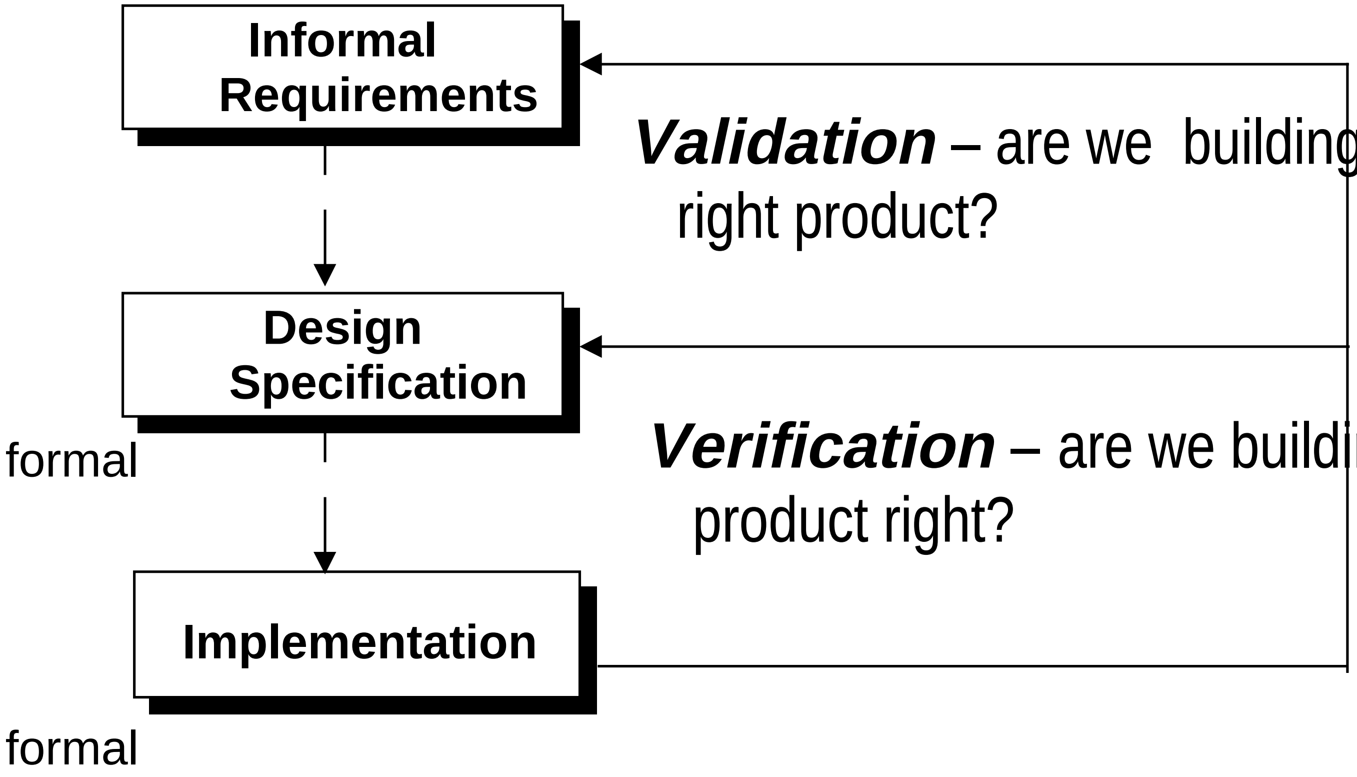
LIFECYCLE VERIFICATION & VALIDATION



Validation – are we building the right product?

Verification – are we building the product right?

“Testing” Principles

- Testing must be an inherent component of the software process
  - should not be a separate phase after integration and before maintenance
- Execution-based testing
  - execution of code (primarily the implementation)
- Nonexecution-based testing
  - reviews and static analysis of (non)executable software descriptions
- Verification: comparing to specification
- Validation: checking against user needs
- Software Quality Assurance (SQA)
  - SQA group is responsible for ensuring that all phases are carried out as dictated and that product is "correct"
  - Quality assurance applies to every aspect of the software process
  - SQA group should be managerially independent

