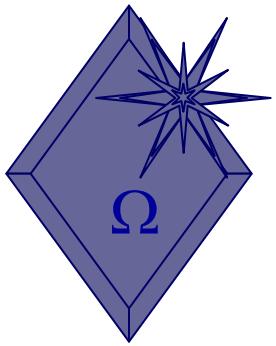
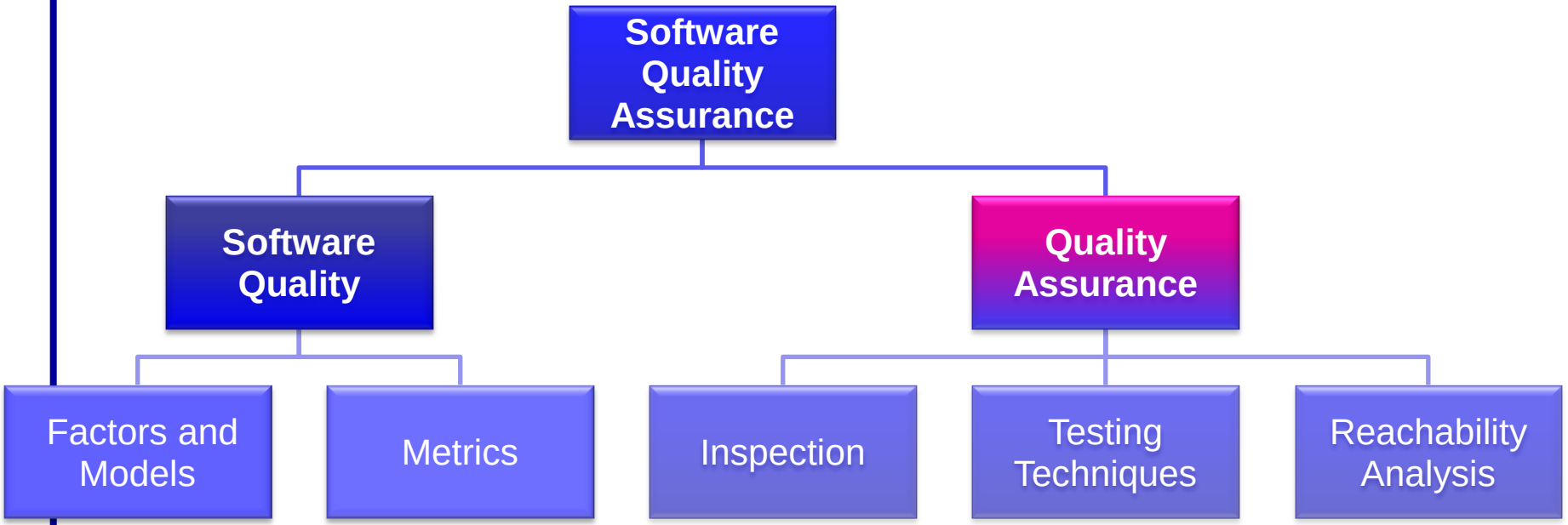
Lecture 3: Verification and Validation

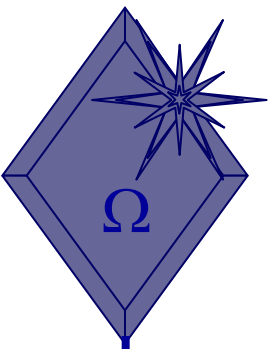


Software Quality Assurance (INSE 6260/4-UU)

Winter 2016

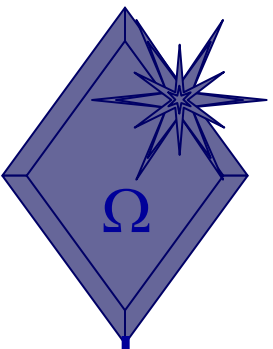
INSE 6260/4-UU





Overview

- Preliminary Notions
- Validation and Verification Approaches
- Software Inspection

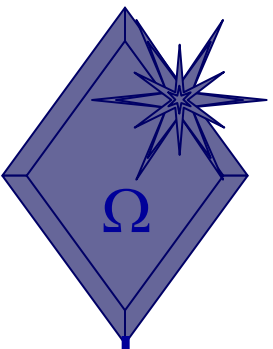


Verification vs. Validation

- Verification:

  "Are we building the product right"
- The software should conform to its specification
- Validation:

  "Are we building the right product"
- The software should do what the user really requires

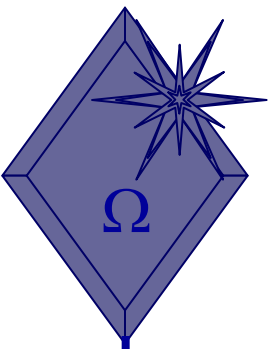


Verification, Validation and Qualification

Verification - The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase

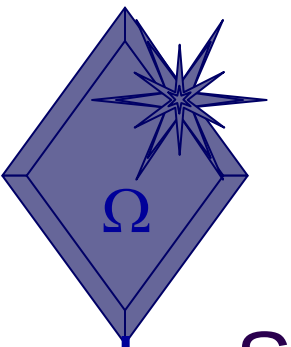
Validation - The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies the requirements

Qualification - The process used to determine whether a system or component is suitable for operational use



The V & V Process

- Is a whole life-cycle process - V & V must be applied at each stage in the software process
- Has two principal objectives
  - The discovery of defects in a system
  - The assessment of whether or not the system is useful and useable in an operational situation

