Repetition

Software Testing: INF3121 / INF4121

04.05.2017 / Yulai de Meer Fjeld

Course Syllabus

I. Fundamentals of testing

II. Testing throughout the software lifecycle

III. Static test techniques

IV. Test design

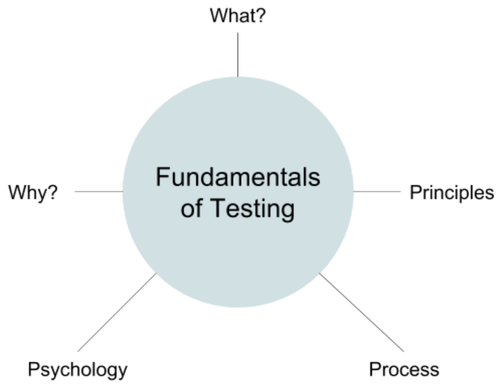
V. Test management

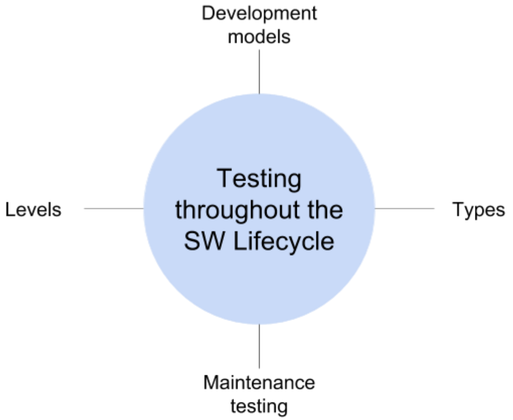
VI. Tool support for testing

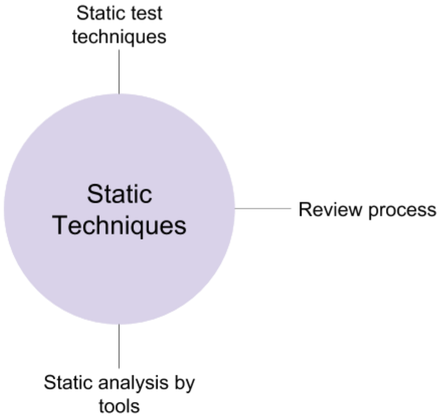
VII. Usability and accessibility testing

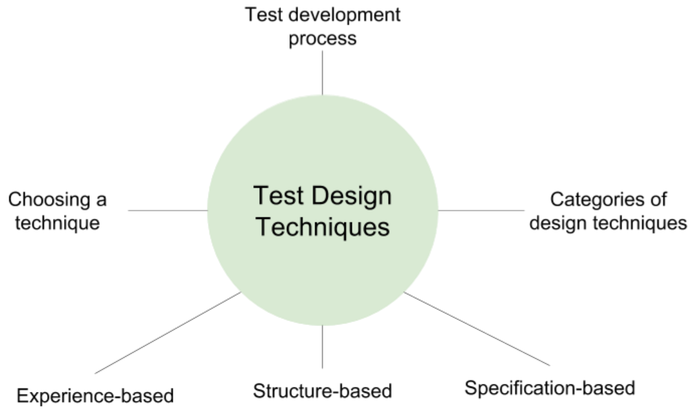
VIII. Exploratory testing

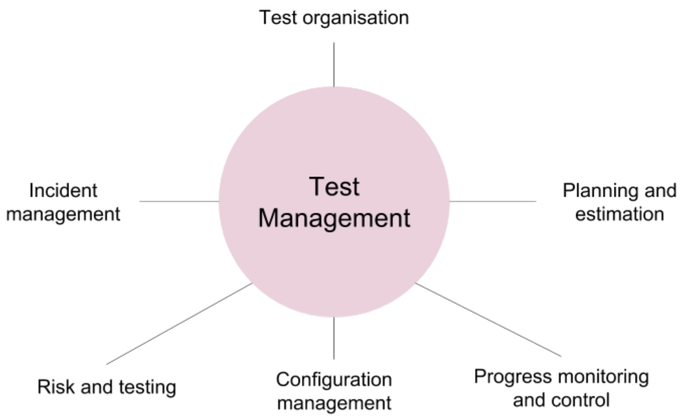
An Overview

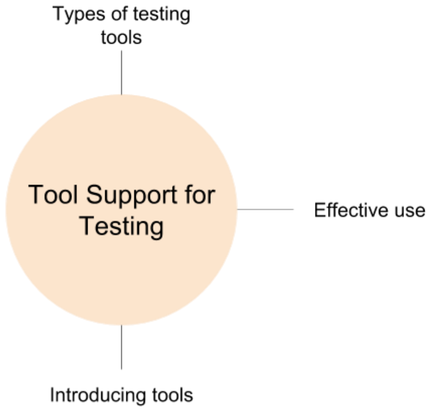

What?
Why?
Fundamentals of Testing
Principles
Psychology
Process

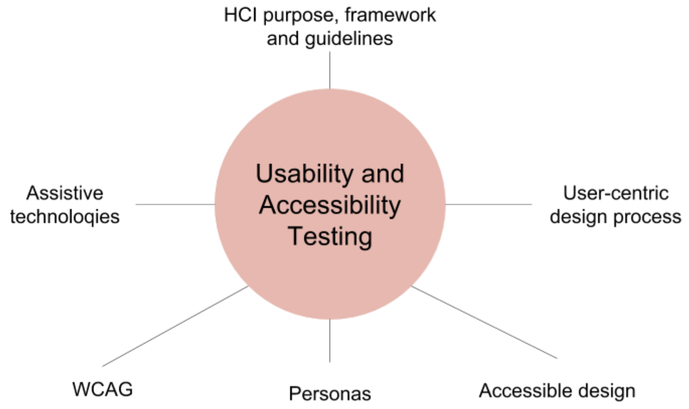

Development models
Levels
Testing throughout the SW Lifecycle
Types
Maintenance testing

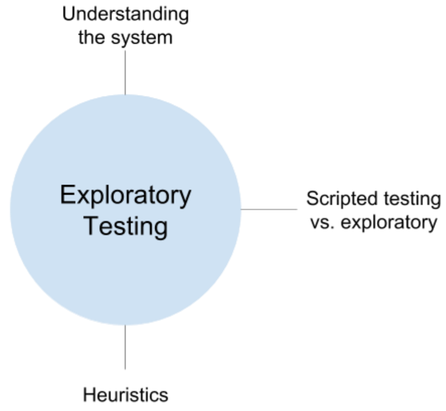

Static test techniques
Static Techniques
Review process
Static analysis by tools


Test development process
Choosing a technique
Test Design Techniques
Categories of design techniques
Experience-based
Structure-based
Specification-based


Test organisation
Incident management
Test Management
Planning and estimation
Risk and testing
Configuration management
Progress monitoring and control


Types of testing tools
Tool Support for Testing
Effective use
Introducing tools


HCI purpose, framework and guidelines
Assistive technologies
Usability and Accessibility Testing
User-centric design process
WCAG
Personas
Accessible design


Understanding the system
Exploratory Testing
Scripted testing vs. exploratory
Heuristics

I. Fundamentals of Testing

Fundamentals of Testing

Why is testing necessary?

- Software systems are everywhere → Important that they function correctly
  - Loss of money | business reputation | fatal consequences
- Reduce risk of problems
- Validation and verification
- Learning more about the system

What is testing?

- Process consisting of all lifecycle activities (static and dynamic) …
- Concerned with planning, preparation, evaluation of software and work products …
- Determine that they satisfy requirements, demonstrate fit for purpose, detect defects

Test Principles

P1: Testing shows presence of defects

- Testing can show that defects are present, but cannot prove there are no defects.
- Testing reduces the probability of undiscovered defects remaining in the software; but even if no defects are found, this is not a proof of correctness.

P2: Exhaustive testing is impossible

- Testing everything is not feasible. We use risks and priorities to focus on test effort.

P3: Early testing

- Testing should start as soon as possible in the development life-cycle and should be focused on defined objectives.

