Software Testing Foundation Level

Lecture 1 – Fundamentals of Testing

Uwe Gühl

Contents

Contents

(Fatal) software defects

Mars Climate Orbiter Loss, September 1999

- At 2 am on September 23 1999, 5 minutes before it was due to go behind the planet, the Mars Climate Orbiter fired it's main engine to go into orbit around Mars.
- No signal was detected from the spacecraft when it was due to come out from behind the planets shadow.
- The plan was for the spacecraft to orbit at an altitude of 153 kilometers, which was far above the minimum survivable altitude of 85 kilometers.
- However the last six to eight hours of data indicate the approach altitude was much lower at just 60 kilometers.
- Why did the spacecraft approach so low?

Reason

- The likely cause of the problem related to the transfer of information between the modules of code written by 2 groups:
  - Mars Climate Orbiter spacecraft team in Colorado
  - Mission navigation team in California

  It seems **that one team used English units** (e.g., inches, feet and pounds) while **the other used metric units** and there seems to have been no conversion between the two.

(Fatal) software defects

Meltdown and **Spectre** work on personal computers, mobile devices, and in the cloud. Depending on the cloud provider's infrastructure, it might be possible to steal data from other customers.

Meltdown breaks the most fundamental isolation between user applications and the operating system. This attack allows a program to access the memory, and thus also the secrets, of other programs and the operating system.

Spectre breaks the isolation between different applications. It allows an attacker to trick error-free programs, which follow best practices, into leaking their secrets. In fact, the safety checks of said best practices actually increase the attack surface and may make applications more susceptible to Spectre

Source: https://meltdownattack.com/

(Fatal) software defects

Boeing 737 MAX Grounding

- Two fatal accidents with new airplanes happened with 346 deaths
  - 189 on Lion Air Flight 610 on October 29, 2018
  - 157 on Ethiopian Airlines Flight 302 on March 10, 2019
- In March 2019, aviation authorities around the world grounded the Boeing 737 MAX passenger airliner
- Because of this longest grounding ever of a U.S. airliner, Boeing had lost over $10 billion as of November 2019

Reason

- The activation logic of MCAS (*Maneuvering Characteristics Augmentation System*) – part of the flight control system of Boeing 737 MAX – has been shown to be vulnerable to erroneous angle of attack data, as analyses have shown.
- Additionally increased regulator scrutiny lead to newly discovered problems.

Source: https://en.wikipedia.org/wiki/Boeing_737_MAX_groundings/

(Fatal) software defects

Pacemaker hack

- In August 2018 at the Black Hat conference, Billy Rios and Jonathan Butts, have demonstrated serious security vulnerabilities in the software of the pacemakers.
- In Medtronic's infrastructure an attacker could run malicious firmware to control implanted pacemakers remotely
- According to Rios, Medtronic had been warned of the dangers 18 months earlier, but did not take action
- In October 2018, Medtronic has pulled the plug on its internet-based software update system

Sources: https://www.wired.com/story/pacemaker-hack-malware-black-hat/
https://techcrunch.com/2018/10/16/medical-device-maker-medtronic-finally-fixes-its-hackable-pacemaker/

Testing is more

- Testing is not only running tests
- ***Testing**:* The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of a component or system and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.

Testing is more

- ***Dynamic Testing**:* Testing that involves the execution of the test item
- ***Static Testing**:* Testing a work product without the work product code being executed.
- So, testing covers as well
  - Evaluation of the quality of a test object.
  - Review of work products such as
    - requirements,
    - user stories, and
    - source code.
- Test activities are organized and carried out differently in different lifecycles

2.1

Objectives of Testing

- The objectives of testing depend on the project, they might include:
  - Prevent defects by evaluating requirements, user stories, design, and code
  - Verify if all specified requirements have been fulfilled
  - Check if the test object is complete and validate if it works as the users and other stakeholders expect
  - Find defects and failures
  - Comply with contractual, legal, or regulatory requirements or standards, and/or to verify the test object's compliance with such requirements or standards

Objectives of Testing

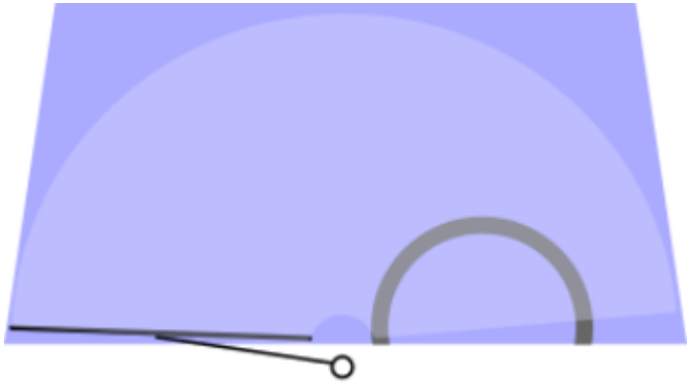


Image source: https://commons.wikimedia.org/wiki/File:Scheibenwischer3.svg/

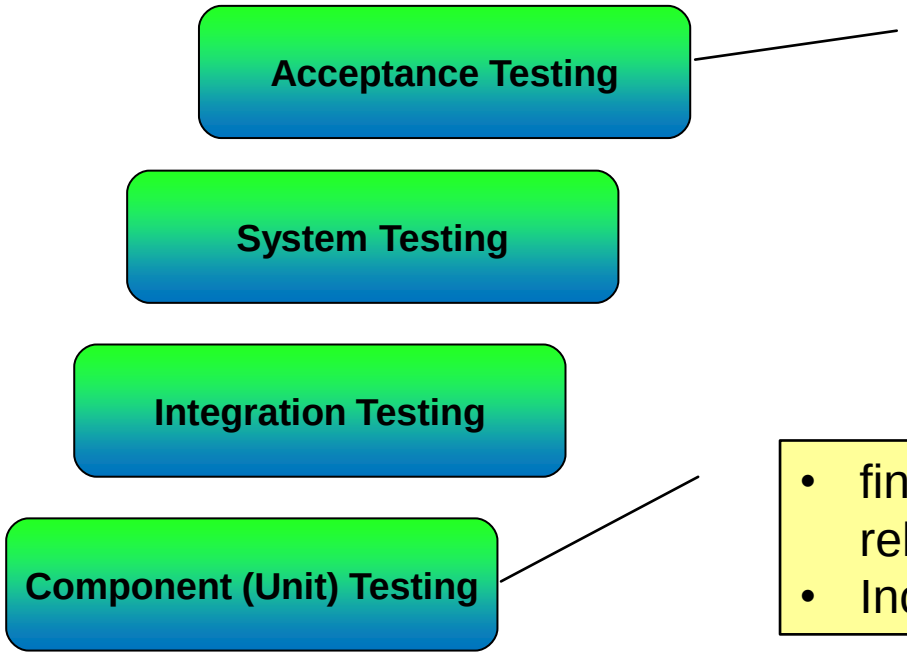
Provide sufficient information to stakeholders to allow them to make informed decisions, especially regarding the level of quality of the test object. For example, based on a specification:

1. Test coverage
2. Defects with criticality
3. Statements concerning software quality criteria

Final test report

Objectives of Testing

- The objectives can vary depending on context, test level, and the software development lifecycle model; for example with following goals:



- confirm that the system works as expected and satisfies requirements.
- give information to stakeholders about the risk of releasing the system at a given time.