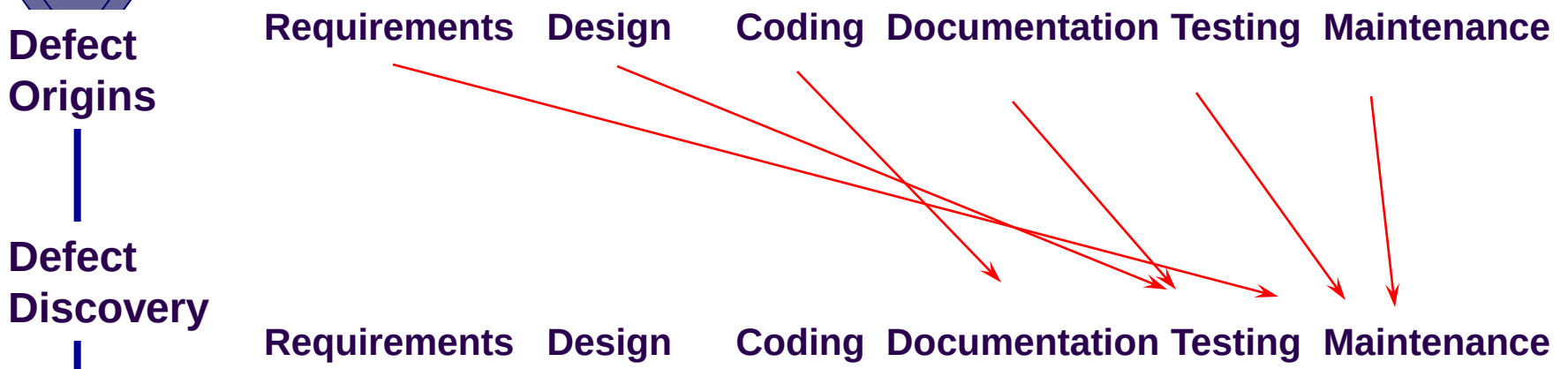


V & V Goals

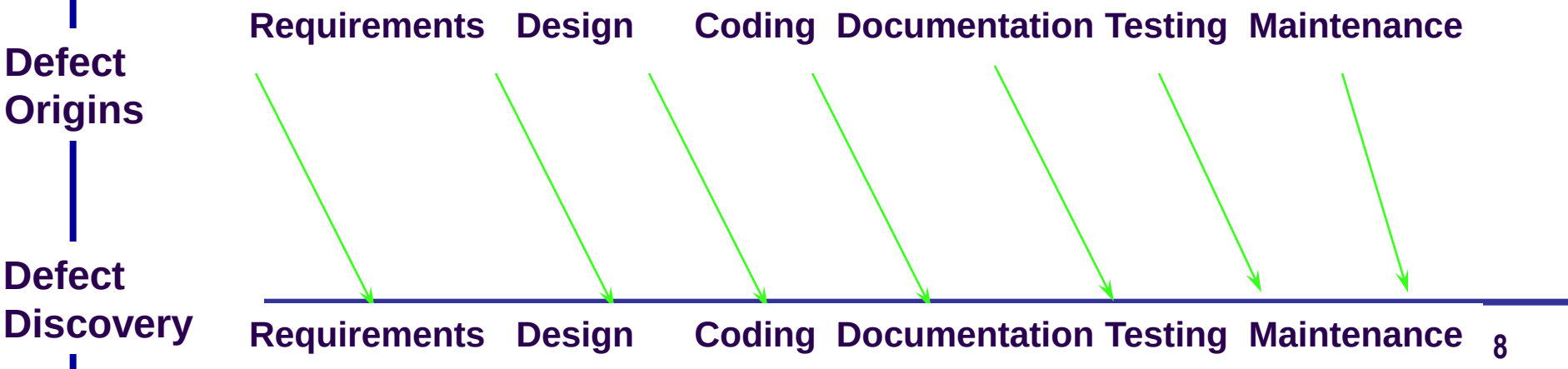
- Should establish confidence that the software is **fit for purpose**
- Does NOT mean completely free of defects
- Rather, it must be good enough for its intended use and the type of use will determine the degree of confidence that is needed

Defect Origins & Discovery

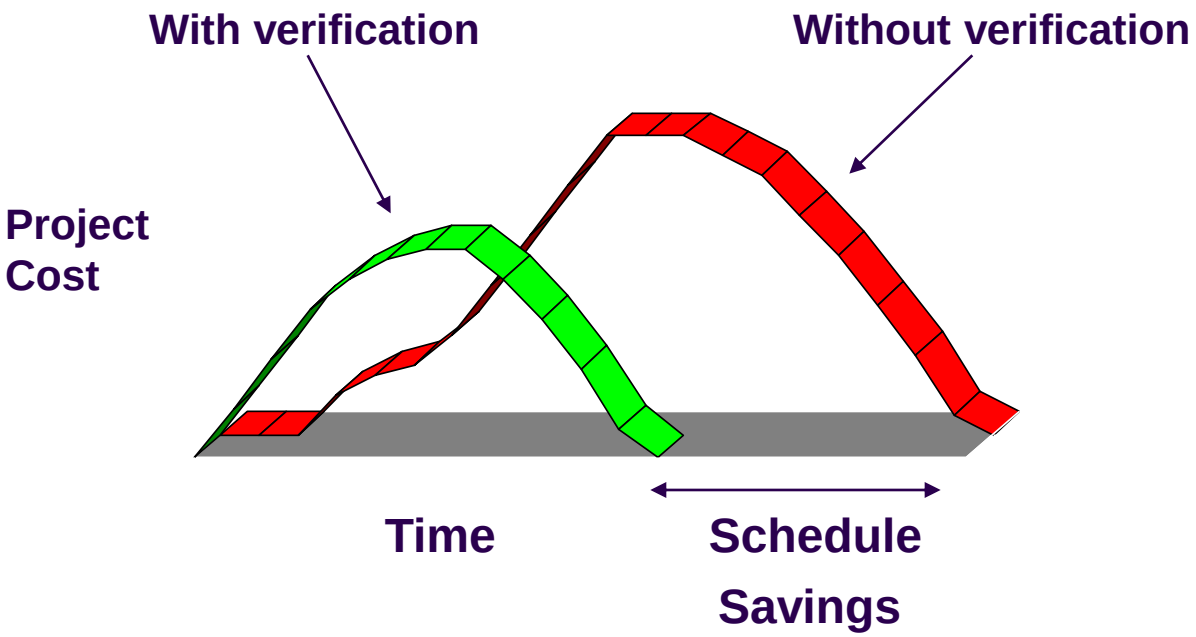
When Validation is the Primary Removal Method:

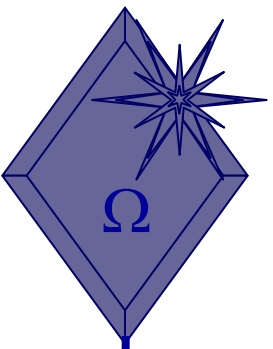


With Technical Reviews and Verification:



Verification Reduces Project Costs & Schedule

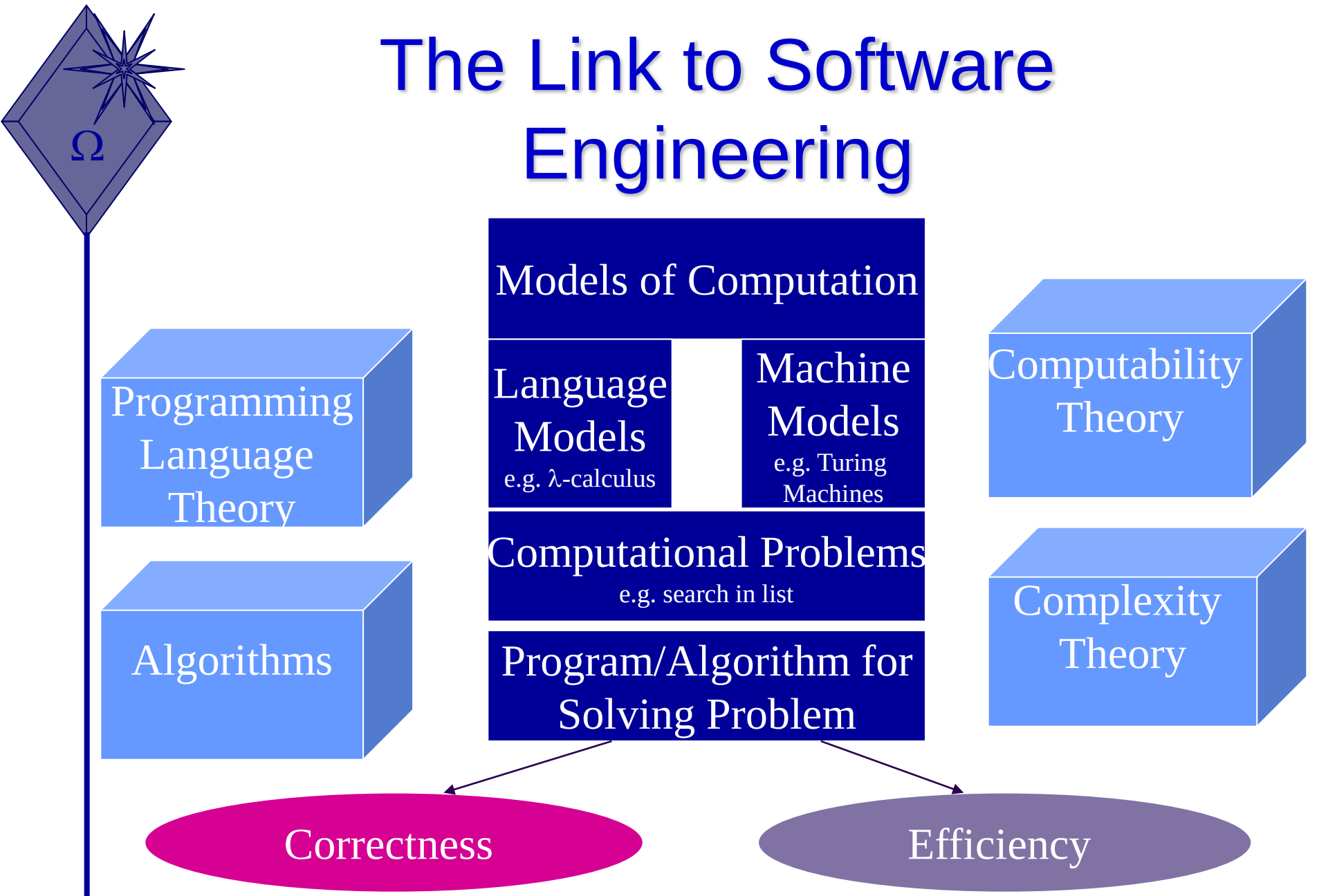


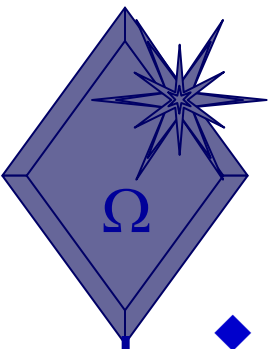


Overview

- ✓ Preliminary Notions
- Verification Approaches
- Software Inspection

The Link to Software Engineering

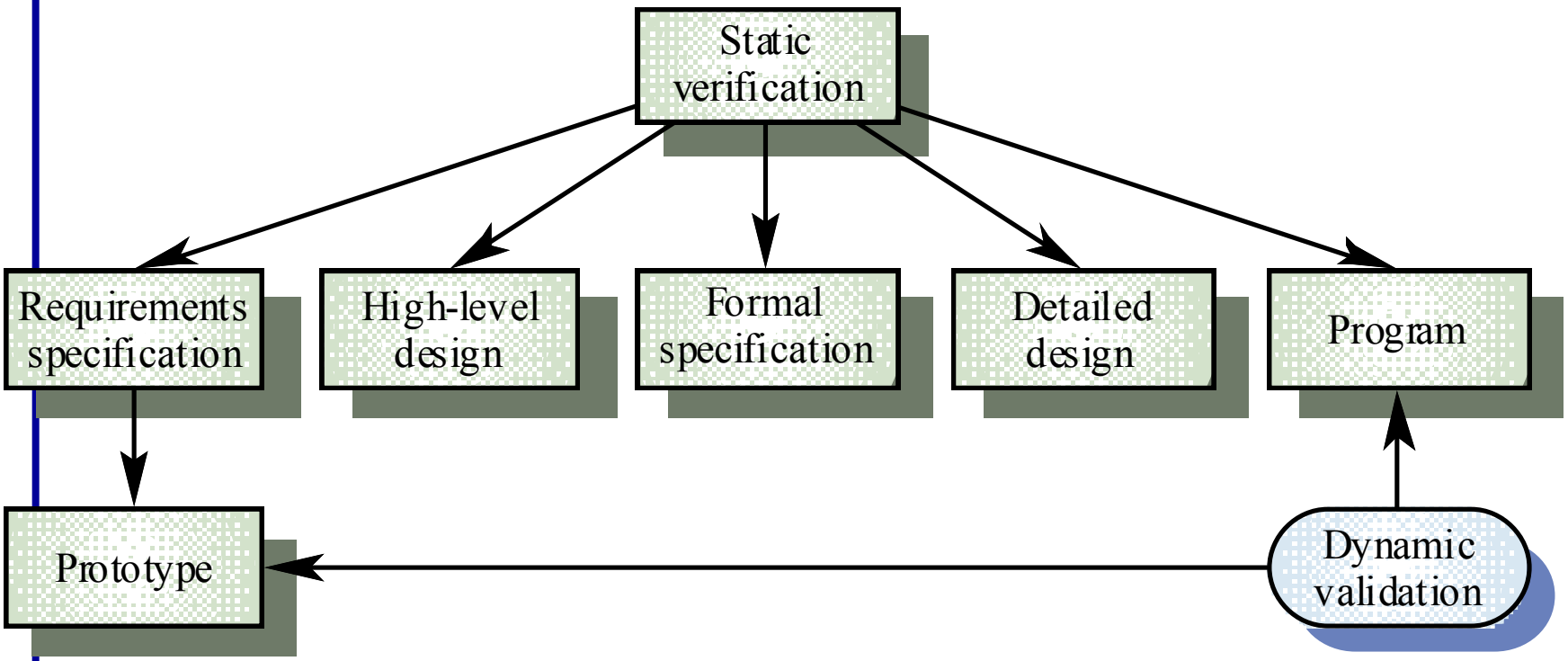


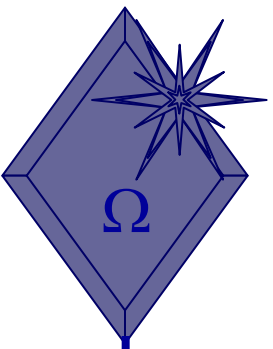


Two Approaches

- Static verification
  - Concerned with analysis of the static system representation to discover problems
  - May be supplemented by tool-based document and code analysis
- Dynamic verification (testing)
  - Concerned with exercising and observing product behaviour
  - The system is executed with test data and its operational behaviour is observed

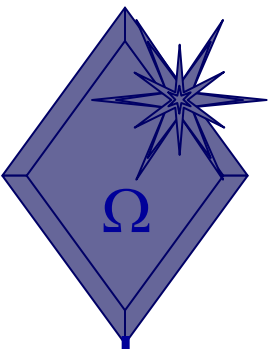
Static and Dynamic Verification





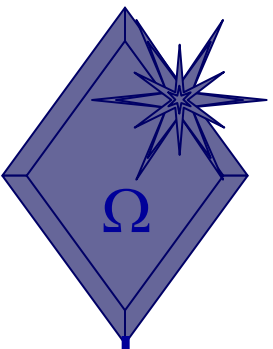
Formal and Informal Verification

- Formal: Applying formal methods to software verification
  - Mathematics
  - Logics
- Informal: Anything else is informal, including review and inspection