P4: Defect clustering

- A small number of modules contain most of the defects discovered during pre-release testing.

P5: Pesticide paradox

- If the same set of tests will be repeated over and over, it will no longer find new bugs.

P6: Testing is context dependent

- I.e. safety-critical SW is tested differently from an e-commerce site.

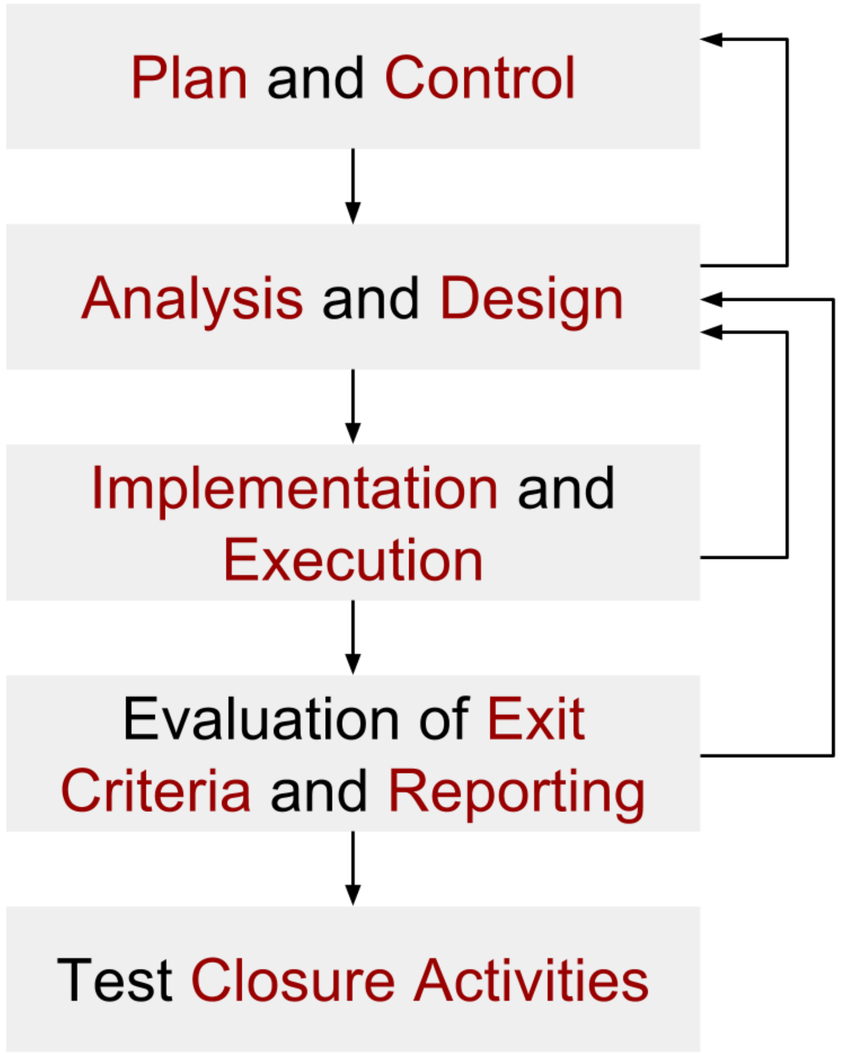
P7: Absence-of-errors fallacy

- Finding and fixing defects does not help if the SW system is unstable or does not meet the user's expectations.

Fundamental Test Process

Plan and Control	What? | How? | When? | By whom? Scope, objectives, risk, test levels and types, documentation
Analysis and Design	Review | Analyse | Design Requirements, interfaces, test cases and conditions, data
Implementation and Execution	Make and run Group tests into scripts, prioritise, write automated tests Run tests, report incidents, repeat test activities
Evaluation of Exit Criteria and Reporting	Assess test results and communicate findings Compare to defined objectives, summarise, more testing?
Test Closure Activities	Archive deliverables Software / source code, tests and results, documentation

