9. Verification, Validation, Testing

(a) Basic Notions

(b) Dynamic testing.

(c) Static analysis.

(d) Modelling.

(e) Environmental Simulation.

(f) Test Strategies.

(g) Tool support.

(h) Independent Verification and Validation

Remark: Only subsections (a) – (d) will be covered in this lecture.

(a) Basic Notions

- Verification is the process of determining whether the output of a life cycle phase fulfils the requirements specified in the previous phase.
  - So task is
    - **not** to demonstrate that the **output** of a development phase is **actually correct**,
    - but that the **output of a phase conforms to its input**.
  - Therefore mistakes in early phases of a project may **propagate** through later stages without detection.

Validation

- **Validation** is the process of confirming that the specification of a phase or of the complete system is appropriate and is consistent with the customer requirements.
- Validation
  - might be performed on individual phases,
  - but is usually performed on the complete system.

Testing

- **Testing** is the process used to verify or validate a system or its components.
  - Sometimes **testing** is used for testing, in which one executes the software in order to check whether it is performing as required.
  - We use testing in the wider sense and **dynamic testing** for this more restricted version of testing.

Testing

- Testing is performed at various stages during the life cycle of a system.
- There are three main activities.
  - **Module testing**.
  - **System integration testing**.
  - **System validation testing**.

Main Testing Activities

- **Module testing** is the evaluation of small, simple functions of hardware or software.
  - Faults detected during module testing are usually easy to locate and to rectify.
- **System integration testing** investigates the characteristics of a collection of modules.
  - Usually investigates the correct interaction between modules.
  - Faults are more difficult to find and more expensive to rectify.

Main Testing Activities

- **System validation testing** tests whether the complete system satisfies the requirements.
  - Problems detected at this stage are usually due to weaknesses of customer requirements or the specification.
  - Problems detected are usually extremely costly to correct, since modifications have to propagate through the entire development process.

Testing Methods

- There are three main testing methods:
  - **Dynamic testing.**
  - **Static analysis.**
  - **Modelling.**

Dynamic Testing

- **Dynamic testing** is the execution of a system or component in order to investigate its characteristics.
- The tests may be carried out
  - in the **system's natural working environment**,
  - or within **simulation of that environment**.
    - Often more cost effective.

Dynamic Testing and Simulation

- Dynamic testing might as well be carried out on one or a few system components by using **simulation**.
  - Especially of advantage if one simulates **hardware** which has **not** been **developed yet**.
  - Then simulation is **cost effective**, since it allows to compare various designs of the hardware involved.
  - However, simulation **never** provides **complete information** on the system behaviour, e.g.
    - **real-time operation**,
    - **problems with timing**.

Static Analysis

- **Static analysis** is the investigation of the characteristics of a system or component without operating it.
- **Examples:**
  - Walkthroughs,
  - formal proofs,
  - data flow analysis.
- Automated software testing packages which carry out static analysis are called **static code analysis tools**.
- Many engineers mean by testing only dynamic testing, not static analysis.

Modelling

- **Modelling** means the mathematical representation of the behaviour of a system or component.
  - Usually carried out at an early stage, in order to investigate the basic nature of the proposed system or its environment.
  - **Animation** of a formal specification is an example of modelling.

Use of Testing Methods

- Typically, a software life cycle involves
  - dynamic testing,
  - static analysis,
  - some form of modelling.

Black/White Box Testing

- Testing methods can be categorised by the information available when performing the work.
  - **Black box testing** means the test engineer has no knowledge about the implementation of the system.
  - **White box testing** means that the test engineer has access to the implementation of the system.

Black Box Testing

- In black box testing, the test engineer relies completely on the specification of the system.
- Therefore it is sometimes called **requirements-based testing**.
- May be applied to individual modules or (more common) to subsystems or the complete system.
- Is widely used for testing software tools like compilers.

Comparison

- **Advantage of Black Box Testing:**
  - Greatest level of independence between developer and evaluator.
- **Advantage of White Box Testing**
  - The test engineer can use information about the implementation in order to develop better tests.