- find as many failures as possible so that related defects could be fixed early.
- Increase code coverage

Objectives of Testing

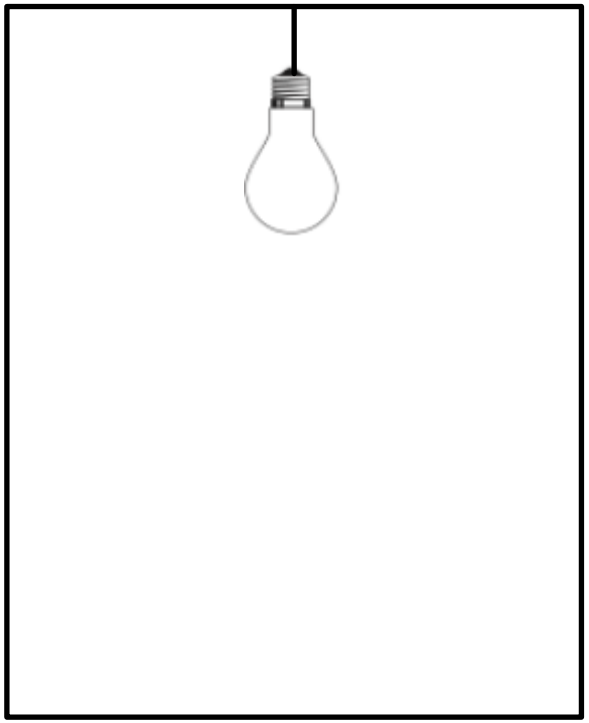
- “In most cases ‘what’ you test in a system is much more important than ‘how much’ you test” (Craig 2002)
- “Prioritize tests so that, when ever you stop testing, you have done the best testing in the time available” (ISEB testing foundation course material 2003)

Testing and Debugging

- Testing
  - Testing can show failures that are caused by defects.
  - Responsible: Tester
- Debugging
  - ***Debugging****:* The process of finding, analyzing and removing the causes of failures in a component or system.
  - Responsible: Developer

Testing and Debugging

- How many testers does it take to change a light bulb?



  - None.
  - Testers just notice that a room is dark.
    Testers don’t fix the problems, they just find them.

Summary

- Testing is a process, and covers below others planning, preparation and evaluation of a component or system and related work products
- Testing has specific objectives depending on
  - Context of the system/component to be tested
  - Test level
  - Software development lifecycle model
- Testing and debugging are different
  - Testing: Find defects
  - Debugging: Analyze and fix defects

Contents

Testing's Contributions to Success

- Appropriate testing could help to reduce defective deliveries, e.g. by
  - Involving testers in requirements reviews and user story refinements
    Related defect fixes increase the probability of correct testable features
  - Validating and verifying software prior to release can detect failures that could be fixed before delivery

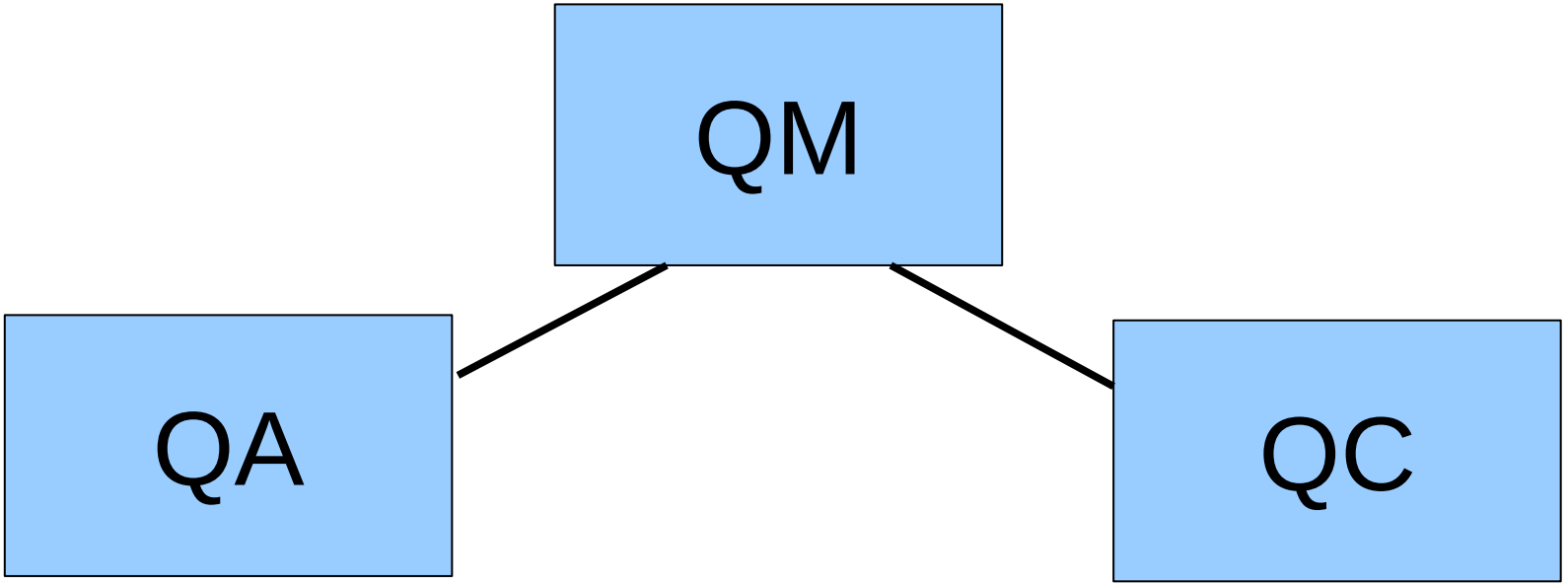
Testing's Contributions to Success

- ***Validation***
  Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled.
  "*Did we build the right product?*"
- ***Verification***
  Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled.
  "*Did we build the product right?*"

Quality Assurance and Testing

- ***Quality management (QM)***
  Coordinated activities to direct and control an organization with regard to quality that include establishing a quality policy and quality objectives, quality planning, quality control, quality assurance, and quality improvement
- ***Quality assurance (QA)***
  Activities focused on providing confidence that quality requirements will be fulfilled.
- ***Quality control (QC)***
  A set of activities designed to evaluate the quality of a component or system.
- ***Test process improvement***
  A program of activities designed to improve the performance and maturity of the organization's test processes and the results of such a program.

Quality Assurance and Testing



Are we building the product right?

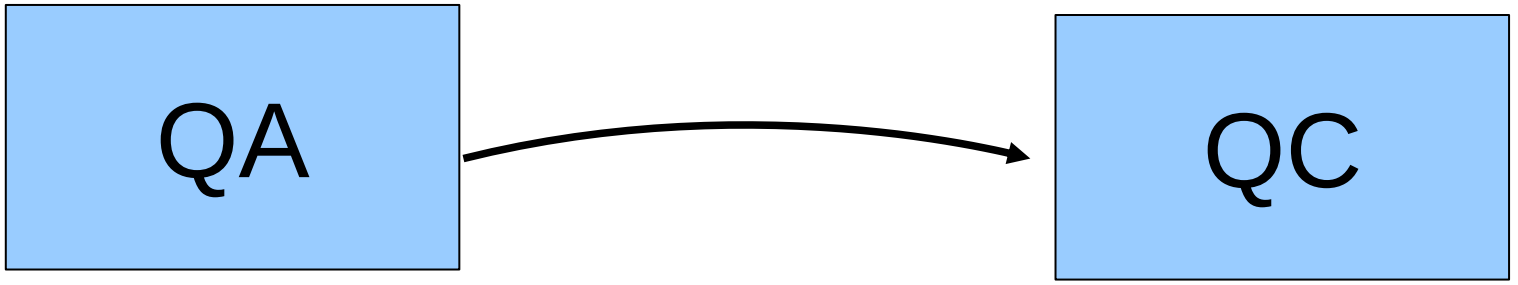
Prevention of faults
by inspecting and testing the **process**

Are we building the right product?

Detection of faults
by inspecting and testing the **product**

Quality Assurance and Testing

- Relationship QA – QC
  As QA inspects the processes, it investigates in test processes as well, resulting in test process improvements

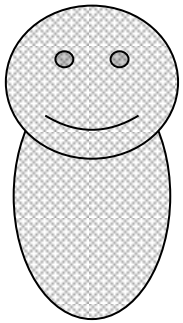


Examples for test processes and test work products

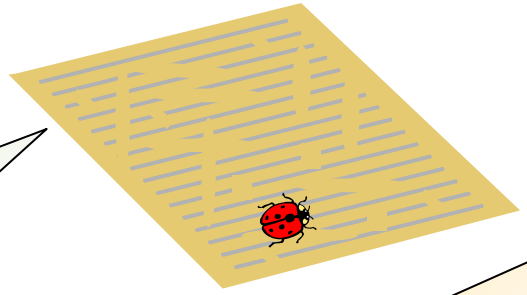
- Defect Management Process
- Test Case Creation Process
- Test Cases
- Test Reports