Fundamental Test Process

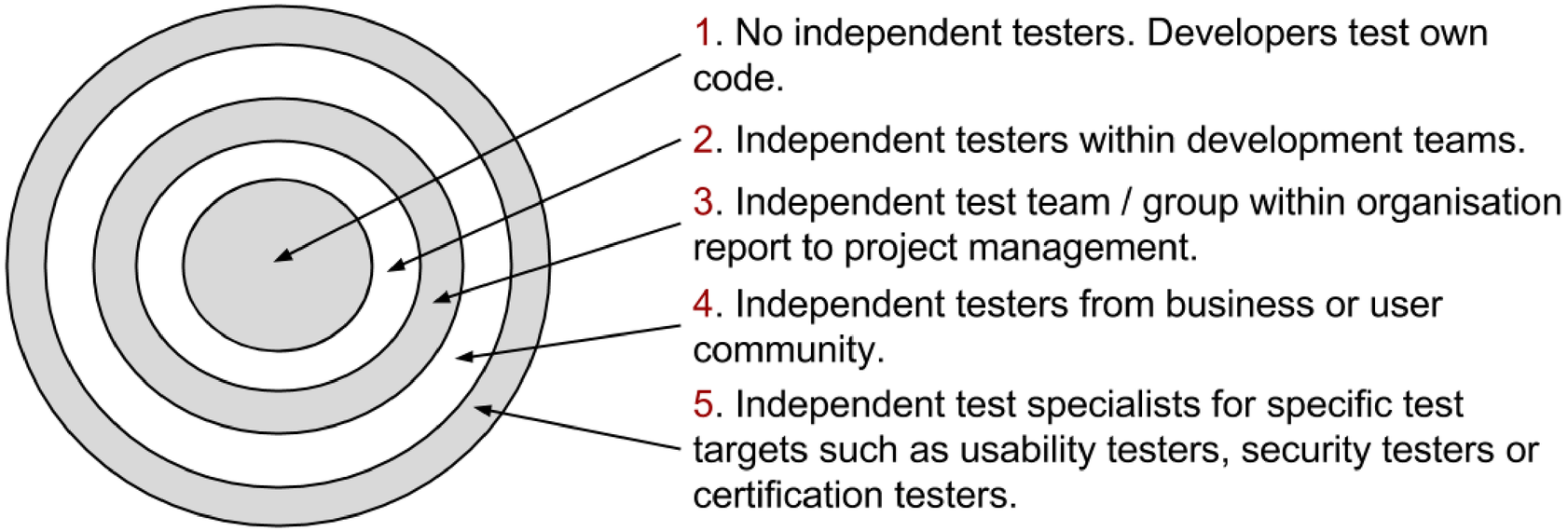


Psychology of Testing

Psychology of testing

Independence in testing

Often more effective at finding defects



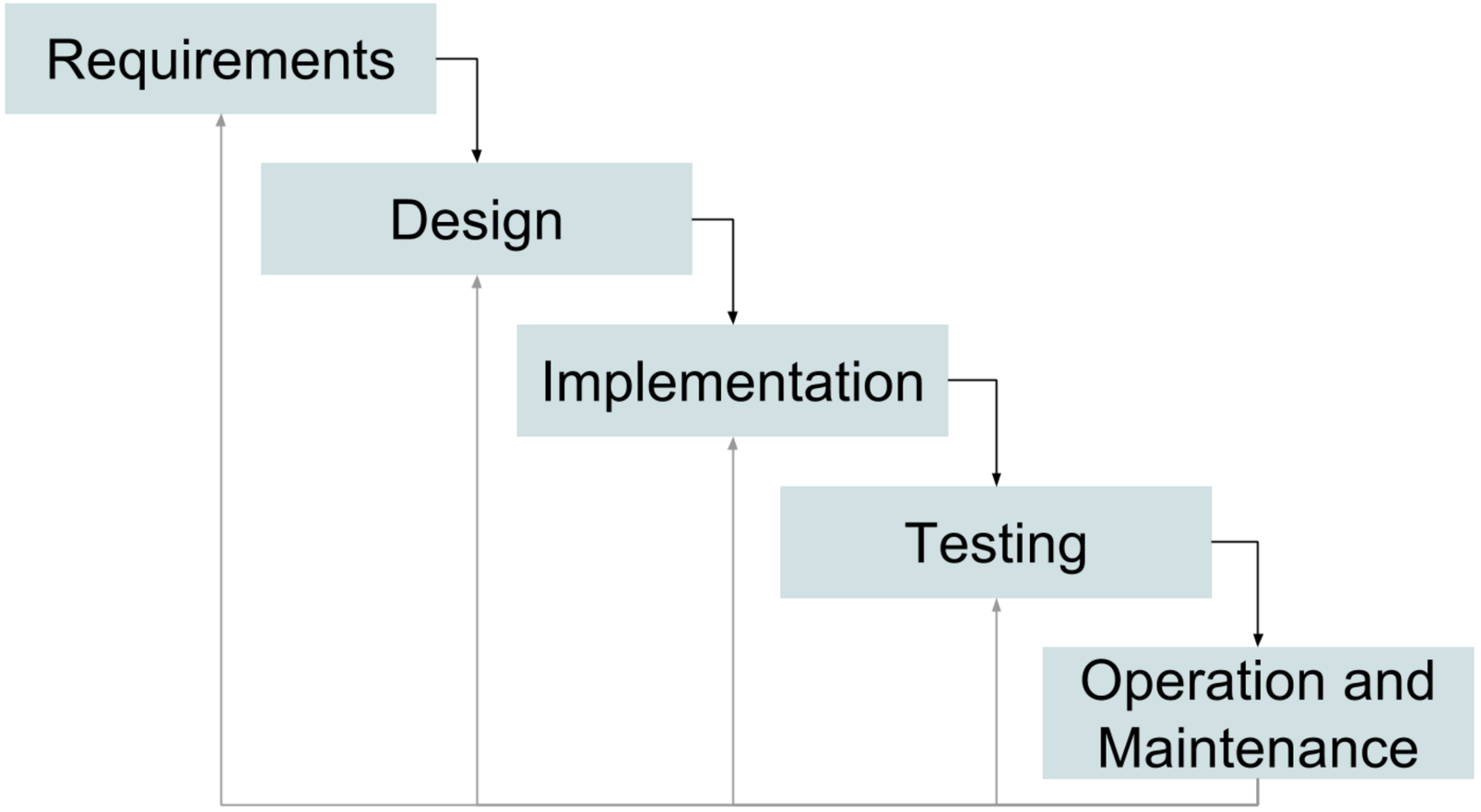
Characteristics for good testing / testers

Curiosity | Professional pessimism | Detail-oriented | Constructive | Good communication