Testing

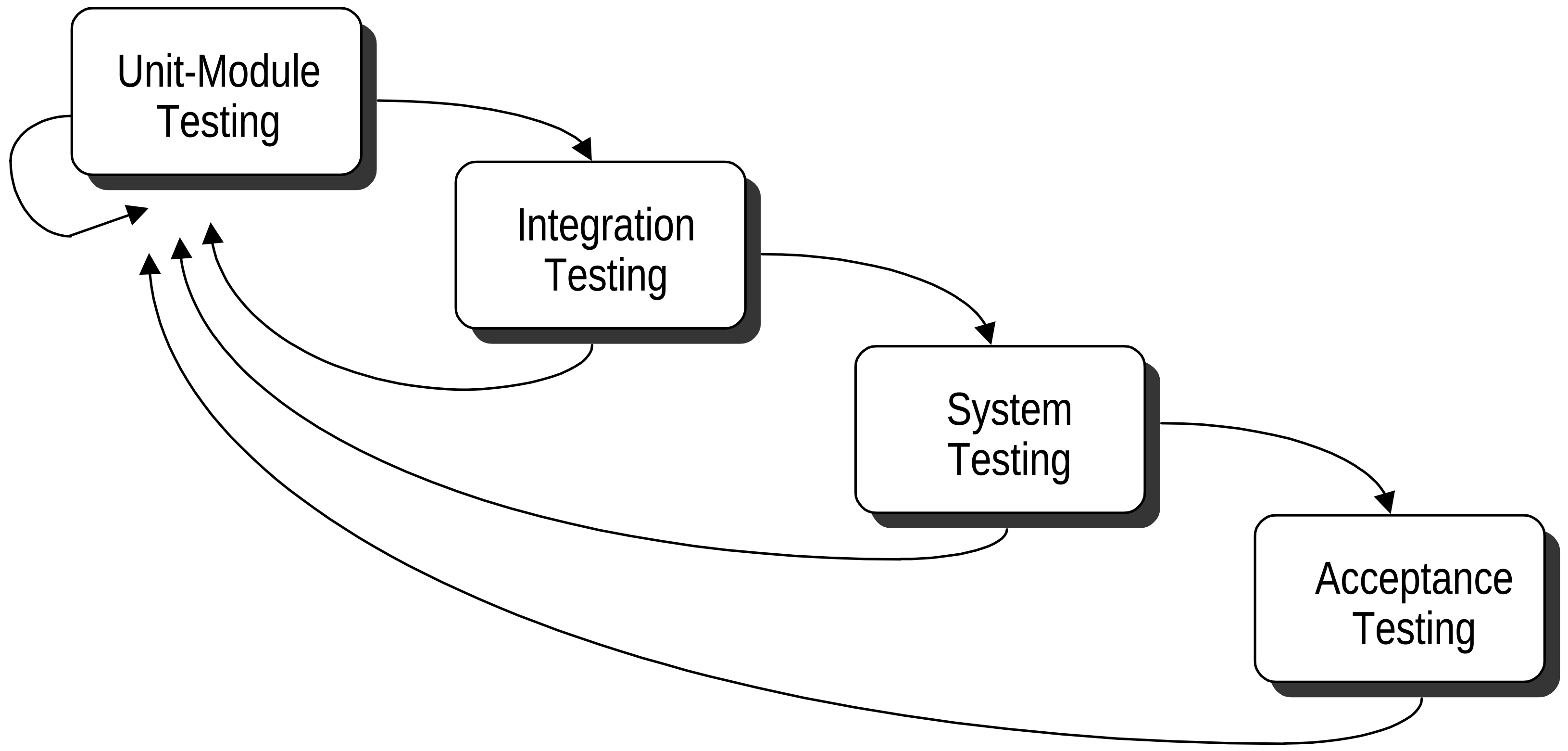
- **Testing is the process of inferring behavioral properties of a product on the basis of execution in a known environment with selected inputs and checking results with a test oracle**
- **What properties should be tested?**
  - **utility**
  - **reliability**
  - **functional correctness**
  - performance
  - robustness
- **Who should test?**
  - **testing is destructive**
  - **testing dichotomy: success is failure and failure is success**
- **When does testing stop?**
  - **only after retirement**

Testing Phases

- ***Unit/Module Testing***
  - **testing of a unit or module (encapsulation of units) comparing it with requirements & make ready for integration**
- ***Integration Testing***
  - **systematic combination and testing of software components to insure consistency of component interfaces**
- ***System Testing***
  - **testing an integrated software system comparing it with software system requirements (in development environment)**
- ***Acceptance Testing***
  - **testing an integrated hardware and software system (in target environment, with customers data)**
  - **also called "*alpha testing*"**
  - **after acceptance "*beta testing*" with a selected group of customers start**