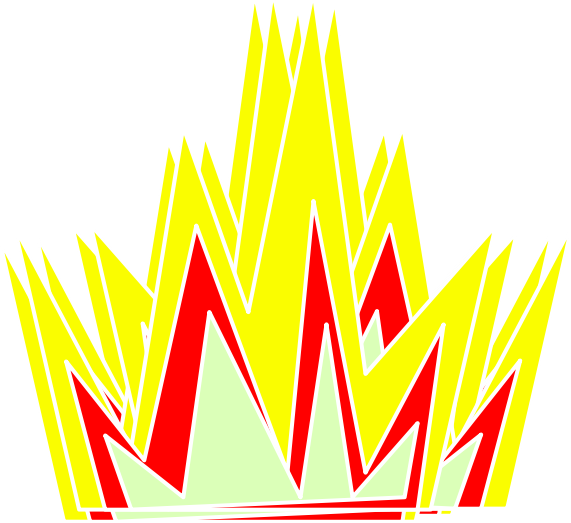
Errors, Defects, and Failures

A person makes an **error** …



… that creates a **defect** in the software





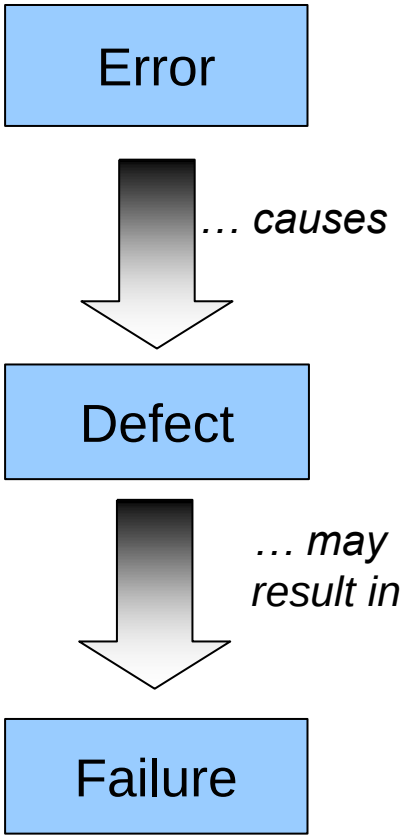
… that can cause a **failure** in operation

Defect – could be detected during static testing

Failure – could be detected only by dynamic testing

Source: https://www.softwaretestinggenius.com

Errors, Defects, and Failures



- ***Error***
  A human action that produces an incorrect result.
- ***Defect*** ***(Synonyms:*** *bug, fault)*
  An imperfection or deficiency in a work product where it does not meet its requirements or specifications.
- ***Failure***
  An event in which a component or system does not perform a required function within specified limits.
- ***Root cause***
  A source of a defect such that if it is removed, the occurrence of the defect type is decreased or removed.
- ***Root cause analysis*** ***(Synonym:*** *causal analysis*)
  An analysis technique aimed at identifying the root causes of defects. By directing corrective measures at root causes, it is hoped that the likelihood of defect recurrence will be minimized.

Errors, Defects, and Failures

- Possible reasons for errors
  - Time pressure
  - Inexperienced or insufficiently skilled project members
  - Miscommunication, especially about requirements
  - Complexity of the code, design, architecture, the underlying problem to be solved, and/or the technologies used
  - Misunderstanding about interfaces
  - New technologies

Summary

- Testing should be considered early in projects, e.g. for reviews of working products.
- Testing and quality assurance are not the same, but related.
- An ***error*** causes a ***defect*** and may result in a ***failure***.
- Communication issues are often reasons for errors.

Contents

Principle 1
Testing shows the presence of defects, not their absence

- “Program testing can be used to show the presence of bugs, but never to show their absence!” (Dijkstra 1969)
- Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

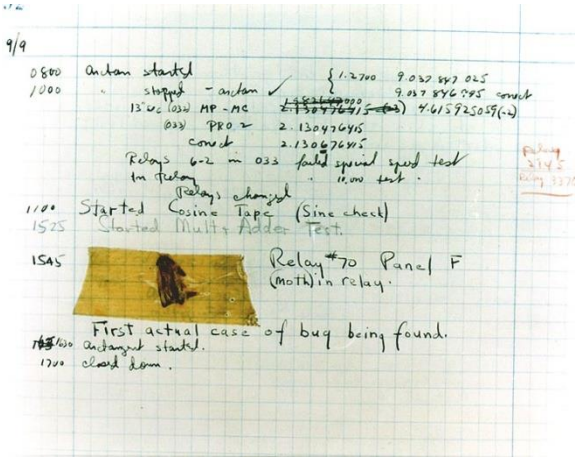


Image source: https://upload.wikimedia.org/wikipedia/commons/8/8a/H96566k.jpg

Principle 2
Exhaustive testing is impossible

A simple program with three integers, up to 16 Bit, should be tested.

Every combination should be considered.

How long is the duration assuming 100.000 tests / second?

Solution: $2^{16} * 2^{16} * 2^{16} = 2^{48}$ combinations

= 281.474.976.710.656 combinations

Duration: About 90 years

Image source: https://commons.wikimedia.org/wiki/File:GHS-pictogram-explos.svg

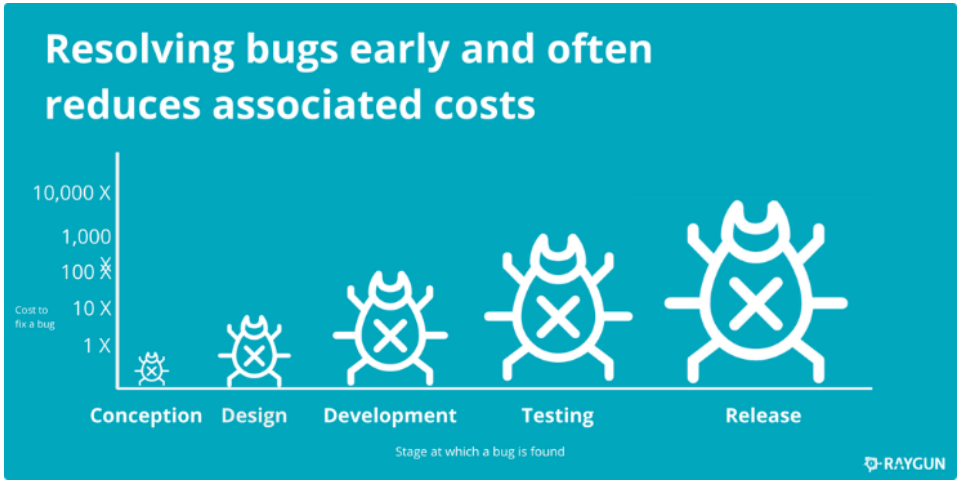
Principle 2
Exhaustive testing is impossible

- ***Exhaustive testing*** (***Synonym:*** *complete testing*):
  A test approach in which the test suite comprises all combinations of input values and preconditions.

  ⇨ not feasible except for trivial cases.
- ***Risk based testing***: Testing in which the management, selection, prioritization, and use of testing activities and resources are based on corresponding risk types and risk levels.

  ⇨ should be used instead; recommendation:
  - Based on most important requirements
    => Business process analysis and prioritization
  - Based on highest risk
    => Risk analysis and prioritization

Principle 3
Early testing saves time and money



3.1

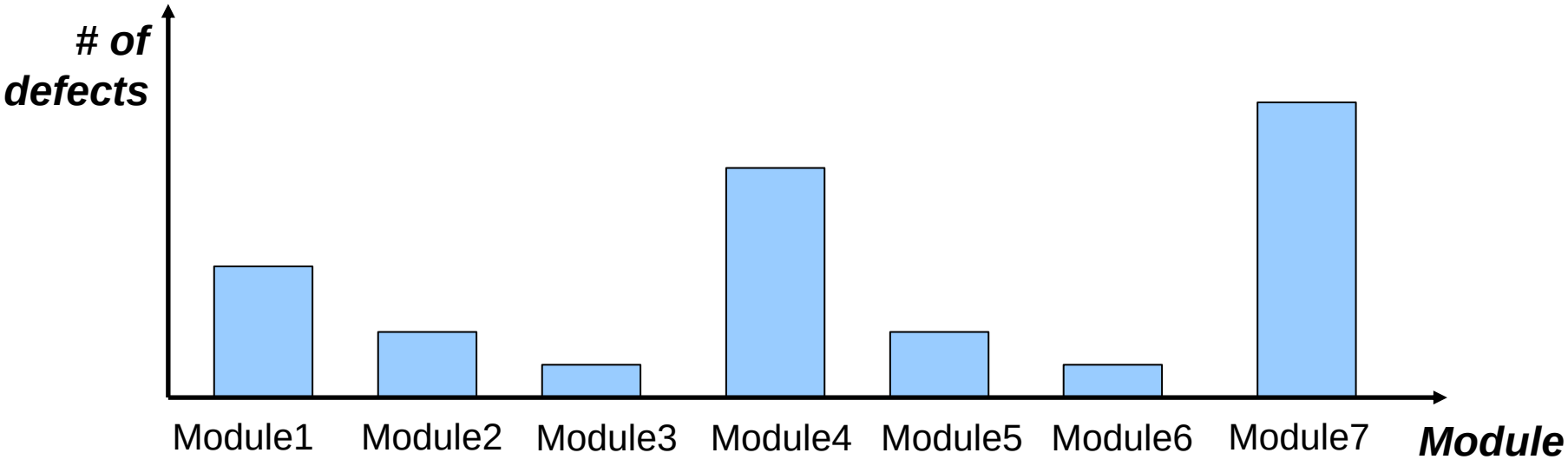
Source: https://azevedorafaela.com/2018/04/27/what-is-the-cost-of-a-bug/