II. Testing throughout the SW Lifecycle

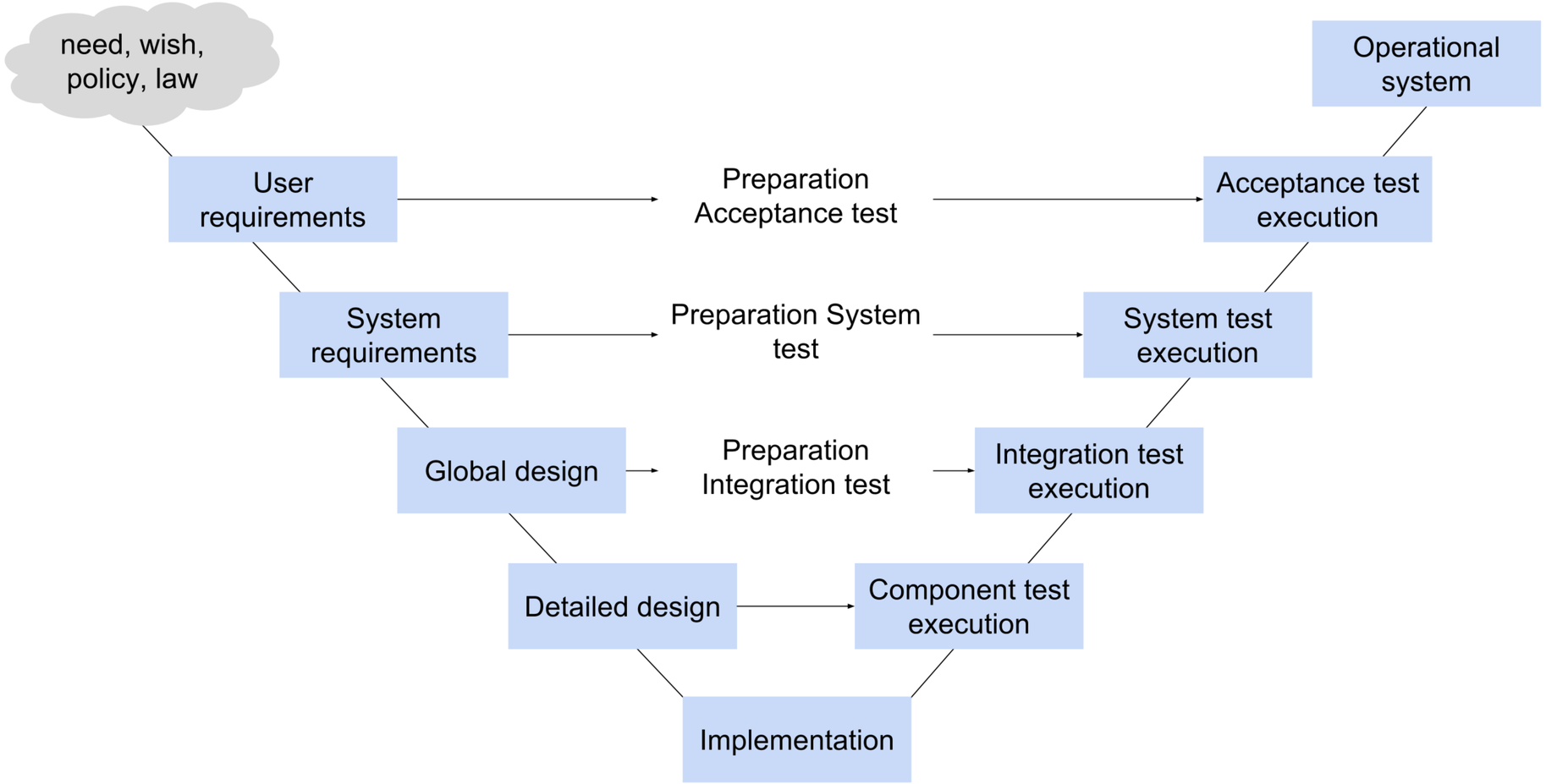
Software Development Models

Waterfall model



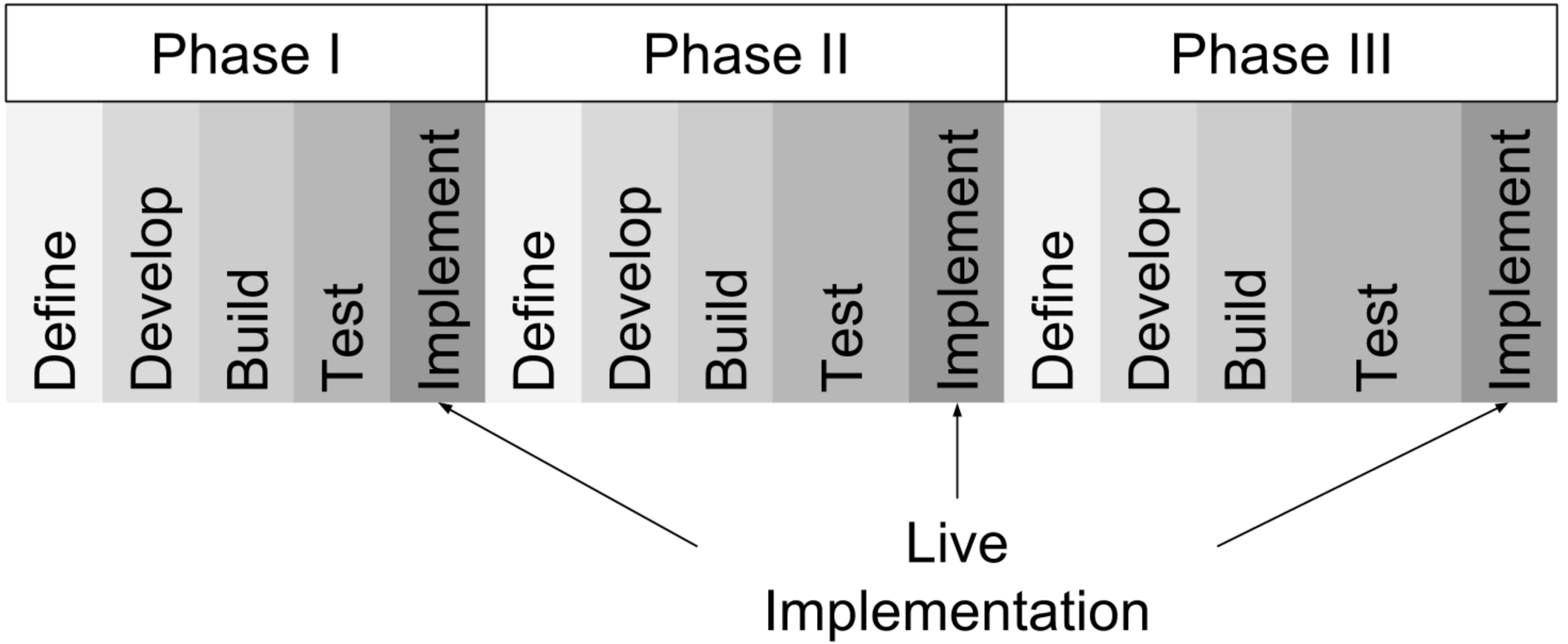
Software Development Models

Sequential development model (V-model)



Software Development Models

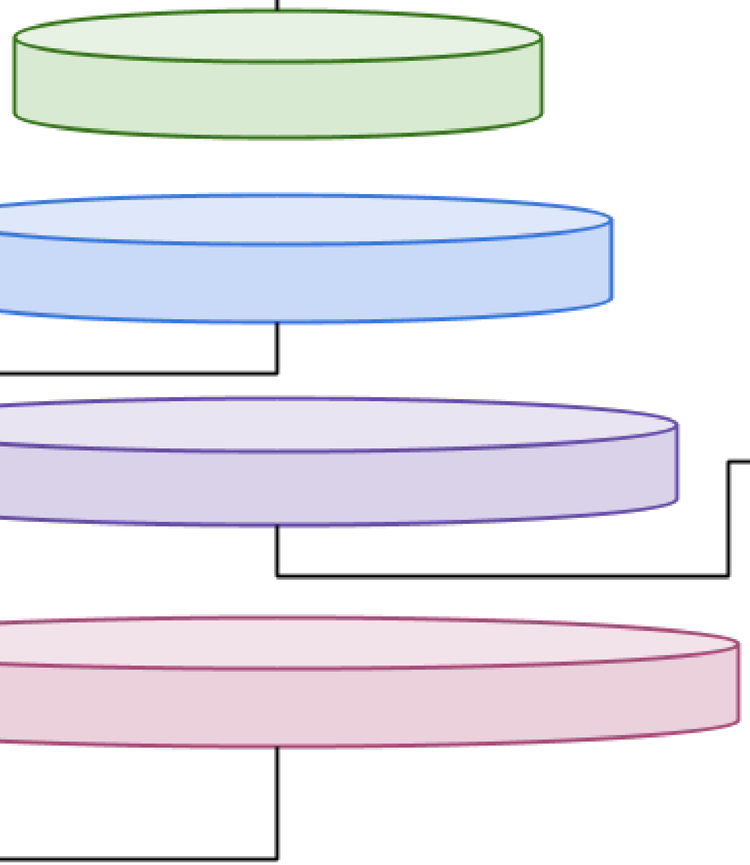
Iterative-incremental development model



Test Levels

- **Acceptance**

Is the responsibility of the customer - in general. The goal is to gain confidence in the system; especially in its non-functional characteristics.



- **System**

The behaviour of the whole system as defined by the scope of the project.

- **Integration**

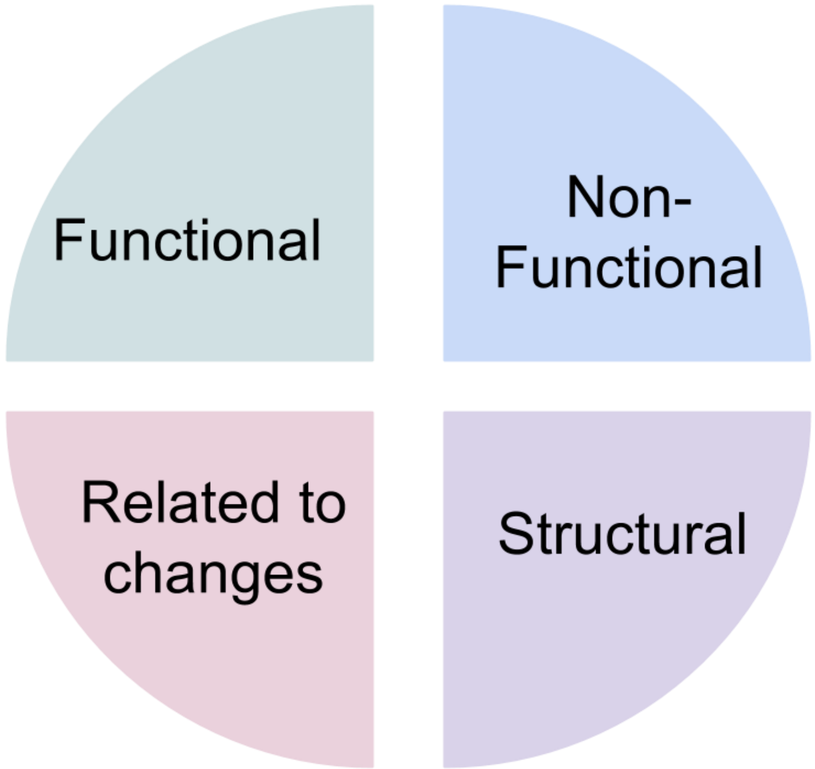
Interface between components; interactions with other systems (OS, HW, etc.)

- **Unit**

Any module, program, object separately testable

Test Types

"What" the system does
- Suitability
- Interoperability
- Security
- Accuracy
- Compliance



"How" the system works
- **Performance**, **Load**, **Stress**
- **Reliability** (robust, fault tolerant, recoverable)
- **Usability** (understand, learn, operate, like)
- **Efficiency** (time, resource utilisation)
- **Maintainability** (analyse, change, stabilise, test)
- **Portability** (adapt, install, co-exist, replace)

After changes
- Confirmation testing
- Regression testing

Inspection of code, modules
- Code coverage

III. Static Test Techniques

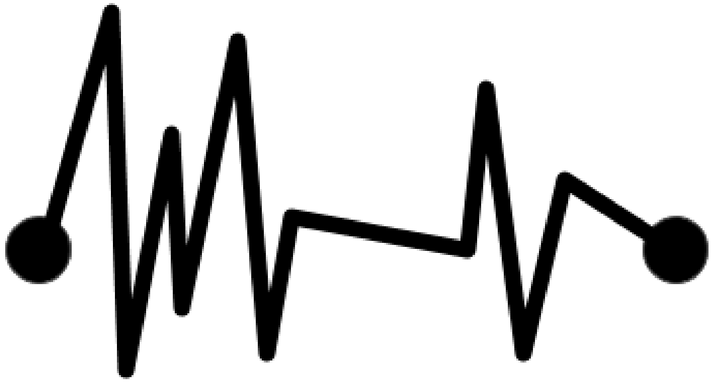
Static Test Techniques

Know the difference between static and dynamic

Static testing

- Examination of code without executing it
- Can be applied to other work products