Testing Phases - 2

- *Regression Testing*
  - testing a modified system to ensure unmodified part has not regressed



Test Documentation

- **Test Plans**
  - must be developed during all development phases
  - test cases for phase-specific decisions
  - important to have testing objectives
  - important to avoid overconfidence
  - plans can be reused for regression testing
- **Test Histories**
  - must be maintained during all testing phases
  - error logs
  - change reports
  - documentation for later reference
  - important for process improvement

Test Plan/History Documentation

- **Test Plan Objective**
  - test plan type
  - system/component being tested
  - criteria/requirements
- **Testing Process: how to accomplish this test plan**
  - order of execution, process description
- **Test Cases and Test Histories**
  - ID: purpose
  - environment/procedure (drivers, stubs, state)
  - test data input, expected output
  - actual output, problems revealed, modifications
- **Justification: how the test case set satisfies the objective**
- **Test Plan Status: the current status of this testing process**

Qualitity Assessment

- **There is a critical need to produce high quality software**
  - **increasing safety-critical applications**
  - **required qualities are widely-varied**
- **Quality assessment must be formalized**
  - **facilitated with formal specifications**
  - **specification, design and verification technologies have not been shown to be sufficient**
  - ***Testing* is a viable approach, but it must be done systematically**

V&V *not* restricted to

Implementation and Integration

Quality Assessment must permeate the process

- **Quality assessment (testing, verification and validation) should occur at each phase**

to start:

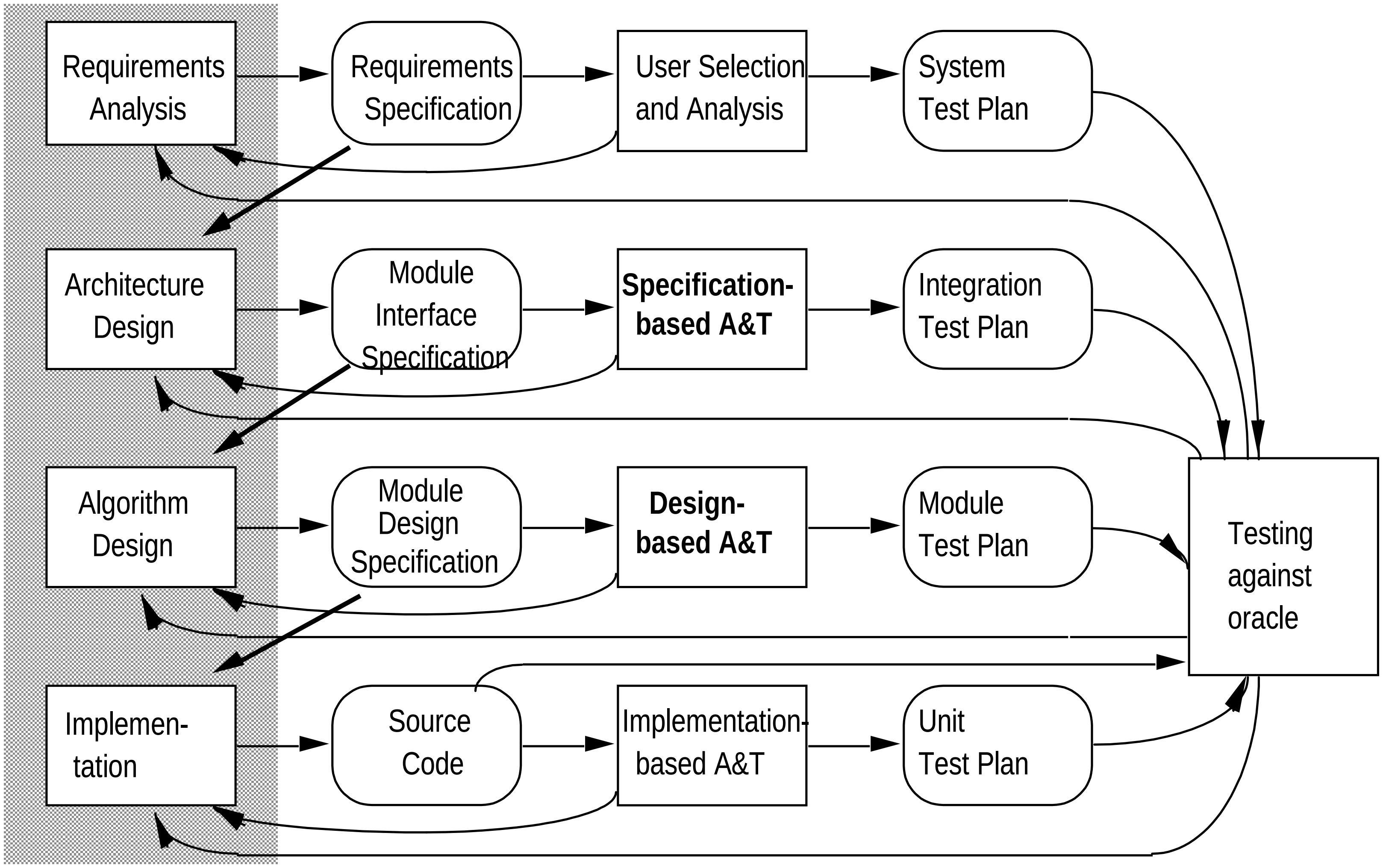
- – requirements validated against user needs
- – requirements shown internally consistent
- – requirements assured of high quality