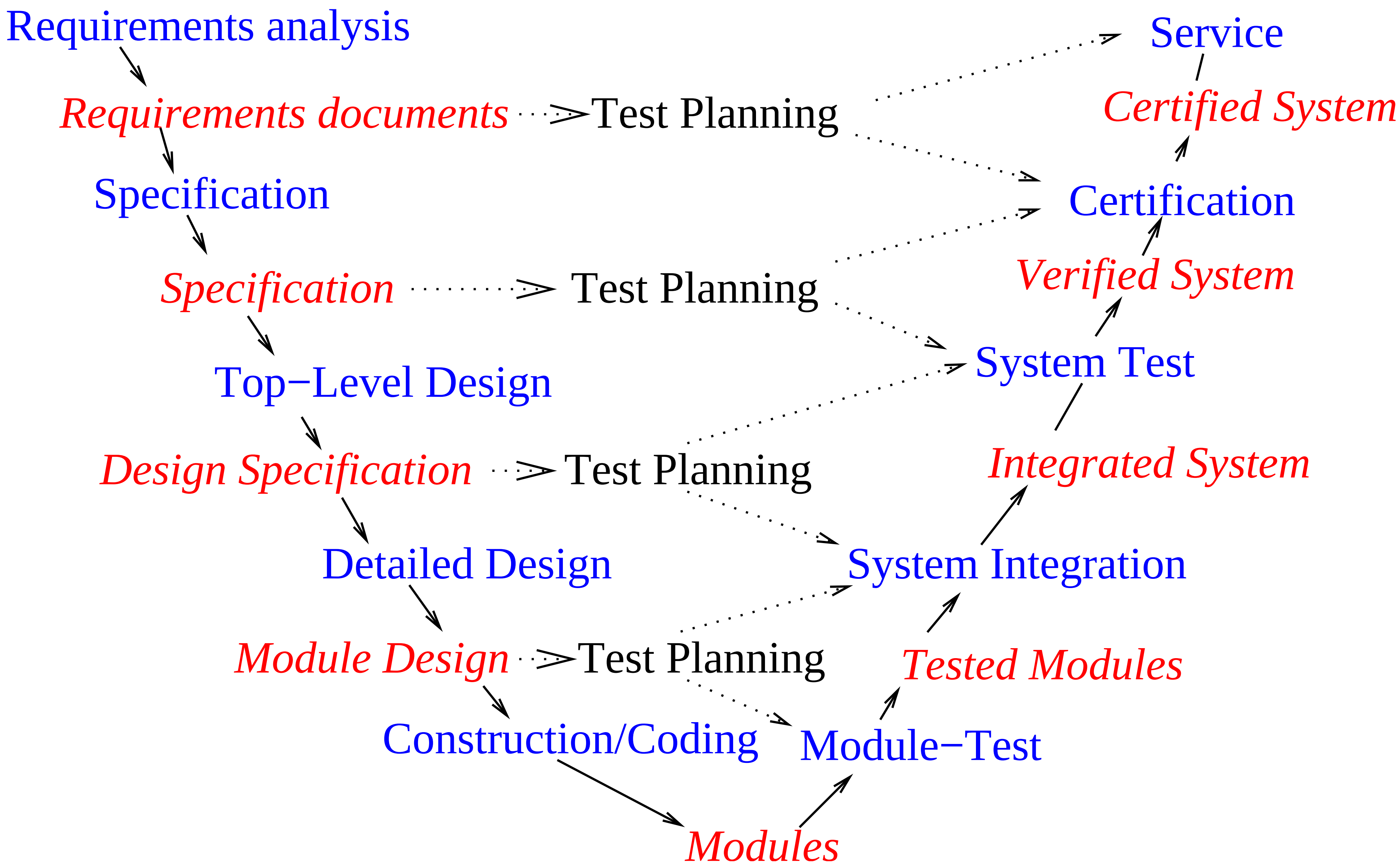
Black/White-Box vs. Static/Dynamic

- **Dynamic testing** can be white-box and black-box testing.
- **Static analysis** is necessarily white-box testing.
- **Mathematical modelling** doesn't use the system software and hardware, so categories white/black-box testing don't apply to it.

Planning for Verific./Valid.

- **Test planning** is an essential part of the software life cycle.
- The next slide shows test planning within the V-model.

V-Model and Test Planning



Testing for Safety

- **Overall safety validation** is the test that a system is in accordance with the safety requirements.
  - The results of it are documented in an **overall safety validation report**.
- Some standards require **traceability**, i.e. that the key safety requirements are traceable throughout all stages of the software life cycle.

Testing for Safety

- Testing for safety requires that that tests are performed which show that **each identified hazard is effectively countered**.
  - **Dynamic testing** might be **sufficient**.
  - Since **exhaustive dynamic testing** is impossible, usually **static analysis** and **mathematical modelling** is **required**.
  - Properties like **reliability** and **failure rates** can usually **not** be **tested dynamically**, therefore static analysis is required.

The Roles of Testing

- Testing has three purposes in a safety-critical project:
  - **Development testing.**
  - **Validation testing.**
  - **Production testing.**

Development/Validation Testing

- **Development testing** is aimed at **locating** faults within the system, so that they may be removed.
  - Uses dynamic, static and modelling techniques.
- **Validation testing** aims at demonstrating the absence of faults and to demonstrate other positive features.
  - Uses again dynamic, static and modelling techniques.