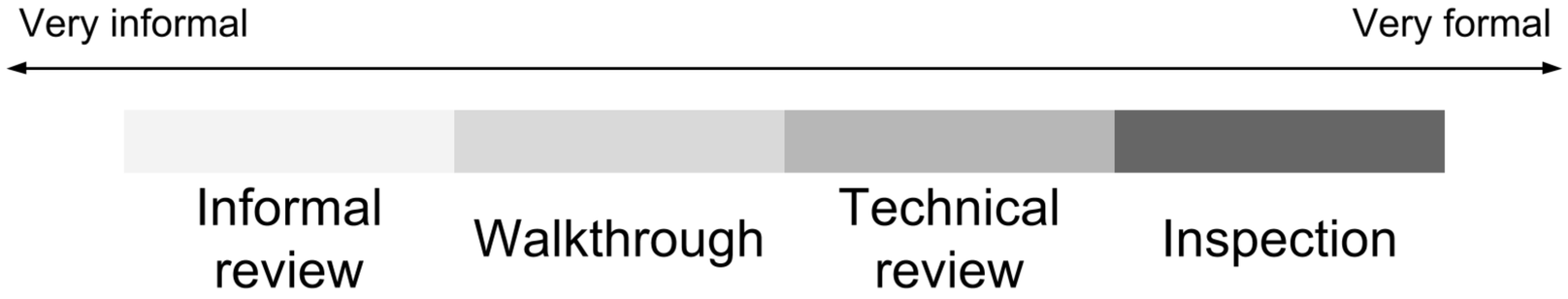
Dynamic testing



- Requires source code to be executed

Review Process

Different types of reviews



Roles and responsibilities

Manager

- Decides on execution of reviews
- Allocates time in project schedules
- Determines if review objectives are met

Moderator

- Leads the review
- Plans the review
- Runs the meeting
- Follow-up after meeting
- Mediates between various points of view

Author

- Writer of the documents being reviewed, or
- Responsible for the documents being reviewed

Reviewers

- Individuals with specific technical or business background
- Identify and describe the findings in product under review

Scribe

- Documents the entire review meeting
- Issues, problems, open points that have been identified

Static Test Techniques

Static analysis by tools

- Analysis of source code and generated output
  - Control flow, data flow, HTML, XML
- Improve quality of code
- Typical defects discovered
  - Referencing a variable with an undefined value
  - Variables that are never used
  - Syntax violations and breach of coding standards
  - Deadlocks / unreachable code
  - Inconsistent interface between modules / components
  - Security vulnerabilities

IV. Test Design

Test Development Process

Understand what and why we are testing

Process can vary from very informal to very formal, depending on …

The maturity of the testing process	The maturity of the development process	The organisation	The constraints	People involved

Analysis

What to test | Test conditions

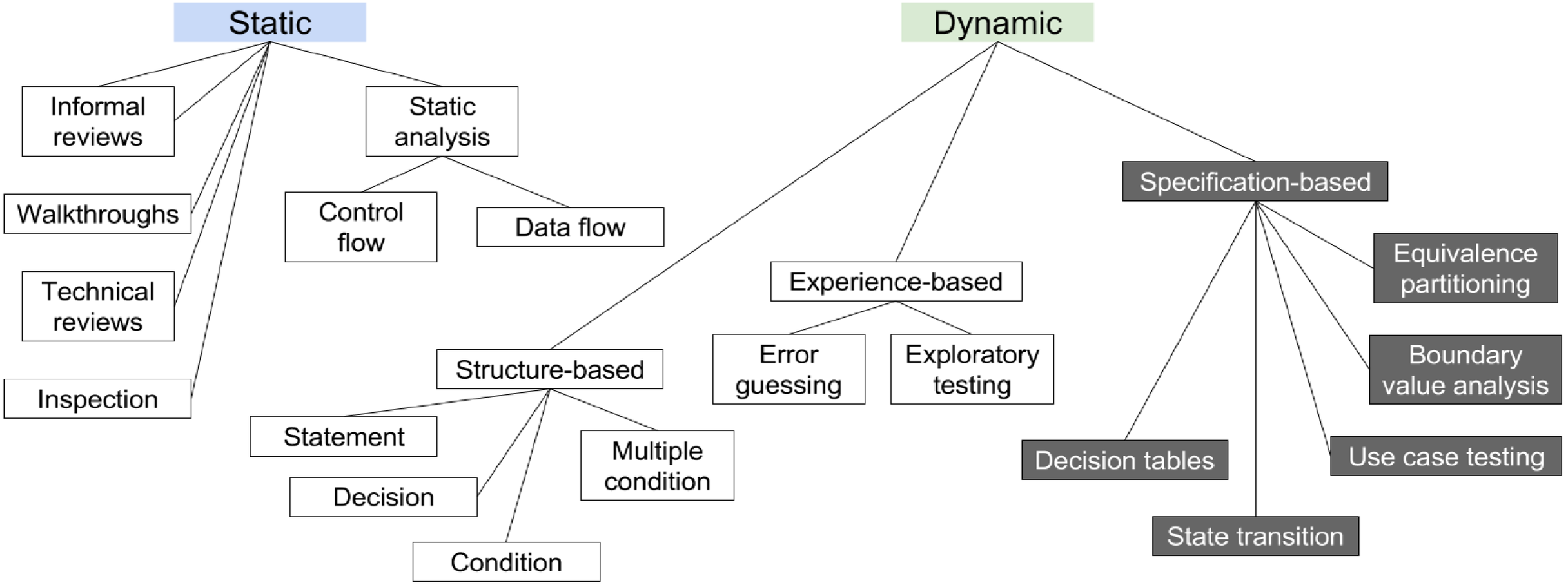
Design

Create and specify test cases | data

Implementation

Develop | Implement | Prioritise | Organise test cases

Categories of Test Design Techniques



Specification-Based Techniques

Equivalence partitioning

Identify variables who will be treated the same

Group values into equivalence classes

Boundary value analysis (BVA)

Testing at the edges of each equivalence class

Things are more likely to go wrong here!

The techniques are often used in combination

Specification-Based Techniques

Use case testing

- Identify test cases that exercise the whole system
  - Transaction by transaction basis
  - From start to finish
- Describes interactions between actor and system
  - Achieve a specific task
  - Produce something of value to the user
- Defined in terms of the actor, not the system
  - Describes process flows through a system
    - Based on its actual use
  - Can uncover integration defects

Use case name	<name>	
Actor(s)	<actor1>, ...	
Pre-conditions	<cond1>, ...	
Post-conditions	<cond1>, ...	
Main Success Scenario	Step	Description
	1	A: <action>
	2	S: <response>
	3	A: <action>
	4	S: <response>
	...	...
Extensions	Step	Description
	S.X	<cause> S: <response>
	S.Y	<cause> S: <response>