Formal Verification

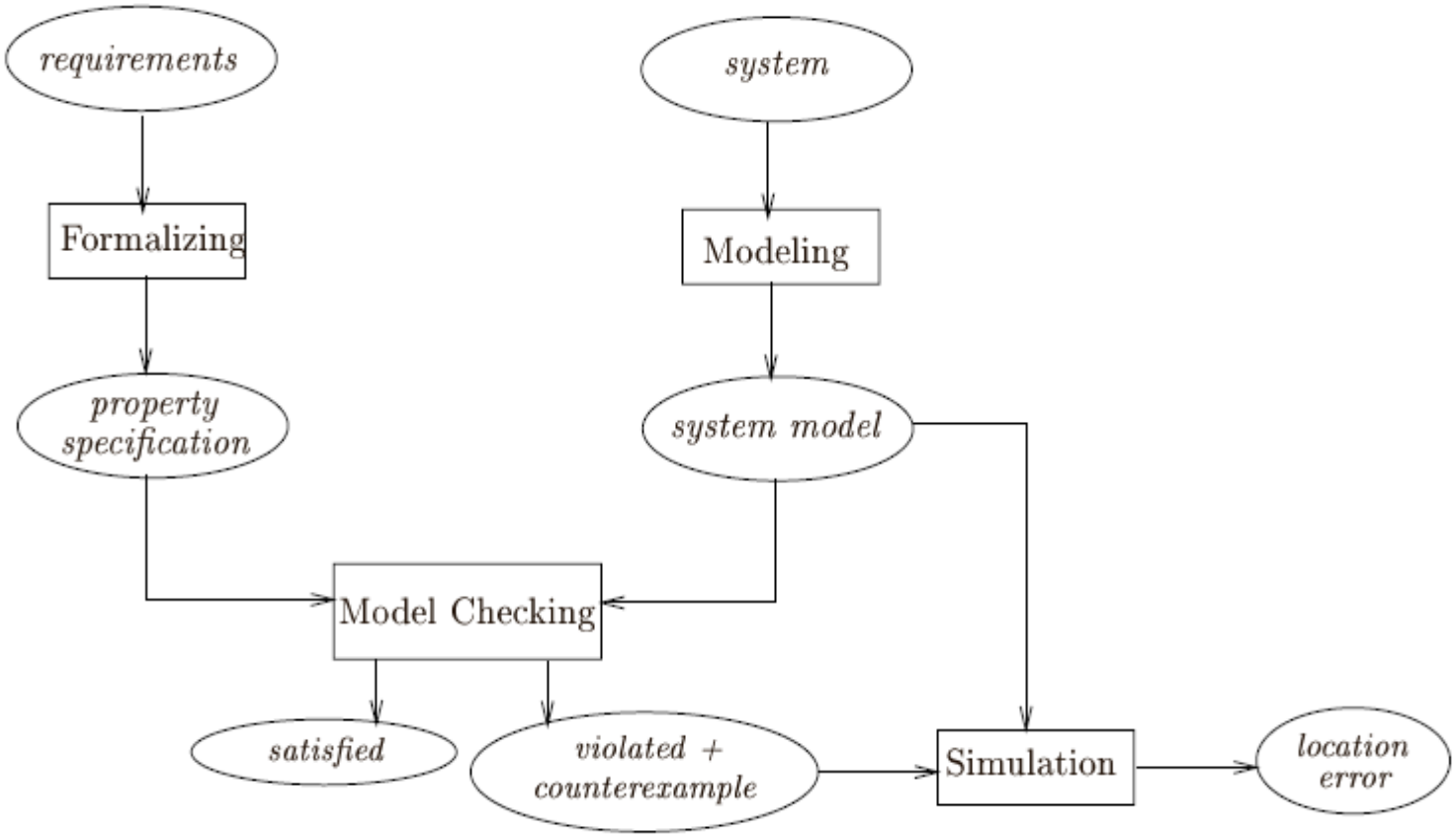
- Applying mathematics at large for modeling and analyzing software
- Establishing software correctness with mathematical rigor
- Two classes of formal verification techniques:
  - Proof-based techniques: theorem proving
  - Model-based techniques: model-based testing, model-based simulation, model checking

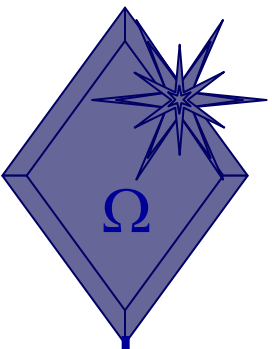


Model Checking

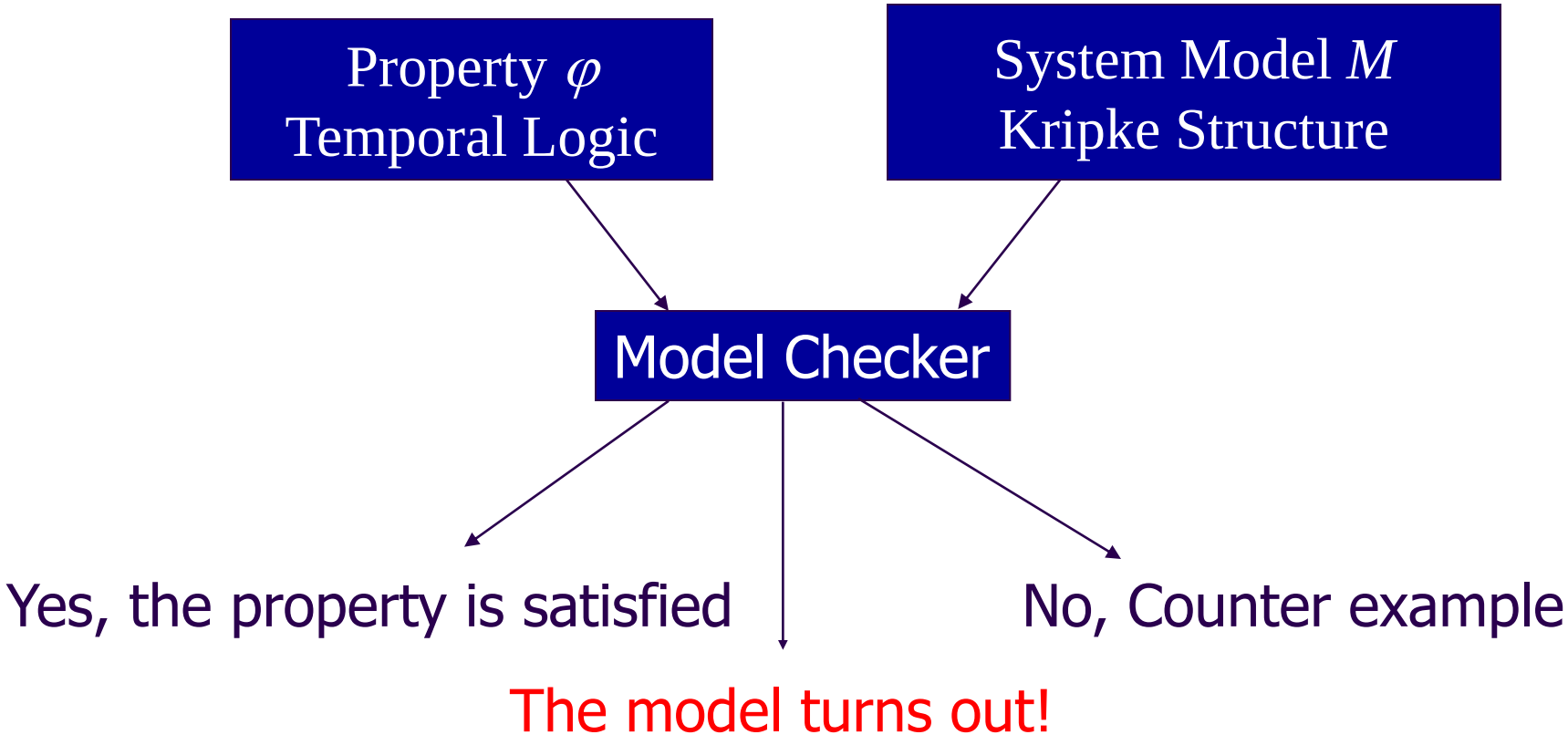
- Model checking: Developed independently by **Clarke, Emerson, and Sistla** and by **Queille and Sifakis** in early 1980's
  - It consists of three parts:
    1. A framework for modeling software (some kind of specification language)
    2. A specification language for describing the properties to be verified
    3. A verification method for establishing if the software description satisfies the specification