Production Testing

- **Production testing** aims at testing whether a individual unit has defects as a result of manufacturing or component fault.
  - Tests the accuracy of the replication of the appropriate design.
  - Production tests of software are easy and use usually techniques like checksums.
  - Production tests of hardware are very complicated,
    - since the number of possible faults is extremely big.
  - Production testing is always **dynamic**.

(b) Dynamic Testing

- Dynamic testing means that one operates the system under test.
  - Done by the execution of **test cases**, which investigates certain aspects of the system.
  - Each test set consists of
    - a set of input **test data**
      - often called **test vector**.
    - a specification of the expected output,
      - output is often called **output vector**.
    - a statement of the function being tested.

Basic Notions

- With each test cases one associates
  - **pre-conditions**
    - specify the state of the system before the test is executed,
  - **post-condition**
    - define the state the system must be in after the test.
- Some tests investigate the operation of the system under the condition that the **pre-conditions are not met**.
  - Used in order to check what happens if the system deviates from its operation.

Basic Notions

- The **input space** of a system is the set of possible inputs.
  - If a system has n inputs of a simple type like integer, floating point numbers, it has an **n-dimensional input space**.

Categories of Dynamic Testing

- There are 3 main categories of dynamic testing:
  - **Functional testing,**
  - **structural testing,**
  - **random testing.**

Functional Testing

- **Functional testing** is the testing of functions of the system as defined by its specification.
  - For each aspect of the operation tests are carried out.
  - However, tests might cover more than one function.
  - One has to make sure that all functions are covered by the tests.
- It is **black-box testing**, no details about the implementation are needed.
- Often a **test-matrix** is written, which associates each function with tests. See next slide.
  - Used in order to make sure that one has complete coverage of all functions.

Example Test Matrix

	Function investigated					
Test	1	2	3	4	5	6
1	x					
2		x				
3		x	x			
4		x		x		
5	x			x		
6			x	x		
7			x		x	
8	x					x
9			x			x

Structural Testing

- **Structural testing** looks at the internal structure of a system, and uses it into order to check the operation of individual components and their interactions.
  - In case of **hardware testing** uses test signals to investigate particular modules in the system.
  - In case of **software testing**, this involves tests in order to check certain routines or certain execution paths.
    Allows to investigate critical conditions.
- **Coverage-based testing** is structural testing with the goal of testing a large proportion of the system, by having tests for every branch or loop in the system.
- Structural testing is necessarily **white-box testing**.

Random Testing

- **Random testing** uses a test data which are randomly chosen from the input space.
  - Could be randomly sampled from the entire input space.
  - Could be sampled following some probability distribution.
    - The distribution might match the one expected for the operation.
- Aims at detecting fault conditions which are missed by more systematic techniques.

Dynamic Testing Techniques

- We list some of the techniques used.
- Test cases based on equivalence partitioning.
  - The input and outputs of the system/component to be tested is **partitioned** into sets of ranges which are **equivalent**, i.e. expected to be treated the same way.
  - Tests are performed to investigate **each partition**.
  - Both valid and invalid values are partitioned and tested.
    - E.g. for a function dealing with student marks, one might expect that
      - the ranges $40-49\%$, $50-59\%$ etc. form valid partitions.
      - the ranges $< 0\%$, $> 100\%$ form invalid partitions.

Dynamic Testing Techniques

- Test cases based on boundary value analysis.
  - Tests the performance of the system at the **boundaries** of equivalent partitions of inputs and outputs.
  - Again both valid and invalid values are partitioned and tested.
  - For instance, in the above example one might check for
    - valid boundary values like 50%, 49% etc.,
    - for invalid boundary values like -1%, 101%,
    - for valid values at the boundary to invalid values like 0%, 100%.

Dynamic Testing Techniques

- **State transition testing** identifies the different **states** of the component and system.
  - Then tests are preformed in order to investigate
    - **transitions** between states,
    - **events causing** such **transitions**,
    - **actions resulting from** such **transitions**.
- **Probabilistic testing** determines the reliability of a system.
  - Attempts to measure failure rates over a given period of time, or failures on demand.
  - This testing is **difficult** to perform for **critical systems**, since there a very low failure rate is demanded, so probabilistic testing should return a failure rate of 0.

Dynamic Testing Techniques

- **Process simulation** is the simulation of the process or equipment to be controlled by the system.
  - Allows to reproduce lots of situations quickly and safely.
- **Error guessing** means that the test engineer predicts input conditions which are likely to cause problems.
- **Error seeding** means the insertion of errors into a system to see if they are detected by the testing procedures.
  - Is a test for the testing process.
  - May allow to predict the number of unfound errors.

Dynamic Testing Techniques

- **Timing and memory tests** investigate response time and memory consumption of a system.
- **Performance testing** tests that necessary levels of performance are reached.
  - E.g. that a certain number of operations per time unit are achieved.
- **Stress testing** tests the performance of a system under a very high workload.
  - Important for instance for the test of (web-, data base- and other) **servers**.