Specification-Based Techniques

State transition testing

System can be in a finite number of different states

Elements of state transition models

States → The SW may occupy

E.g. open / closed, active / inactive

Transitions → From one state to another

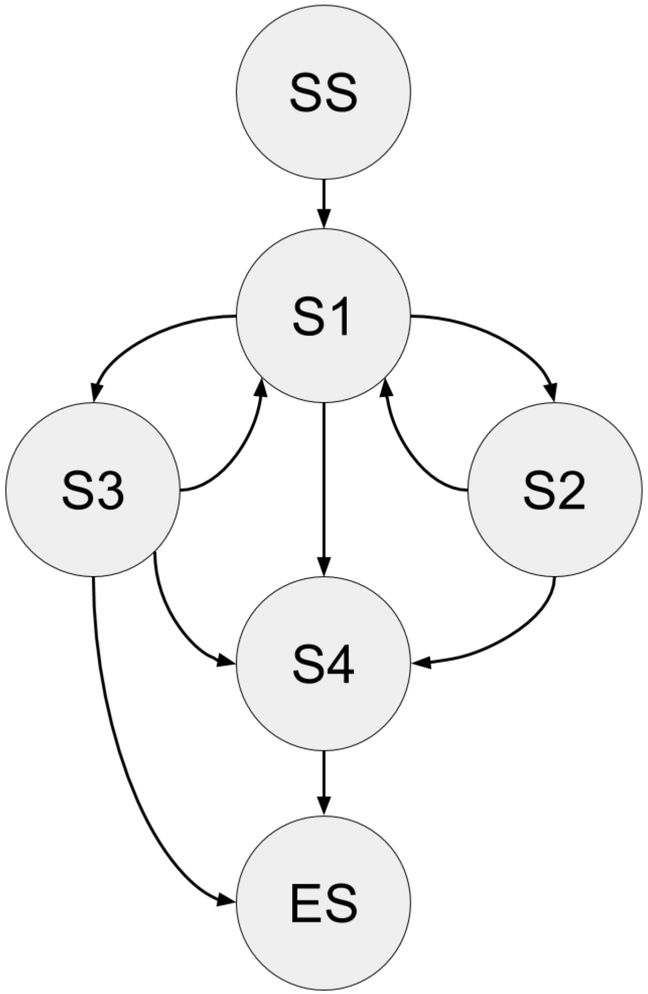
Not all transitions are allowed

Events → Causing state transitions

E.g. closing a file, withdrawing money

Actions → Actions resulting from transitions

E.g. error message



Specification-Based Techniques

Decision tables

Cause-effect table

Used when inputs and actions can be expressed as Boolean values

Systematic way of stating complex business rules

Help testers identify effects of combinations of different input

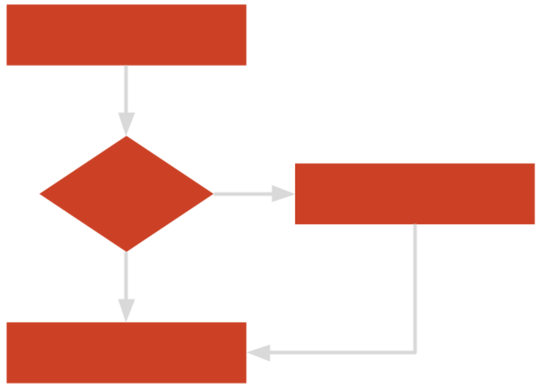
Effective approach to reveal faults in the requirements

Conditions	**R1**	**R2**	**R3**	**R4**
Condition_1	T	T	F	F
Condition_2	T	F	T	F
Actions				
Action_1	?	?	?	?
Action_2	?	?	?	?

Structure-Based Techniques

Statement coverage

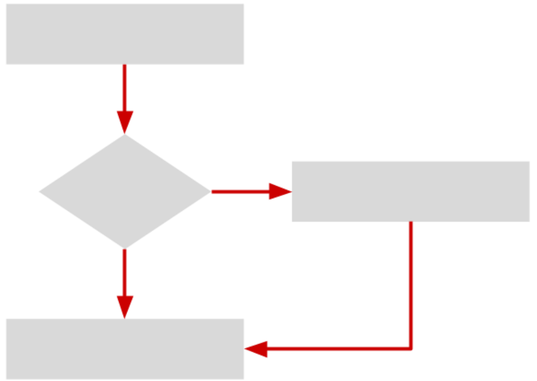
Percentage of statements exercised



$$\text{Statement coverage} = \frac{\text{Number of statements exercised}}{\text{Total number of statements}} \times 100$$

Decision coverage

Percentage of decisions exercised



$$\text{Decision coverage} = \frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100$$

Decision coverage is stronger than statement coverage

100% decision coverage guarantees 100% statement coverage

Not the other way around!

Experience-Based Techniques

Error guessing and Exploratory testing

- Tests derived from skill / knowledge / experience / intuition
  - Both of technical and business people
  - Different groups yield different perspectives
- Often based on similar applications and technologies

When?

- Used predominantly to complement more formal test techniques
- When testing under severe time constraints
- Lacking specification / documentation

Drawbacks

- Success / Effectiveness is highly dependent on the testers skill and experience

Choosing a Test Technique

Internal factors affecting the choice of test techniques

- Testers knowledge and experience
  - How much do testers know about the system / various techniques?
- Likely defects
  - Each technique is good at finding particular defects
- Test objective
    - What do we want from the test effort? → Helps us define approach
- Documentation
  - Exists? Updated? Content → Serves to guide the test effort
- Lifecycle model
  - Sequential → More formal techniques | Iterative → More informal techniques

Choosing a Test Technique

External factors affecting the choice of test techniques

- Risk
  - The greater the risk, the greater the need for more thorough testing
- Customer / Contractual requirements
  - Contracts may specify particular testing techniques to be used
- Type of system
  - Influence techniques used
  - E.g. Financial application → Benefits from boundary value analysis
- Regulatory requirements
  - Some industries have standards / laws that pose external requirements on the system
- Time and Budget

V. Test Management

Tasks of Tester and Test Leader

Test leader

- Coordination
- Planning
- Estimation of time, effort, cost
- Introduce metrics
- Decides what should be automated
- Selects test tools
- Monitor results, and check exit criteria
- Writes summary reports
- Controls test progress and effort

Tester

- Review and contributes to test plan
- Analyses and assesses user requirements
- Creates test specifications
- Prepares the test data
- Implements tests on all levels
- Automates the tests
- Executes and logs the tests
- Documents the results
- Helps other testers

Test Planning

Planning is influenced by

Scope of testing | Test policy of organisation | Objectives | Risks | Constraints | Criticality | Testability | Availability of resources

Activities

Scope and risk

- Determine the scope and risk of testing

Objectives

- Identify the objectives of the test effort

Approach

- Define approach for testing
  - Test levels
  - Entry and exit criteria

Activities

- Integrate and coordinate
  - Development
  - Operation, maintenance

Strategy

- Decisions about
  - What to test
  - Who will test
  - How to test
  - Evaluating results

Schedule

- Schedule
  - Analysis and design
  - Implementation and execution
  - Evaluation

Resources

- Assign resources for the different activities defined

Metrics

- Select metrics for
  - Monitoring and controlling
  - Risk and defect resolution

VI. Tool Support for Testing

Types of Test Tools

Test Management Tools

- Requirements management tools
- Incident management tools
- Configuration management tools

Static Testing Tools

- Review tools
- Static analysis tools (D)
- Modelling tools (D)

Test Specification Tools

- Test design tools
- Test data preparation tools

Execution and Logging Tools

- Test execution tools
- Test harness tools
- Unit test framework tools

Performance and Monitoring Tools

- Test comparators
- Coverage measurement tools (D)
- Security tools
- Dynamic analysis tools (D)
- Performance, load, stress testing tools
- Monitoring tools

Tools for Specific Testing Needs

VII. Usability Testing

HCI Purpose

Human Computer Interaction

> “*The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use*.” (ISO 9241-11)

Purpose of HCI

To make a software system

- Understandable
- Easy to learn
- Easy to use
- Easy to remember
- Satisfactory to use