Principle 3
Early testing saves time and money

- Costs of testing depend on various factors like maturity of the development process or quality of the software
- In general finding defects as early as possible lowers the costs
- To find defects early ...
  => start testing activities as early as possible in the software or system development life cycle,
  => focus on defined objectives.

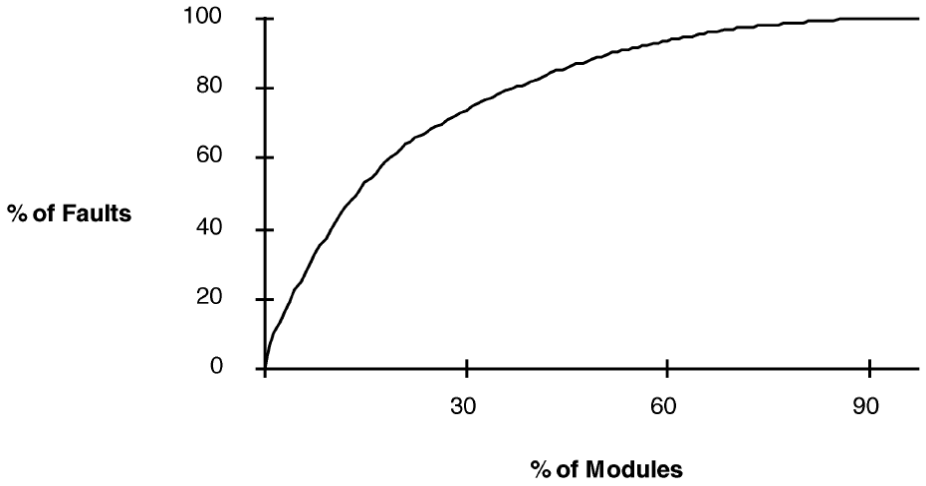
Principle 4
Defects cluster together

- A small number of modules usually contains most of the defects discovered during pre-release testing, or is responsible for most of the operational failures.
- Pareto principle – the 80-20 rule
  Approximately 80 per cent of the problems are caused by 20 per cent of the modules.



Principle 4
Defects cluster together

- Fenton and Ohlsen detected in empirical investigations that 20 % of the modules (equals to about 30 % of the code) are source of 60 % of the defects [FO00].



The diagram shows
% of modules versus
% of faults for a release n

Source: Norman E. Fenton, Niclas Ohlsen:
Quantitative Analysis of Faults and Failures in a Complex Software System;
IEEE Transactions on Software Engineering, Vol. 26, No. 7, July 2000

Principle 5
Beware of the pesticide paradox

- If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.
- To overcome this “pesticide paradox”:
  - Regularly review and revise test cases
  - Write new and different tests to exercise different parts of the software or system to find potentially more defects.

Principle 6
Testing is context dependent

- Basic for Testing is the needed software quality.
- Testing is done differently in different contexts.
  Compare
  - Quality requirements of medical software ⇔ web application
  - Testing of safety-critical software ⇔ e-commerce mobile app
  - Testing in Agile project ⇔ a sequential software development lifecycle project

2.1

- Balance
  Effort for testing must be related to expected quality

Principle 7
Absence-of-errors is a fallacy

Finding and fixing defects

- will not ensure the success of a system
- does not help if the system built is unusable and does not fulfill the users' needs and expectations.

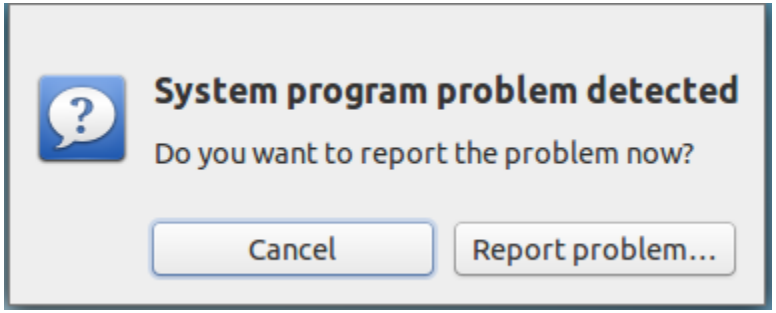


Image Source: https://www.reddit.com/r/softwaregore/comments/1afbjv/most_useless_error_message_2013/

Summary

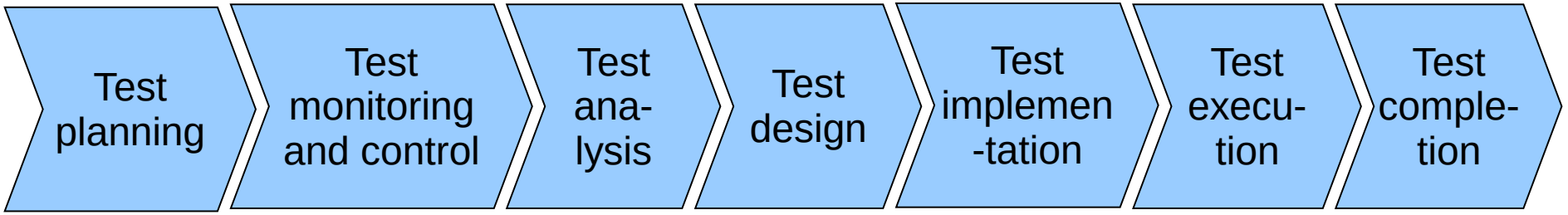
Experience over the past 50 years resulted in general guidelines common for all testing known as principles

1. Testing shows the presence of defects, not their absence
2. Exhaustive testing is impossible
3. Early testing saves time and money
4. Defects cluster together
5. Beware of the pesticide paradox
6. Testing is context dependent
7. Absence-of-errors is a fallacy

Contents

Test Process

- There is no one universal software test process, but there are common sets of test activities
- ***Test process***: The set of interrelated activities comprising of test planning, test monitoring and control, test analysis, test design, test implementation, test execution, and test completion.
- An organization‘s test strategy defines details
- For projects tailoring is required – depending on context