(c) Static Analysis

- Static testing investigates a system **without operating it**.
- Techniques can be
  - performed **manually**,
    - e.g. walkthroughs, inspections, use of checklists,
  - or using automated **static code analysis tools**
    - e.g. conformance tests for hardware, formal methods, data/information flow analysis, semantic analysis, complexity measurement, range checking.

Static Analysis

- Static analysis aims at establishing properties of the software or software which are **true under all circumstances**.
  - In contrast with **dynamic testing**, which can only test a **small subset** of the input set.

Static Analysis Techniques

- A **code walkthrough** means that an engineer leads colleagues through the design or implementation of software and convinces them of its correctness.
- **Design review** means peer review and systematic investigation of documents by a number of engineers.
- **Checklists** consists of a set of (usually very general) questions used in order to critically and systematically check certain aspects of a system.
- **Formal proofs** are used to show the correctness of some aspects of the design or implementation of a system.

Static Analysis Techniques

- **Fagan inspections** form a systematic audit of quality assurance documents in order to find errors and omissions.
  - Consists of **5 stages**:
    - planning,
    - preparation,
    - inspection,
    - rework,
    - follow-up.

Static Analysis Techniques

- Control flow analysis
  - Analysis of software to detect poor and potentially incorrect program structure.
  - Looks for inaccessible code, infinite loops, poor or error-prone structural program elements.
  - Performed in SPARK Ada.

Static Analysis Techniques

- Data flow analysis
  - Analysis of the flow of data through a program.
  - Checks appropriateness of operations and comparison between actual and required data flow.
  - Checks
    - whether variables are initialised,
    - the input/output behaviour of variables,
    - the dependencies between variables.
  - Performed in SPARK Ada.

Static Analysis Techniques

- **Symbolic execution** uses algebraic variables instead of numeric inputs and computes the result of the program in the form of algebraic expressions.
  - Results of a program can be compared with those predicted by the specification.
  - Usually results too complicated to be analysed, need some form of user guidance.
  - Some tools (**semantic analysers**) perform automatic simplification of data.
  - Check of verification conditions in SPARK Ada together with the simplifier form an example of symbolic execution.

Static Analysis Techniques

- **Metrics** are measures for certain properties of the software.
  - Measure for instance reliability and complexity.
  - Tools perform the analysis of such metrics.
  - Such tools measure for instance:
    - The **graph theoretic complexity** based on the complexity of the program graph.
    - **Module accessibility**, the number of ways a module can be accessed.
    - **Complexity measures**.
    - **Number of entry and exit points per module**

Static Analysis Techniques

- **Sneak circuit analysis**.
  - **Sneak currents** are latent conditions in a system, which cause it to malfunction under certain conditions.
  - Might be
    - physical paths,
    - timing irregularities,
    - ambiguous display messages,
    - and others.
  - **Sneak circuit analysis** aims at locating such weaknesses by looking at basic topological patterns within hardware and software.

(d) Modelling

- Modelling used especially in the early phases of project development.
- Particularly important when producing the **specification** and the **top-level design**.
- Plays as well an important role later, especially during **system validation**.

Modelling Techniques

- **Formal methods** can be used to model a system.
- **Software prototyping/animation** means that a software prototype is created which represents certain features of the specification.
  - Used for the validation of the specification.

Modelling Techniques

- **Performance modelling** consists of the following steps:
  - A **model of the system processes** and their interactions is constructed.
  - Then the **requirements of processor time** and **memory requirements** for each function of the system are determined.
  - Finally the **total system demand** is determined under average and worst-case conditions.
  - This is used in order to **guarantee** that the system **always satisfies the demand**, including margins for safety.

Modelling Techniques

- **State transition diagrams** means that
  - the system is represented by finitely many discrete states;
  - with the transitions formed by the system, one obtains a **finite state machine**.
  - the system can now be analysed and checked for **completeness**, **consistency**, **reachability**.
  - **Model checking** is a technique based on state transition diagrams.
    - Used especially in hardware verification.

Modelling Techniques

- **Process algebras** and **Petri-nets** model a system in terms of various processes.
  - Conditions like **correctness**, **termination**, **deadlock-freedom** can be examined using these techniques.
  - Commonly used especially for concurrent systems, e.g.
    - railway interlocking systems,
    - networks,
    - verification of the Netscape web-browser.

Modelling Techniques

- **Data flow analysis** (see above) can be considered as well as a modelling technique.
- **Structure diagrams** represent the program structure by a structure chart, which is a tree representing the relationship between the program units.
- **Environmental modelling** means that one simulates the operating environment of a system in order to test it in an almost real environment.