for each phase:

- – validate current phase against user needs
- – use information from previous phase to verify current phase

- **Test plans should begin with requirements and be reviewed and refined with each phase**
  - – test plans should be executed as early as possible to further facilitate early error detection

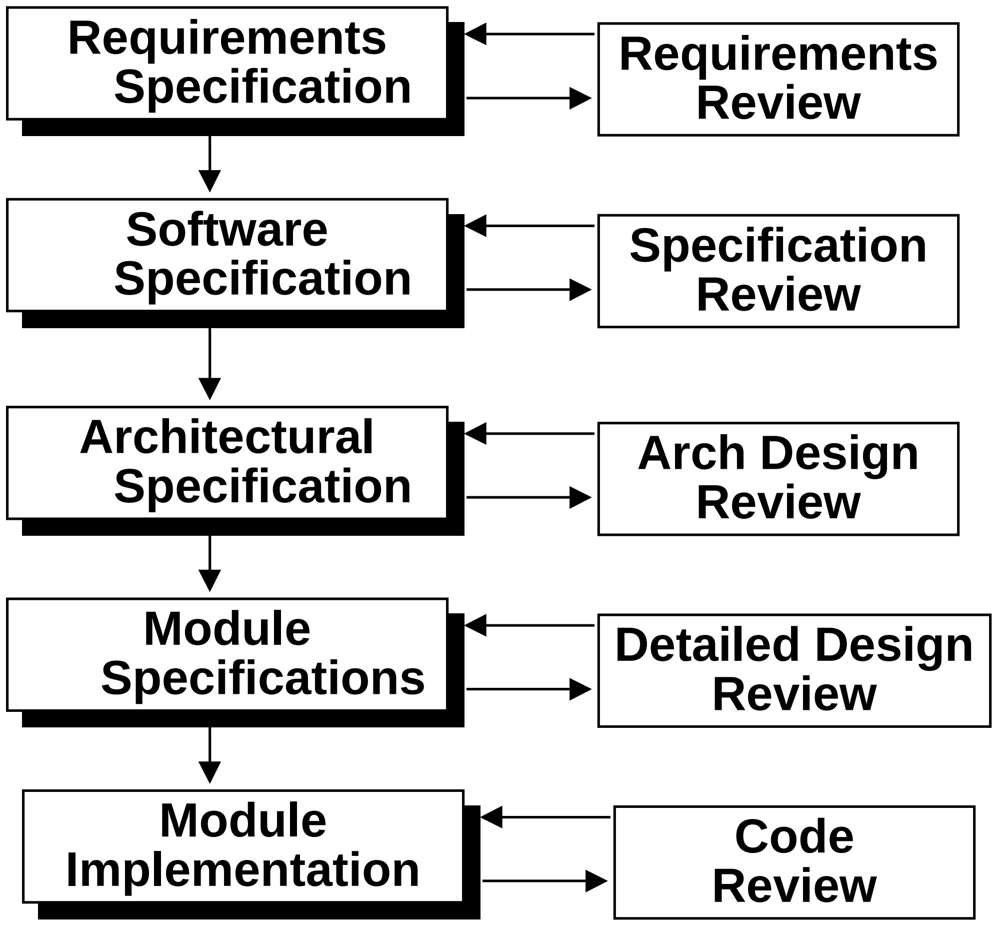
Test Planning and Testing



Lifecycle Reviews: Goals and Objectives

- Review all lifecycle artifacts
- Discover “all” defects currently present in the product under development (as early as possible)
- Verify that inspected specification conforms with requirements or detect cases of non-conformance
- Detect defects in software specification
- Detect defects in a specification's representation
- Evaluate techniques and tools
- Measure development process
- Measure product quality
- Feedback for specifiers to improve
- Feedforward for process and quality control