Test Process in Context

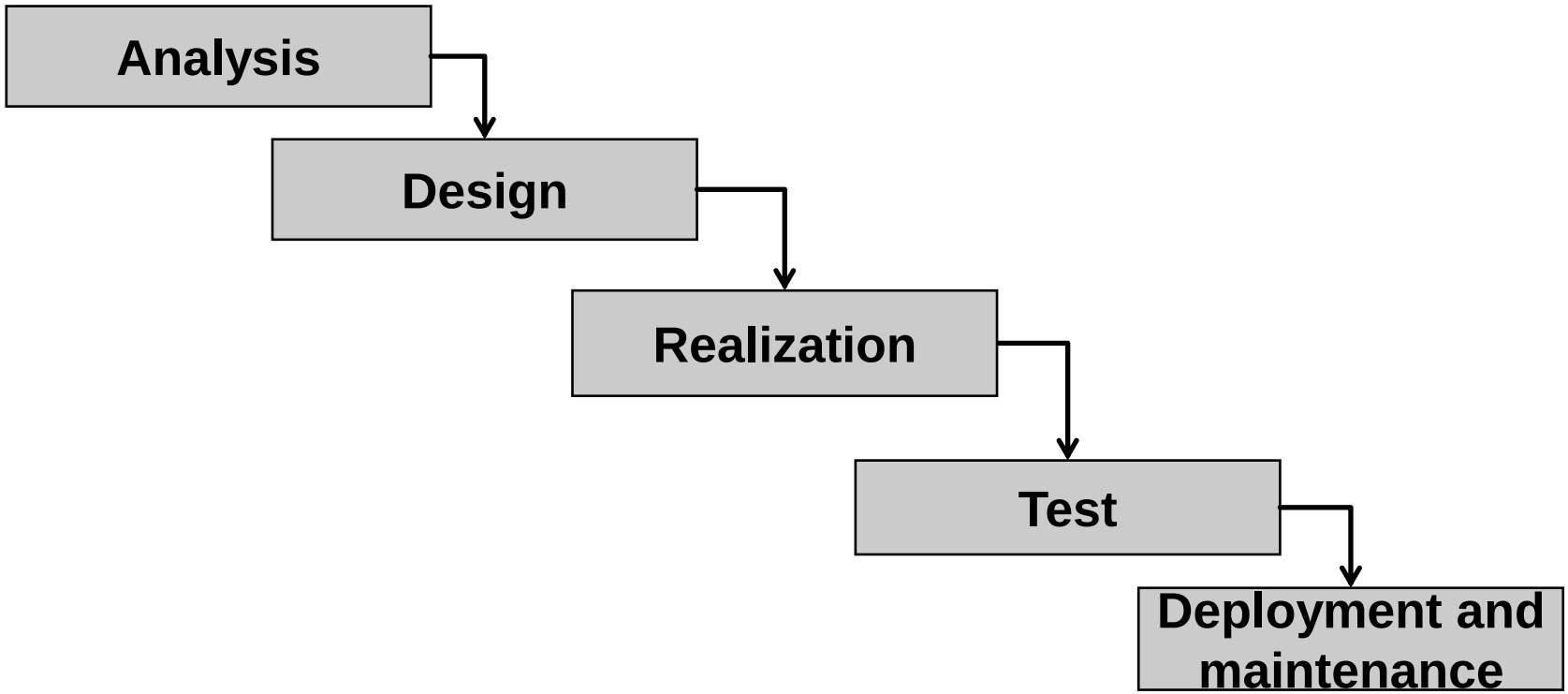
Following factors could influence the test process

- Software development lifecycle model and project methodologies being used
- Test levels and test types being considered
- Product and project risks
- Business domain
- Operational constraints like
  - Budgets and resources
  - Timescales
  - Complexity
  - Contractual and regulatory requirements
- Organizational policies and practices
- Required internal and external standards