HCI Framework

Considerations for HCI

Left to right, from most to least important

Interface Standards

Underlying standards for designing user interfaces

- Adhering to best practices
- Consistent behaviour and design

Usability

Concerned with how *easy* something is to use

- Effectiveness
- Efficiency
- Satisfaction

Interface Dynamics

Related to the dynamic aspects of interfaces

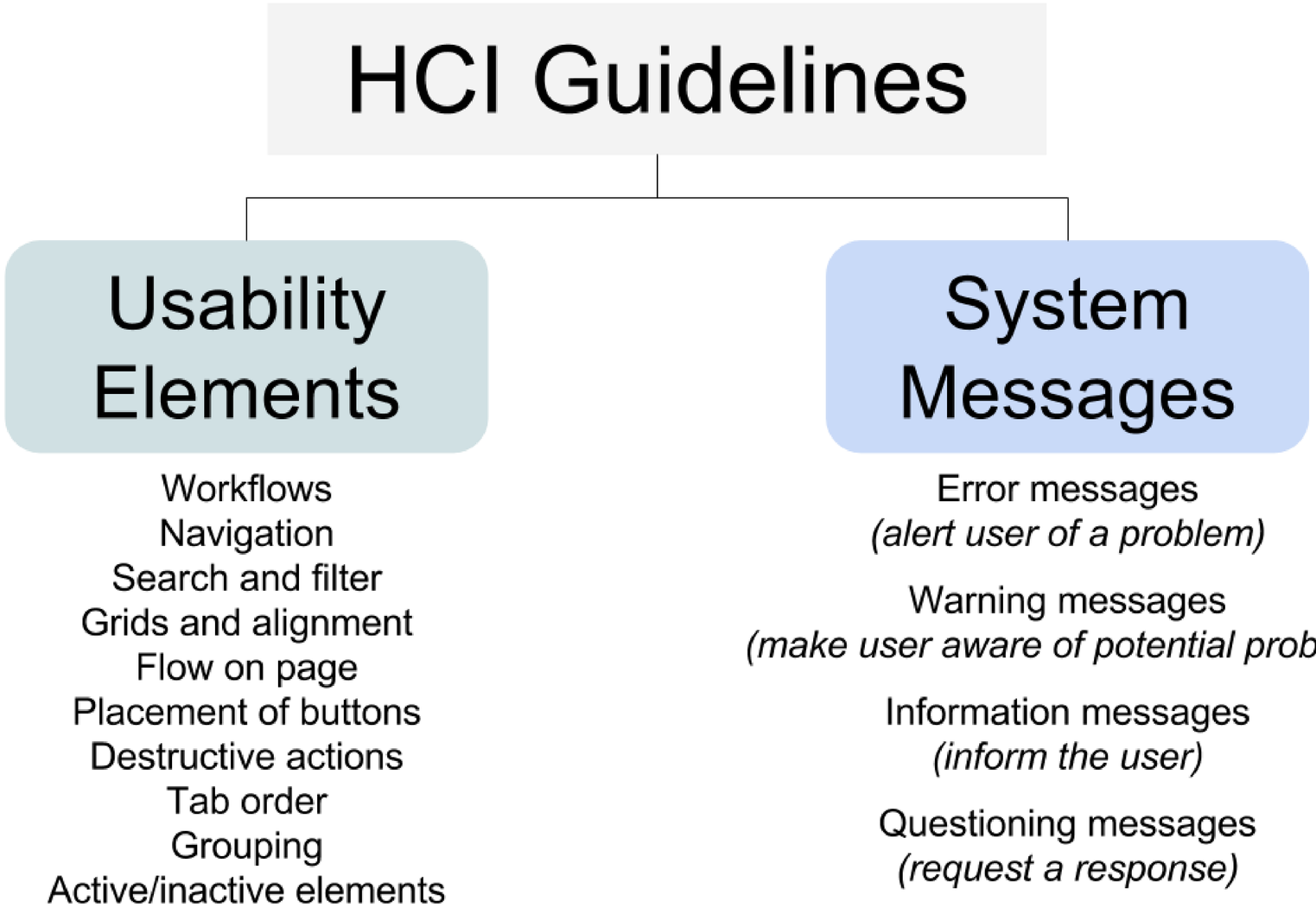
- Responsive and fast
- Adaptable to user needs
- Empowering

Aesthetics

How the system appears

- Fresh, modern, appealing

HCI Guidelines



VII. Accessibility Testing

Accessibility

Definition

> "*The usability of a product, service environment or facility by the people with the widest range of capabilities*." (ISO 9241-20)

In other words, accessibility is:

> *"The degree to which a product, device, service, or environment is available to as many people as possible."*

Barriers

What problems will stop someone from being able to use a software product?

Critical	Serious	Annoying (moderate)	Noisy (minor)
Barriers that stop someone from using a software product (or some features) successfully	Problems that cause frustration, slow someone down, or require work-arounds	Things that are frustrating, but won't stop someone from using the product / site	Minor issues that might cause someone a problem, but which mainly damage credibility

Personas

Design and specification tool

- Description of a representative user

Provide information about

- Who the users are
- Goals, motivations, and activities of usage

Informed based on research and checked to validate assumptions

Role of personas

- Help take different users into account
- Help organising increasing amounts of data
- Build consensus around a clear, consistent view on accessibility needs

Accessibility Personas

Considerations for main persona categories

Characteristics, aptitude and attitude

Assistive technologies

Autism spectrum disorder	Cerebral palsy	Blindness with some light perception	Fibromyalgia (fatigue)
• Text preference settings • Power keyboard user	• Communicator with speech generator • iPad, wheelchair	• Screen reader • Audio note-taker • Braille display	• Split keyboard • Speech recognition software

Deaf-mute	Visual impairment	Age-related macular degeneration	Non-English speaker
• Sign language • Communication-assisted real-time translation • Captions, Video	• Contrast adjustment • Screen magnification • Personalised style sheets	• Test enlargement	• Online translation sites • Plain-English option

WCAG

WCAG → Web Content Accessibility Guidelines

Recommendations for making web content more accessible

P.O.U.R: The Four Principles

<table>
<tr><th></th><th>Description</th><th colspan="3">Examples</th></tr>
<tr><td rowspan="3">Perceivable</td><td rowspan="3">Information and user interface components must be presented in perceivable ways</td><td colspan="2">Text-alternatives</td><td>Adaptable</td></tr>
<tr><td colspan="2">Provide alternatives for any non-text content</td><td>Create content that can be presented in different ways</td></tr>
<tr><td colspan="2">Large print, braille, symbols, etc.</td><td>Simpler layout</td></tr>
<tr><td rowspan="2">Operable</td><td rowspan="2">User interface components and navigation must be operable</td><td>Keyboard accessible</td><td>Enough time</td><td>Navigable</td></tr>
<tr><td>Make all functionality available from the keyboard</td><td>Provide users with enough time to read and use content</td><td>Provide ways to help users navigate and find content</td></tr>
<tr><td rowspan="2">Understandable</td><td rowspan="2">Information and operation of the user interface must be understandable</td><td>Readable</td><td>Predictable</td><td>Input assistance</td></tr>
<tr><td>Make text content readable and understandable</td><td>Make web pages appear and operate in predictable ways</td><td>Help users avoid and correct mistakes</td></tr>
<tr><td rowspan="2">Robust</td><td rowspan="2">Content must be robust enough to be interpreted by a wide range of user agents</td><td colspan="3">Compatible</td></tr>
<tr><td colspan="3">• Maximise compatibility with current and future user agents, and assistive technologies.
• Avoid content that relies on technologies that are not accessibility-supported</td></tr>
</table>

VIII. Exploratory Testing

Doing Exploratory Testing

Learning

Anything that can guide us in

What to test / How to test / Recognise a problem

Design

Creating / fashioning / constructing according to plan

Design is not scripting

Execution

Executing the test and collecting the results

Can be automated or manual

Interpretation

What we learn from the system under test

Information about the system

Information about how we are testing the system

Testing is Context-Dependent

Testing depends on the following

Supporting technologies	Development lifecycle	Product interfaces	Project management
New technology Old technology Someone else's technology	Agile project Traditional, plan-driven project	Batch system GUI system API-only system	Well-managed project Poorly managed project

Type of project	Project size	Type of product	Other characteristics
New project Ongoing project	Small project Huge project	Experimental Mission-critical Life-critical Regulated	Embedded systems Real-time systems

