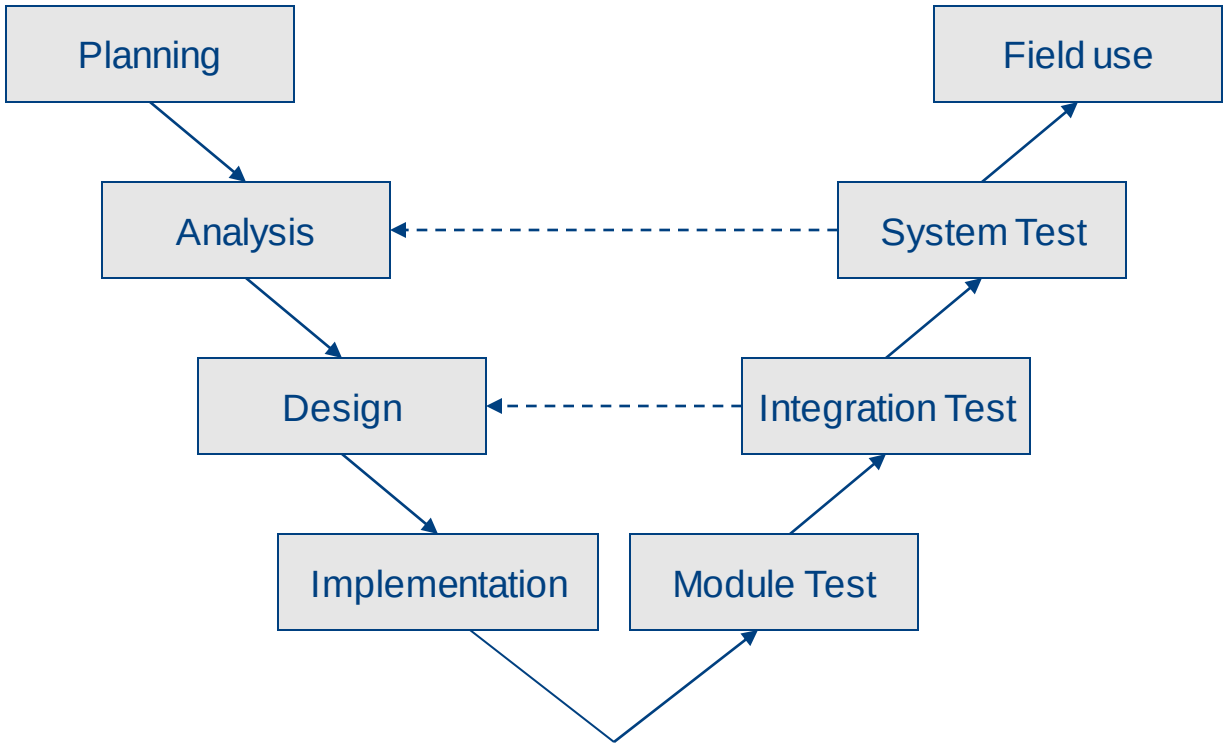
# Consequences

- Software quality has to be assured:
  - Evaluation, validation and improvement of development processes
  - Accompanying quality assurance during the early development phases
  - Testing of the implemented software (the code)
- The software is large ➔ several test phases are required
- Highly different demands on software (*experimental prototype* up to *engine control of a commercial aircraft*) ➔ need of different methods between „trial" and „proof"
- It is not possible to guarantee, that code is failure-free ➔ it is required to determine the residual risks ➔ quantitative analysis methods

# Test Phases

- The precondition for testing large software systems is their modular structure. Monolithically realized large systems cannot be tested.
- Module test
  - Testing of the modules
  - Testing the correct function of a module w.r.t. the module specification.
- Integration test
  - Testing of the interaction of the modules
  - Incremental assembly of the modules building the integrated system. Testing of their correct interaction.
- System-/Acceptance test
  - Testing of the functionality and efficiency of a software with regard to the requirements determined in the definition phase.
- Benefit of testing in different phases is the reduction of the respective complexity to a reasonable level.

# Test Phases



# Classification of the Quality Assurance Methods

- The analytic quality assurance techniques are
  - dynamic or
  - static.
- They aim at either
  - the proof of the correctness,
  - the detection of faults or
  - the determination of particular module properties.
- Analytical quality assurance can be divided into
  - Formal verification,
  - Symbolic testing,
  - Dynamic Testing, and
  - Static analysis.
- Sub-categories are necessary.

- Properties of dynamic testing:
  - The executable program is provided with concrete input values and is executed
  - Program may be tested in the real environment
  - Never complete (it is not possible to test all possible inputs)
  - Correctness of the tested program cannot be proven.
- Characteristics of the application of dynamic test methods in practice:
  - widely-used.
  - Often unsystematically applied.
  - Tests often not reproducible.
  - Diffuse activity (management difficulties).

# Test Methods
# Static Analysis

- Properties:
  - No program execution is required.
  - No input values are selected.
  - The static analysis concentrates on particular partial aspects.
  - It is no proof of correctness.
  - Some static analyses can detect faults directly.

- Sub-categories:
  - Measurement (Metrics)
  - Generation of diagrams and tables
  - Data flow anomaly analysis
  - Testing of programming conventions
  - Inspection and review techniques

- Properties:
  - Formal verification uses mathematical techniques to prove the consistency between specification and implementation.
  - A formal specification is necessary.
  - Verification may be almost completely automated (exception: e.g. finding loop invariants).
  - Requires preconditions which are often not fulfilled in practice.