Test Process in Context

Examples for implemented test processes

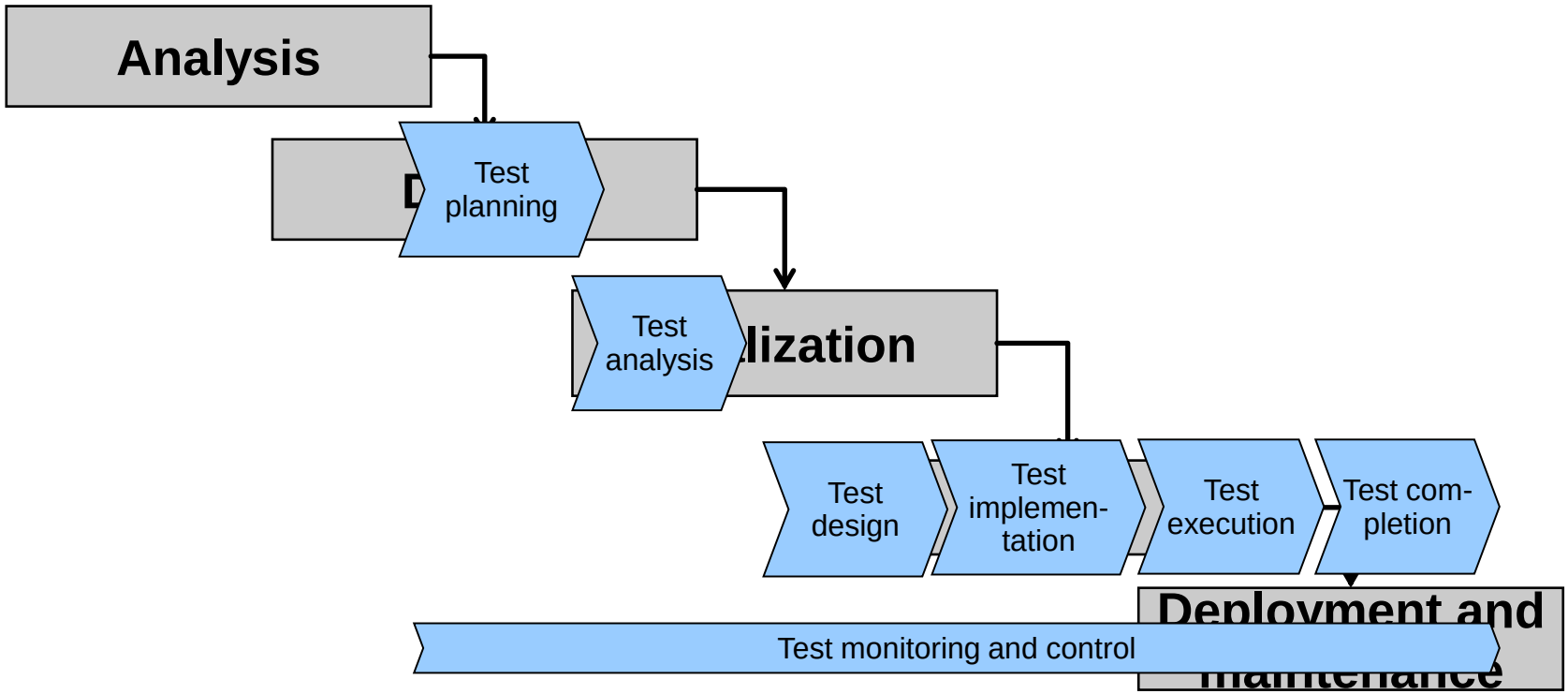
1. Sequential software development



Test Process in Context

Examples for implemented test processes

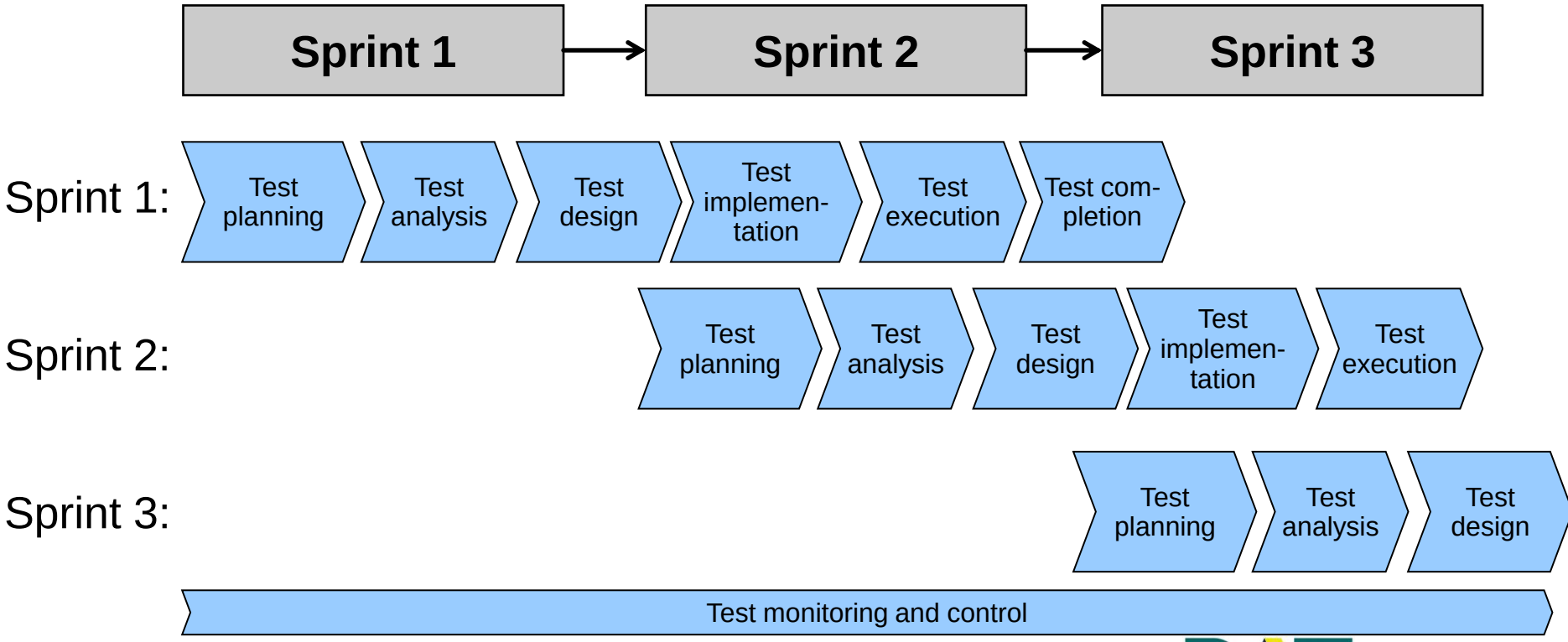
1. Sequential software development



Test Process in Context

Examples for implemented test processes

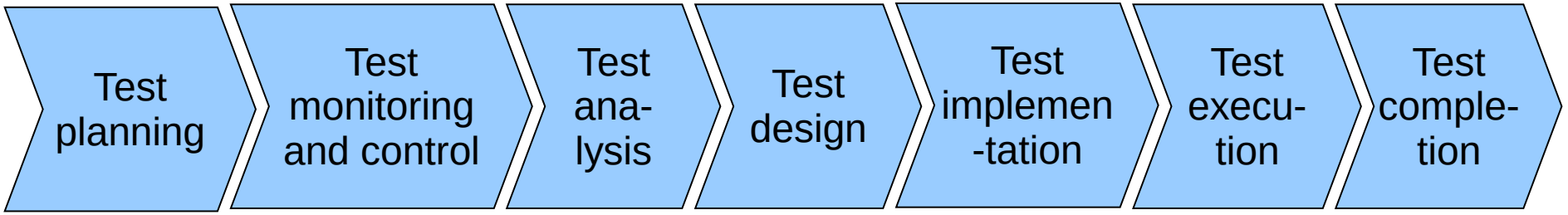
2. Agile development

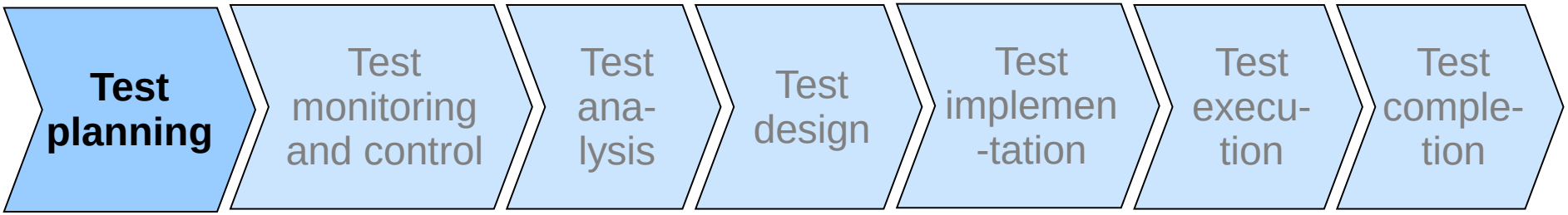


Test activities

In following for each test activity will be described the related

- tasks
- working products





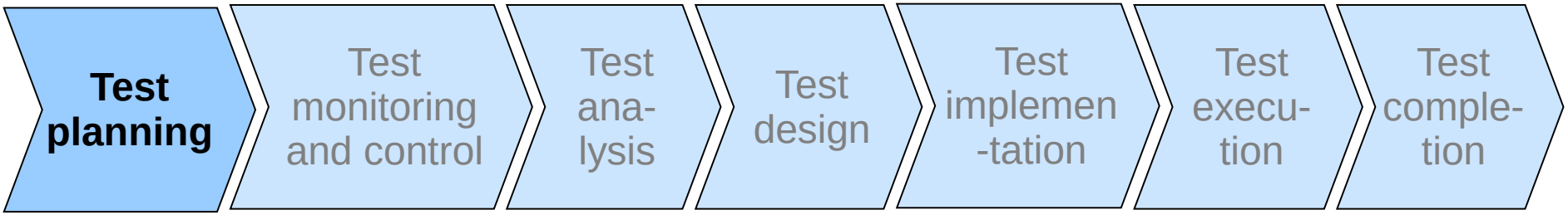
Test planning: The activity of establishing or updating a test plan.

- Tasks
  - Define the objectives of testing
  - Define the approach for meeting test objectives within constraints imposed by the context
  - Define ***Entry criteria***
    The set of conditions for officially starting a defined task
  - Define ***Exit criteria***
    The set of conditions for officially completing a defined task, like e.g. coverage criteria
  - Update test plans depending on feedback from monitoring and control activities.

Agile projects: Definition of ready

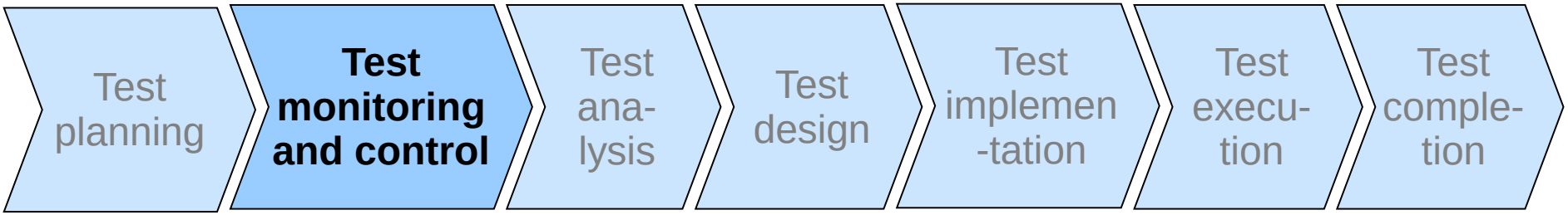
Agile projects: Definition of done

- ***Coverage***: The degree to which specified coverage items have been determined or have been exercised by a test suite expressed as a percentage.
- ***Coverage item*****:** An attribute or combination of attributes that is derived from one or more test conditions by using a test technique that enables the measurement of the thoroughness of the test execution.



- Working products
  depending on size of project and context:
  - ***Master test plan***
    A test plan that is used to coordinate multiple test levels or test types.
  - ***Test plan***
    Documentation describing the test objectives to be achieved and the means and the schedule for achieving them, organized to coordinate testing activities

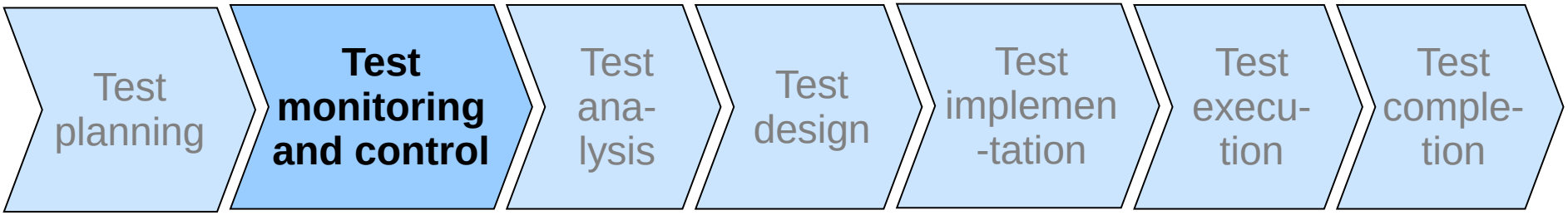
5.2



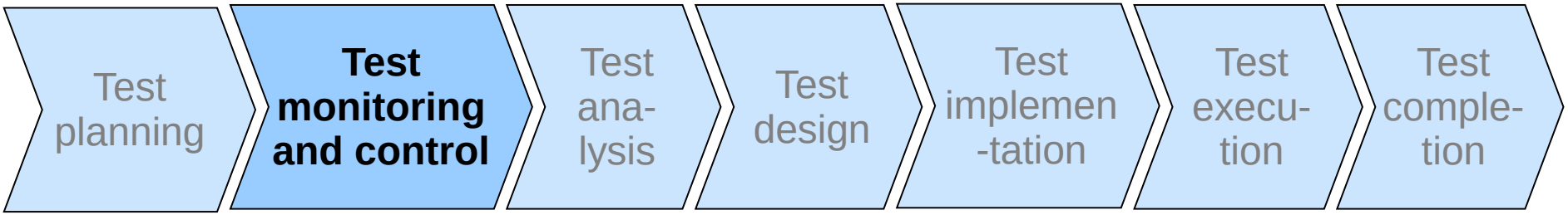
***Test monitoring*: The activity that checks the status of testing activities, identifies any variances from planned or expected, and reports status to stakeholders.**

Test control: The activity that develops and applies corrective actions to get a test project on track when it deviates from what was planned.