Lifecycle Reviews



find defects in requirements

find defects in requirements

find defects in design structure and decomposition

find defects in module specifications

find defects in implementation

Lifecycle Reviews: Products

- **Software problem reports**
- **Software change reports**
- **Error Classification**
  - ***inconsistency* – specification won't work and/or doesn't meet requirements**
  - ***inefficiency* – specification imposes barrier to efficient programming or system use**
  - ***ambiguity* – specification admits varying interpretations**
  - ***inflexibility* – specification does not accomodate change well**

➪ **Higher quality software**

Walkthroughs vs. Inspections

- **Participants**
  - specification rep – development rep
  - client rep – SQA rep
  - ⇨typically more participants for inspections
- ***Walkthroughs* are a two-step process**
  1. preparation: reviewers read documents
  2. group analysis: chaired by SQA rep for objectivity
- ***Inspections* [Fagan,1976] are a five-step process**
  1. overview: tutorial presentation of software to be inspected
  2. preparation: reviewers read documents
     includes a checklist of questions to aid in finding flaws
  3. group inspection: round-table discussion to find and document defects
  4. rework: describe and correct defects
  5. follow-up: ensure every identified problem solved

Specification Review Process

❶ Identify desired properties

❷ Make representation reviewable

❸ Separate types of reviews desired

❹ Classify reviewers – give participants roles
Moderator in charge

❺ Distribute a questionnaire/checklist

❻ Conduct review

❼ Resolve problems and follow-up

Can be applied to any software lifecycle artifact

❶ Desired Specification Properties

- Well structured (*wrt* principles such as information hiding)
- Standardized representation
- Simple
- Efficient
- Flexible (*wrt* requirements changes)
- Practical (not overly general nor specific)
- Implementable (*wrt* resources)
- Verifiable (*wrt* requirements)

❷ Reviewable Representation

- **Make assumptions explicit**
  - **capabilities of operations**
  - **types of parameters**
  - **side effects**
  - **timing**
  - **handling of undesired events**
- **Include redundant information**
  - **assumptions specifiers take as invariant**
  - **usage that specifiers assume will not occur**
- **Organize document for review**

❸Types of Reviews and ❹ Reviewer Classification

- **Types of Reviews**
  - **Assumption validity: are they all correct?**
  - **Assumption sufficiency: are they all specified?**
  - **Assumption/Functions consistency**
  - **Requirements/Functions adequacy**
- **Classification of Reviewers**
  - **Potential Users: capable of assessing satisfaction of user requirements**
  - **Designers/Coders: capable of evaluating specification representation and method**
  - **Testers: capable of assessing verifiability and validating**
  - **Specialists: capable of assessing performance and feasibility**
  - **Problem solvers**
- **Moderator in charge**
  - **trained and approved, drives the inspection, manages the group**

❺ Distribute Questionnaire

- Describe properties for which the reviewer should check
- Sections of the abstract interface should be studied
  - Questions to be completed by reviewer
- Make reviewers take an active stand
  - Seek positive feedback as well as negative
- Include a common checklist of potential faults
  - Lists of fault types found in recent inspections are good aids (enable team members concentrate on areas where most faults have occured)

❻ Conduct the Review

- Conduct the sessions one-on-one
- Present a brief overview of the component to be reviewed
  - show the overall scheme
  - describe this component's location in the scheme
- Reviewers go and do their own thing
- Specifiers read completed questionnaires and meet with reviewers

❼ Resolve and Follow-up

- **Reviewers identify specification defects**
- **Developers isolate fault in specification**
- **Developers repair specification**
- **Follow-up to review repairs**
  - **Moderator must ensure that every single issue raised has been satisfactorily resolved**
  - **All fixes must be checked to ensure that no new faults have been introduced**
  - **If more than 5 % of the material inspected has been reworked, the team reconvenes for a 100 % reinspection**

Cleanroom Software Development
[Mills et al., 1987]