Model Checking Approach

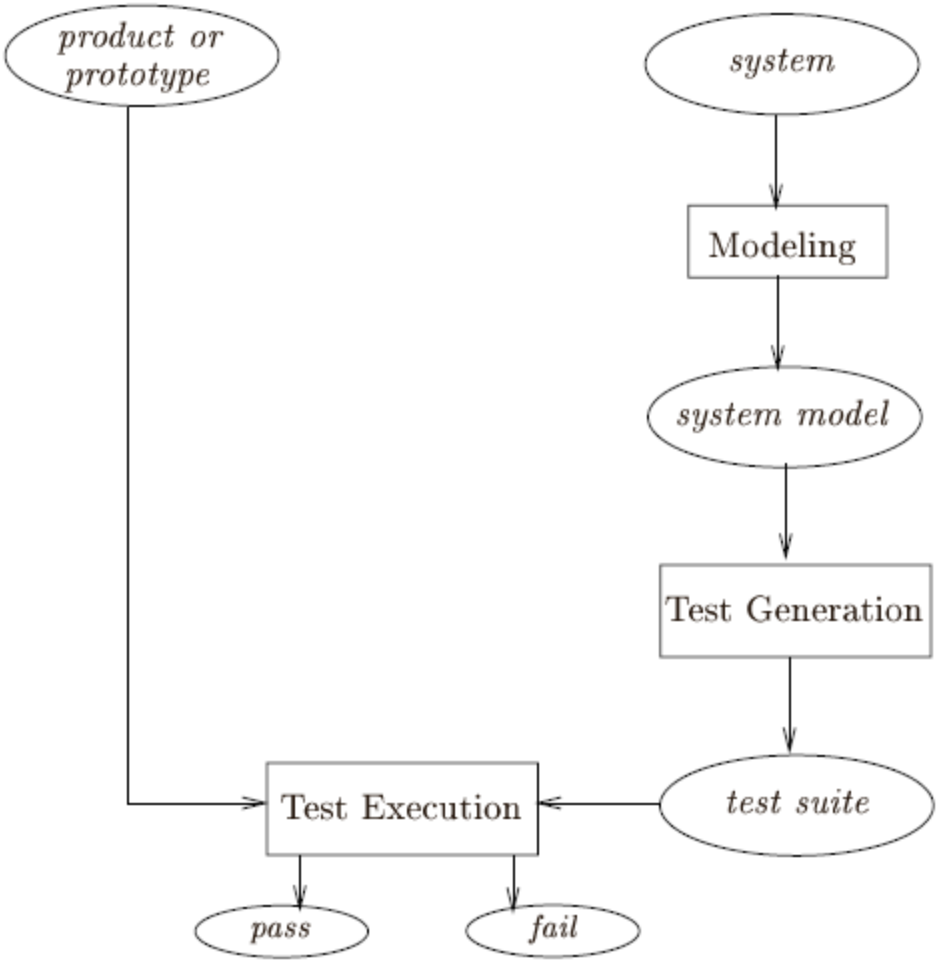


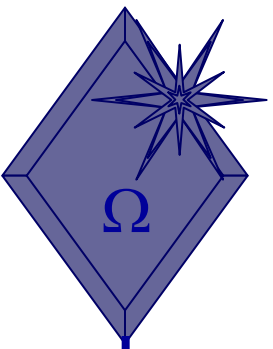


Model Checking



Testing Activity

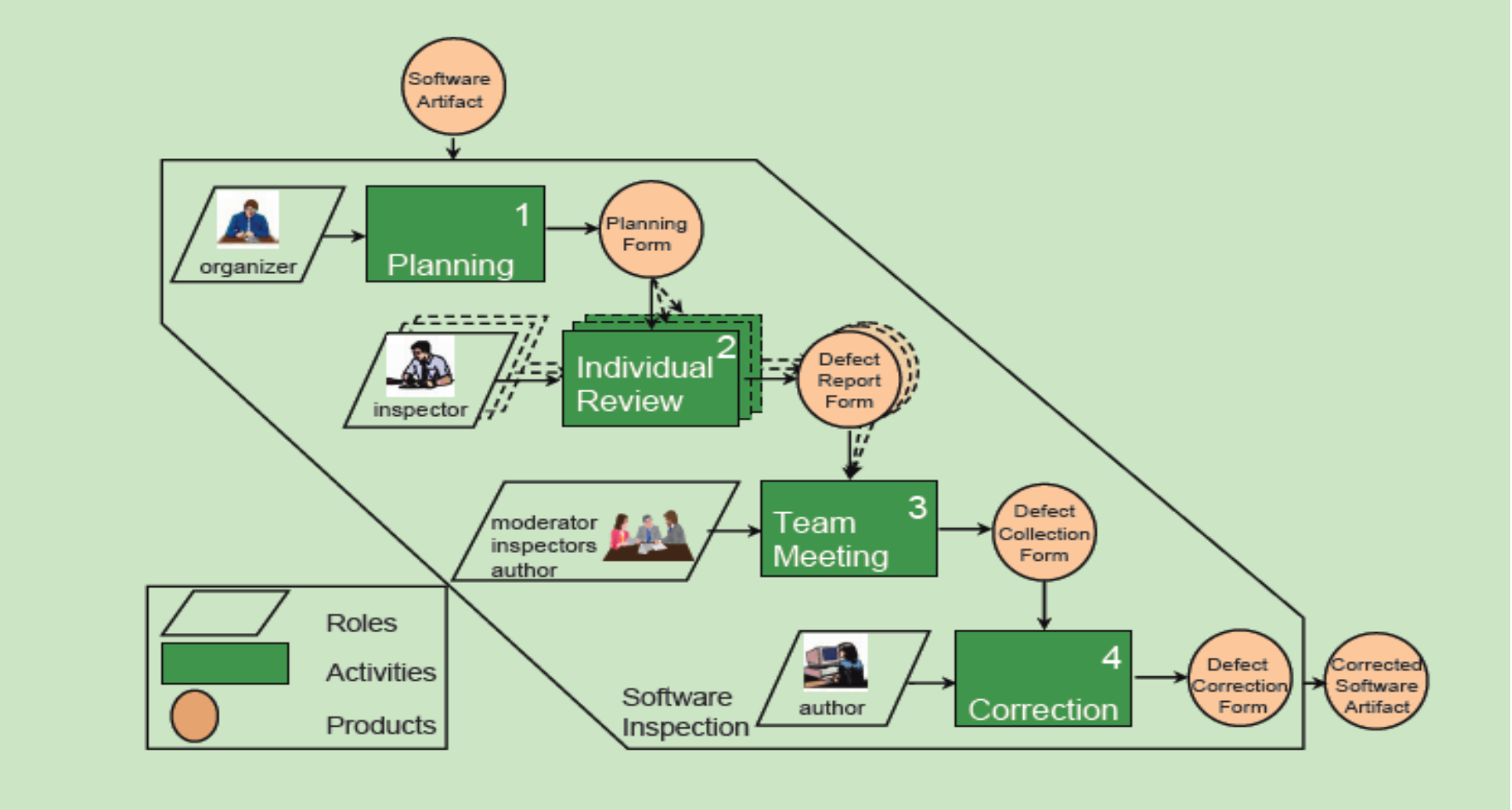


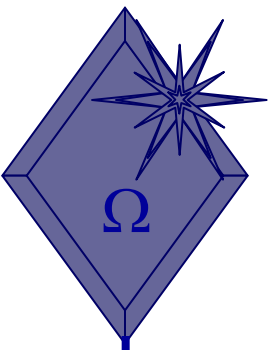


Overview

- ✓ Preliminary Notions
- ✓ Verification Approaches
- Software Inspection

Software Inspection Activities

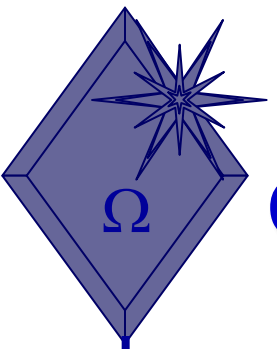




What are Inspections?

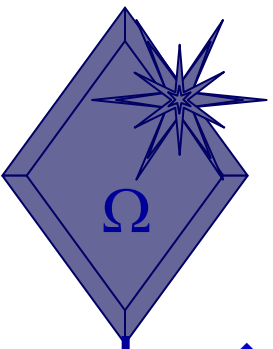
An inspection is a structured peer review:

That Provides:	To:
Defect information	Author
Other perspectives on work	Author
Accurate project status	Product Management
Generic defects (trends)	Management



Candidates for Reviews and Inspections

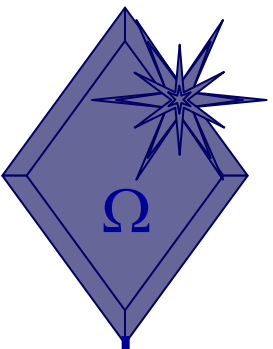
- Strategic Plans
- Contracts
- Requirements
- High Level Designs
- Detailed Designs
- Architectural Documentation
- Code
- Test Plans
- Test Designs
- User Documentation
- Project Plans, etc.