- Tasks
  - On-going comparison of actual progress against planned progress using test monitoring metrics as defined in the test plan.
  - Taking actions necessary to meet the objectives of the test plan
  - Evaluation of exit criteria
  - Communication of test reports, including deviations from the plan, information for decisions

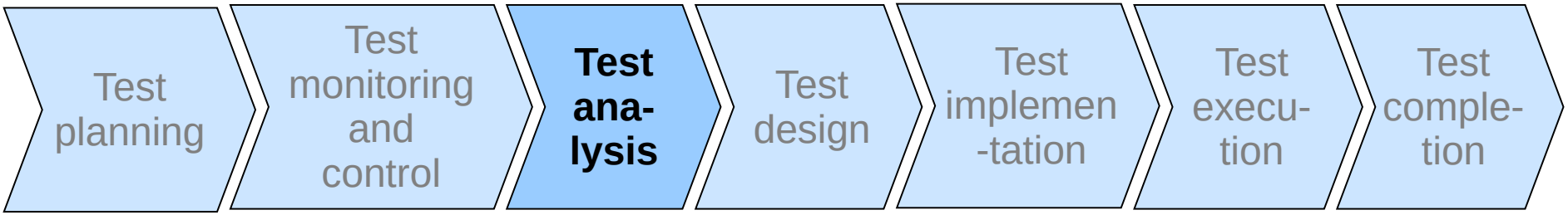


- Example for evaluation of exit criteria
  - Checking test results and logs against specified coverage criteria
  - Assessing the level of component or system quality based on test results and logs
  - Determining if more tests are needed (e.g., if tests originally intended to achieve a certain level of product risk coverage failed to do so, requiring additional tests to be written and executed)

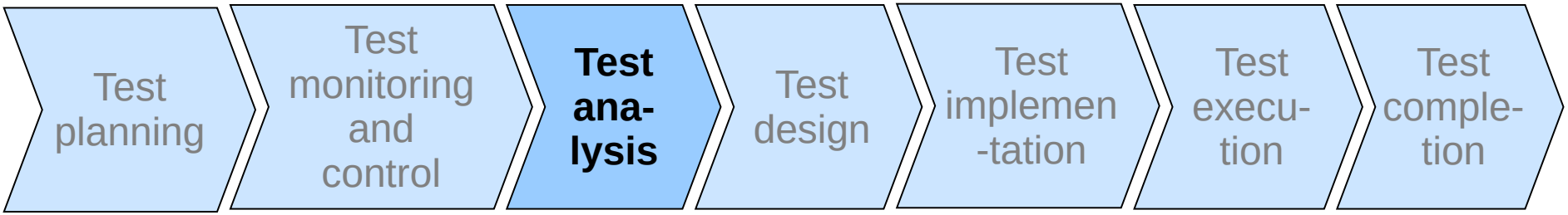


- Working products
  - ***Test report***
    Documentation summarizing test activities and results.
    Differ between
    - ***Test progress report***
      A type of test report produced at regular intervals about the progress of test activities against a baseline, risks, and alternatives requiring a decision
    - ***Test summary report***
      A type of test report produced at completion milestones that provides an evaluation of the corresponding test items against exit criteria.
- Address also project management concerns like task completion, resource allocation and usage, and effort.

5.3



- ***Test analysis***: The activity that identifies test conditions by analyzing the test basis.
  - ***Test conditions***:
    A testable aspect of a component or system identified as a basis for testing.
  - ***Test basis***:
    The body of knowledge used as the basis for test analysis and design.

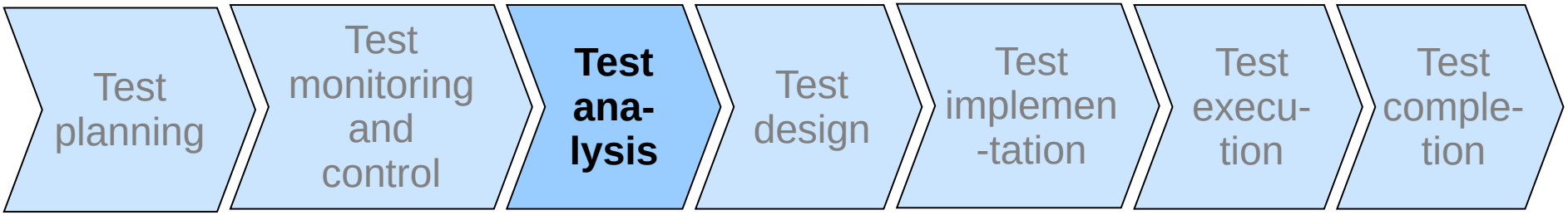


- Tasks
  - Determine „what to test“ focusing on measurable coverage criteria.
  - Analyze the test basis, i.e.
    - Requirements specification like business scenarios, use cases, user stories
    - Design and implementation information like software architecture, interface specifications
    - Implementation of the system including code, database metadata
    - Risk analysis reports consider functional, non-functional, and structural aspects

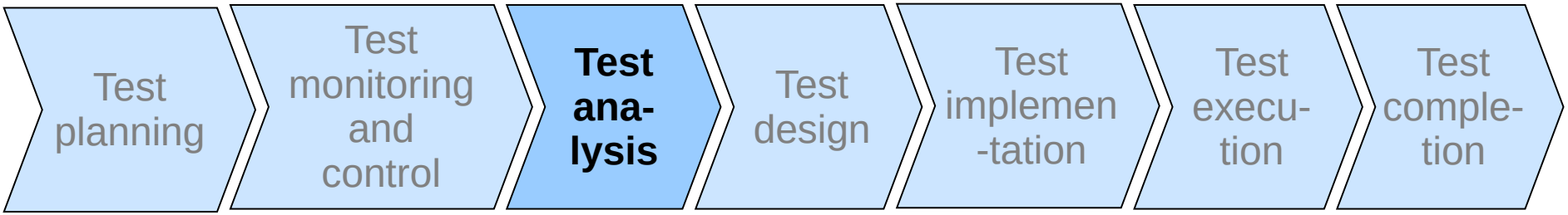
- Tasks
  - Evaluate the test basis to find defects, i.e.
    - Ambiguities
      Example: The last 10 bookings and cancellations of the customer are displayed in the window.
      Examples for ambiguous words: *quick, fast, user-friendly, easy, sufficient, adequate*
    - Omissions
      Example: “The system shall display the user’s defined bookmarks in a collapsible hierarchical tree structure.”
      Change to:
      “The system shall display the user’s defined bookmarks in a collapsible and expandable hierarchical tree structure.”
    - Inconsistencies
      Example: (1) The user types a country in an entry field.
      (2) After the user has chosen a country in a list field, an entry field for city pops up.