- **The “ideal” review process**
- **Based on static verification to ensure error-free development**
  - **defects should be avoided rather than detected and corrected**
  - **defects avoided by developing in an ultra-clean environment (derived by analogy with semiconductor fabrication units)**
  - **structured inspections augmented with formal correctness arguments**
- **Software components are formally specified and verified *instead of* usual development and unit/module testing**

Cleanroom Software Development - 2

❶ **Formal specification:**

– **Software to be developed is formally specified**

❷ **Incremental development:**

– **Software is partitioned into increments which are developed seperately using the Cleanroom approach**

❸ **Structured programming:**

– **Only a limited number of control and data abstraction constructs are used. Stepwise refinement of the specification**

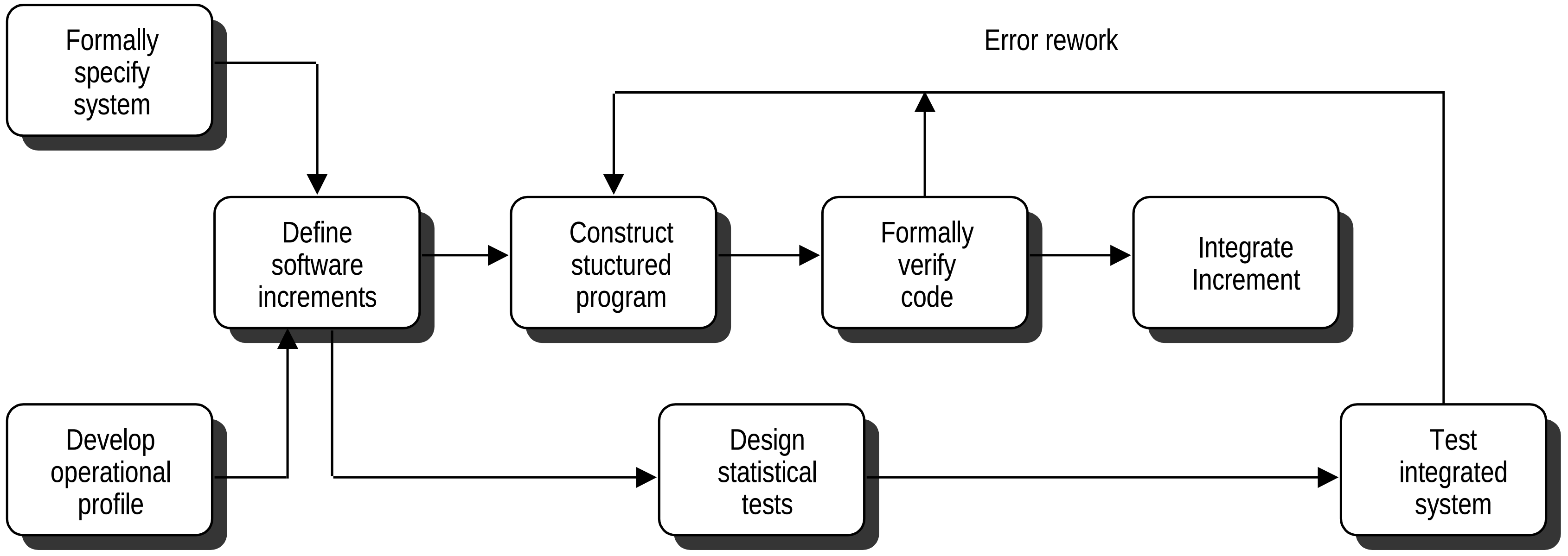
❹ **Static verification:**

– **Developed software commponents are not tested but statically verified using mathematically based correctness arguments**

❺ **Statistical testing:**

– **Integrated software is tested statistically to determine its reliability**

Cleanroom Software Development - 3



Cleanroom Software Development - 4

- **Three Cleanroom teams**
  - ***specification team*: developing and maintaining the system specification**
  - ***development team*: developing and verifying the software. Software is not executed but formal approach to verification (e.g. code inspection) is used**
  - ***certification team*: developing a set of statistical tests based on the formal specification**
- **Cleanroom approach purported to be more effective than "traditional" approach**
  - **experimentation may not have compared to best alternatives or used representative developers**
  - **definitely lends credence to development using formal specification and verification**

V&V of specific qualities

Discussion

- How would you evaluate the following qualities?
  - usability
  - reliability
  - robustness
  - performance
  - correctness
  - portability