Even for the same product → Testing differs from release to release

First release → Positive testing | Subsequent releases → Performance testing

Patch → Negative, regression testing

Heuristics

Simple strategy for making decisions and finding solutions

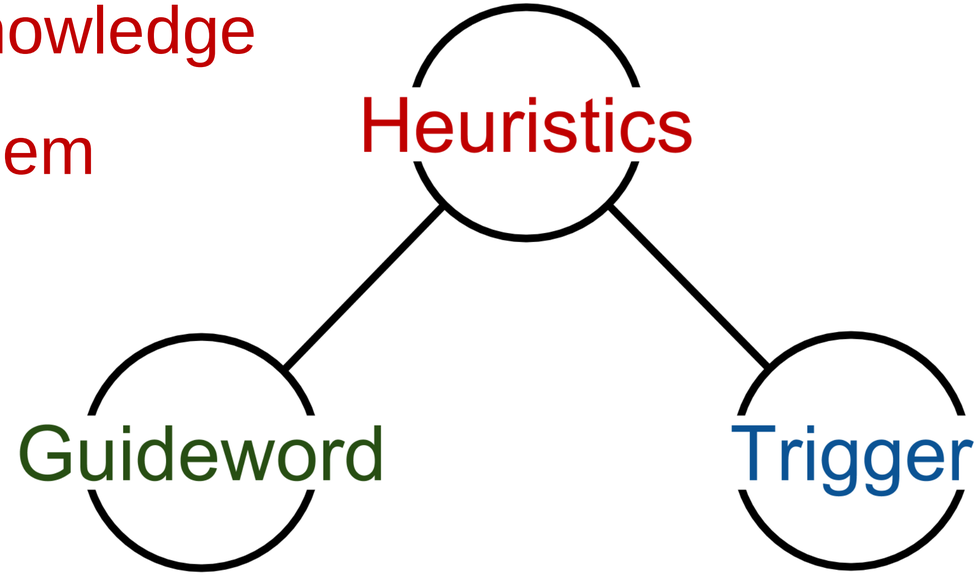
- Does not provide the answers
- Instead: Suggests key elements to consider

Direct attention to the details that are most likely to matter

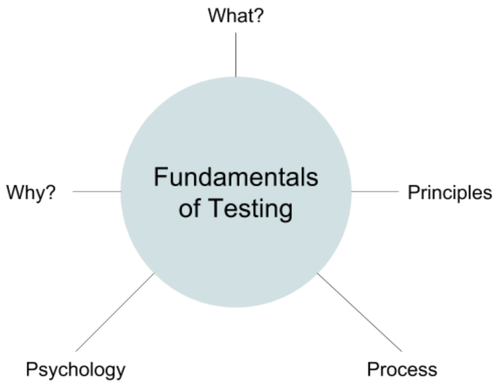
- Labels to access your full spectrum of knowledge
- Effectively analyse and approach a problem
- Based on experience and probability

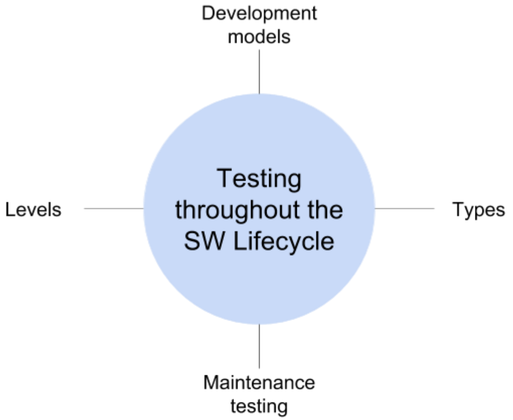
Two categories

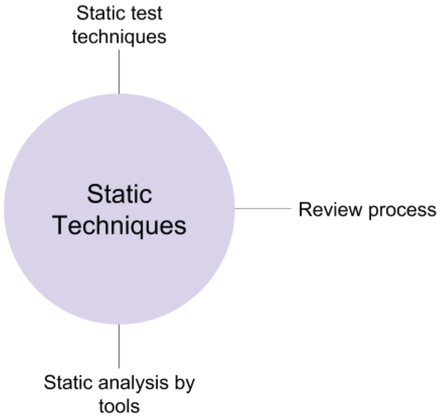
- Guideword heuristics | Trigger heuristics

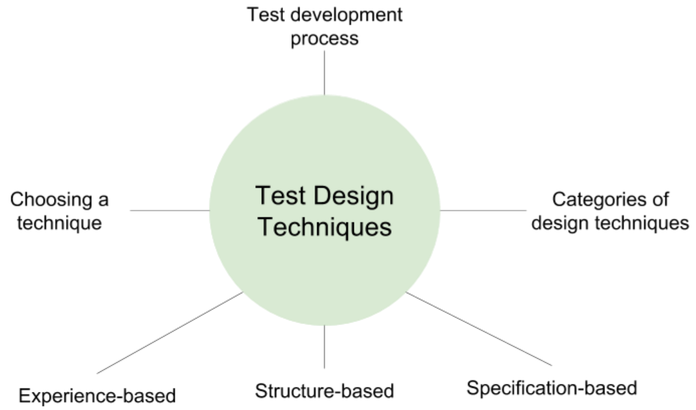


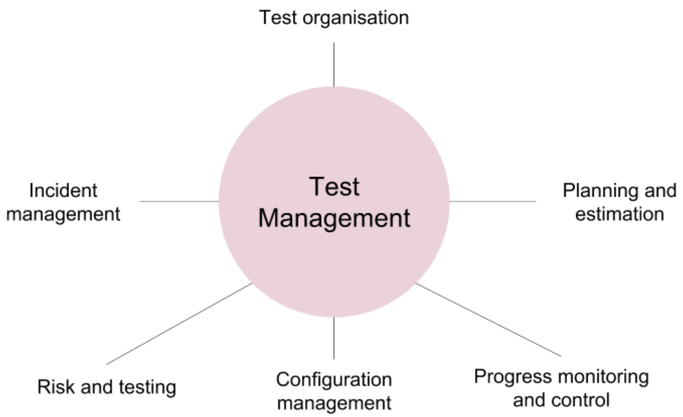
Back to the Overview

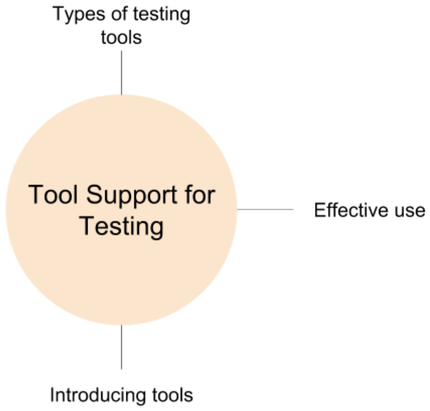


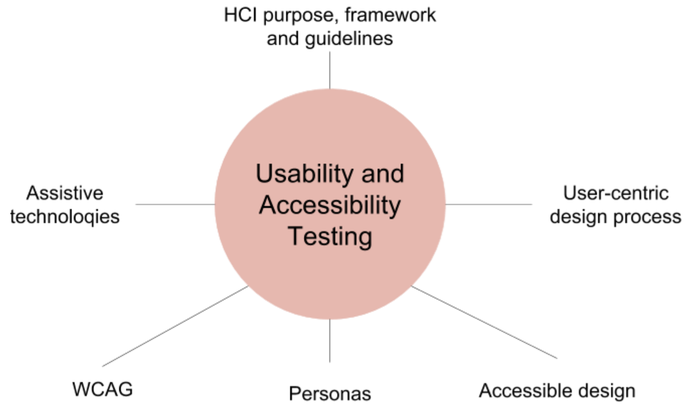


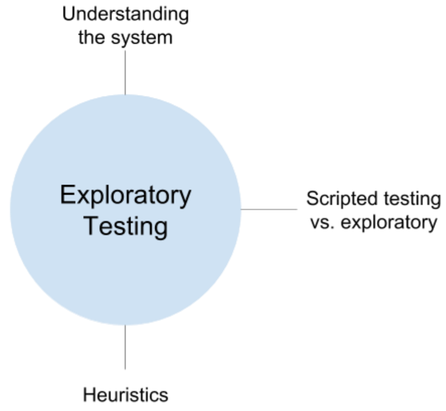












Closing Remarks

The lecture slides were based on

Previous lecture slides by Raluca Florea (2016)

Black, R., van Veenendal, E., Graham, D. (2012). *Foundations of Software Testing: ISTQB Certification 3E*. Cengage Learning.

IEEE 829: Standard for Software and System Documentation