Benefits

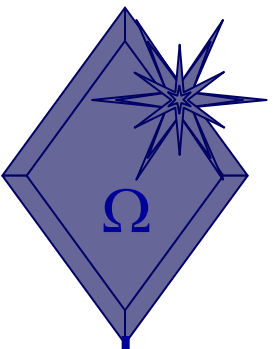
- Inspections provide a powerful way to:
  - Detect defects early in the development cycle
  - Prevent the migration of defects to later phases
  - Improve the quality and productivity of the development and test process
  - Reduce cost and cycle time
  - Reduce maintenance effort

Review early and often



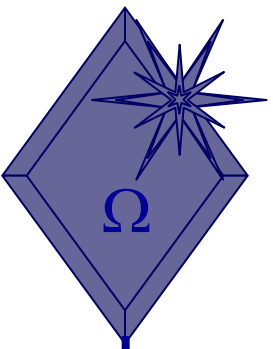
Software Inspections (Static Verification)

- Inspections do not require execution of a system so may be used before implementation
- Not just program source code
  - May be applied to any representation of the system (requirements, design,configuration data, test data, etc.)
- Have been shown to be an effective technique for discovering program errors



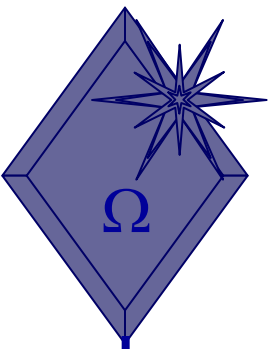
Inspection Success

- Many different defects may be discovered in a single inspection. In testing, one defect, may **mask** another, so several executions are required
- Incomplete versions can be inspected
- Other quality attributes such as coding standards, maintainability, portability can also be checked
- The reviewers reuse domain and programming knowledge so they are likely to have seen the types of error that commonly arise



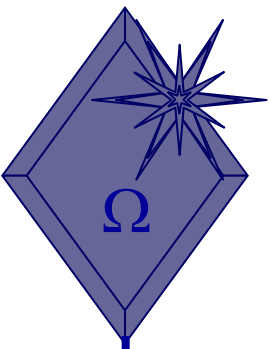
Inspections and Testing

- Complementary and not opposing verification techniques
- Both should be used during the V & V process
- Inspections **cannot** check non-functional characteristics such as performance, usability, etc.



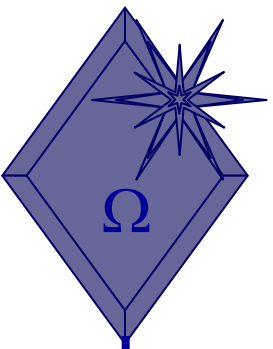
Program Inspections

- A systematic approach to document reviews
- Intended explicitly for defect detection (not correction)
- Defects may be logical errors, anomalies in the code that might indicate an erroneous condition (e.g. an uninitialised variable) or non-compliance with standards



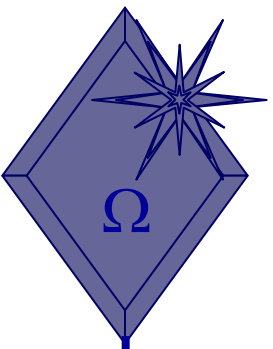
Inspection Pre-conditions

- A precise specification must be available
- Syntactically correct code or other system representations must be available
- An error checklist should be prepared
- Management must accept that inspection will increase costs early in the software process
- Management should not use inspections for staff appraisal i.e., finding out who makes mistakes



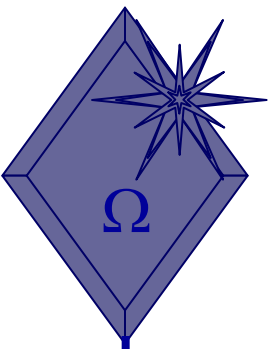
Automated Static Analysis

- Static analysers are software tools for source text processing (e.g., GrammaTech, Coverity Code Advisor, Klocwork, FindBugs, etc.)
- They parse the program text and try to discover potentially erroneous conditions and bring these to the attention of the V & V team
- They are very effective as an aid to inspections



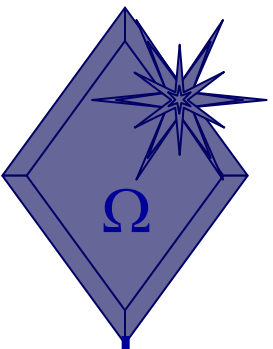
Stages of Static Analysis

- Control flow analysis. Checks for loops with multiple exit or entry points, finds unreachable code, etc.
- Data use analysis. Detects uninitialised variables, variables written twice without an intervening assignment, variables which are declared but never used, etc.
- Interface analysis. Checks the consistency of routine and procedure declarations and their use



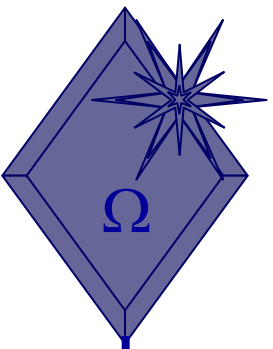
Stages of Static Analysis

- Information flow analysis. Identifies the dependencies of output variables. Does not detect anomalies itself but highlights information for code inspection or review
- Path analysis. Identifies paths through the program and sets out the statements executed in that path. Again, potentially useful in the review process



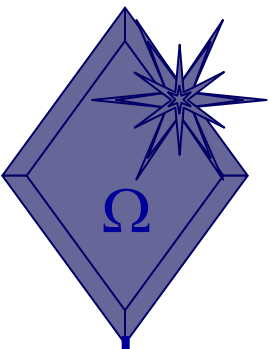
Use of Static Analysis

- Particularly valuable when a language such as C is used which has weak typing and hence many errors are undetected by the compiler
- Less cost-effective for languages like Java that have strong type checking and can therefore detect many errors during compilation



Key Points

- Verification and validation are not the same thing
  - Verification shows conformance with specification
  - Validation shows that the program meets the customer's needs
- Static verification techniques involve examination and analysis of the program for error detection



Key Points

- Program inspections are very effective in discovering errors
- Program code in inspections is systematically checked by a small team to locate software faults
- Static analysis tools can discover program anomalies which may be an indication of faults in the code