Potential benefit

Sources: Helmut Balzert: Lehrbuch der Softwaretechnik, Springer, 2009
https://www.jamasoftware.com/blog/five-ways-ambiguous-language-will-ruin-your-requirements//



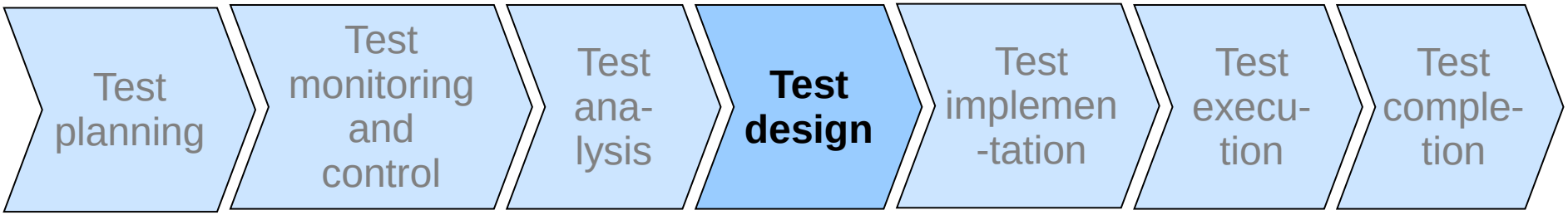
- Tasks
  - Evaluate the test basis to find defects, i.e.
    - Inaccuracies
      Example: The web page should respond fast.
    - Contradictions
      Example: (1) After login of an user the system connects automatically to the database
      (2) The user logins. If the user presses the [Connect] button, the system connects to the database
    - Superfluous statements
      Example: (1) On every page the company logo should appear
      (2) If the user cancels, a page opens with the company logo



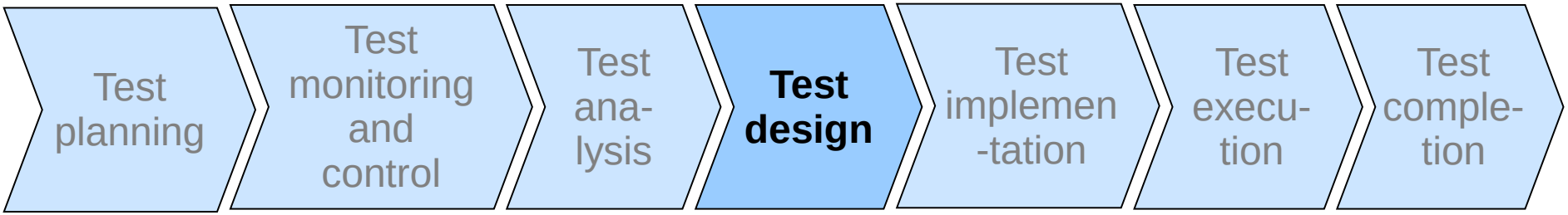
- Tasks
  - Identify features and sets of features to be tested
  - Define and prioritize test conditions for each feature based on analysis of the test basis
  - Ensure bi-directional traceability between each element of the test basis and the associated test conditions

- Working products
  - Prioritized test conditions
    each of which is ideally bi-directionally traceable to the specific element(s) of the test basis it covers.
  - ***Test charters*** (for exploratory testing)
    Documentation of the goal or objective for a test session.
  - Defect reports related to the test basis in case

4.4

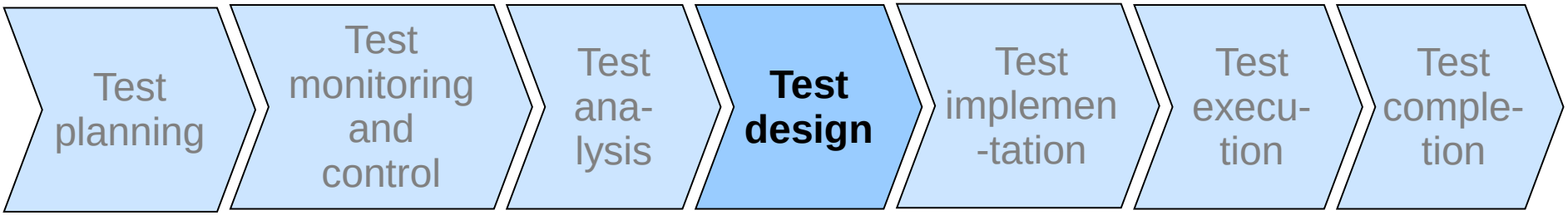


- ***Test design***: The activity that derives and specifies test cases from test conditions.
- ***Test case:*** A set of preconditions, inputs, actions (where applicable), expected results and postconditions, developed based on test conditions.
- Tasks
  - Determine „how to test“
  - Generate ***high-level test cases***:
    A test case without concrete values for input data and expected results.
  - Generate sets of high-level test cases
  - Ensure bi-directional traceability between the test basis, test conditions, and test cases

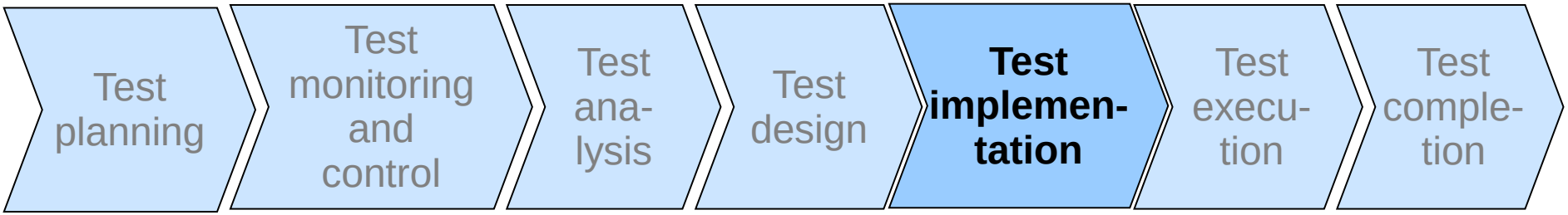


- Tasks
  - Identify necessary ***test data***: Data needed for test execution
  - Design the ***test environment***: An environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test.
    - ➢ identify any required infrastructure and tools
  - In case: Identify defects

Potential benefit

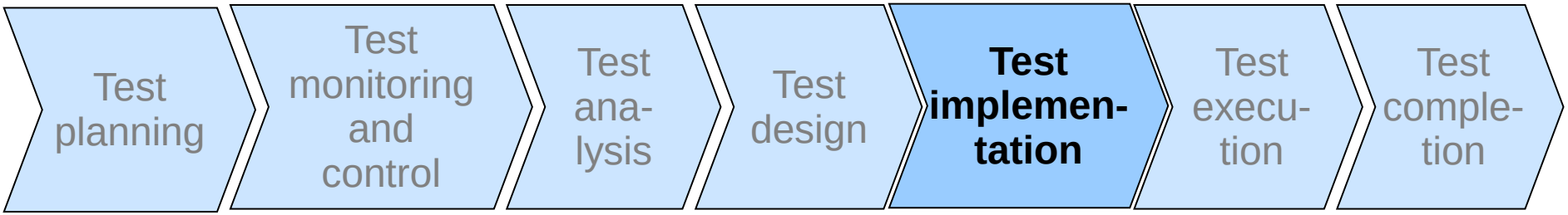


- Working products
  - Test cases (High-level)
  - Sets of test cases
  - Design and/or identification of the necessary test data
  - Design of the test environment
  - Identification of infrastructure and tools



- ***Test implementation***: The activity that prepares the testware needed for test execution based on test analysis and design
- Test design and test implementation tasks are often combined.

- ***Testware***: Work products produced during the test process for use in planning, designing, executing, evaluating and reporting on testing.



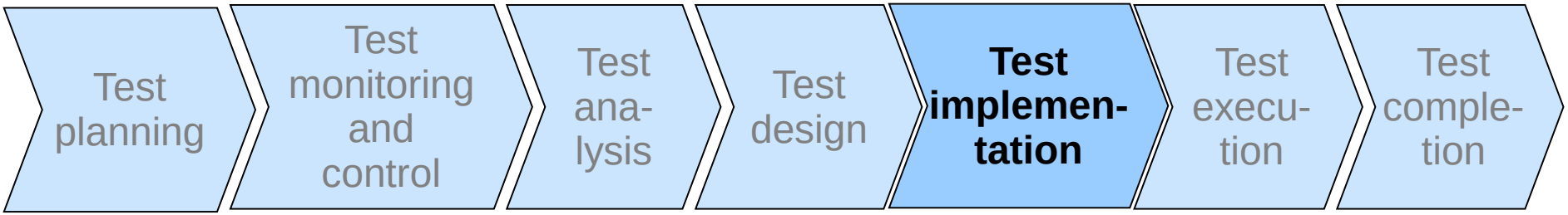
- Tasks
  - Determine: “Do we now have everything in place to run the tests?”
  - Create ***low-level test cases***: A test case with concrete values for preconditions, input data, expected results and postconditions and detailed description of actions (where applicable).

- Test case – Example

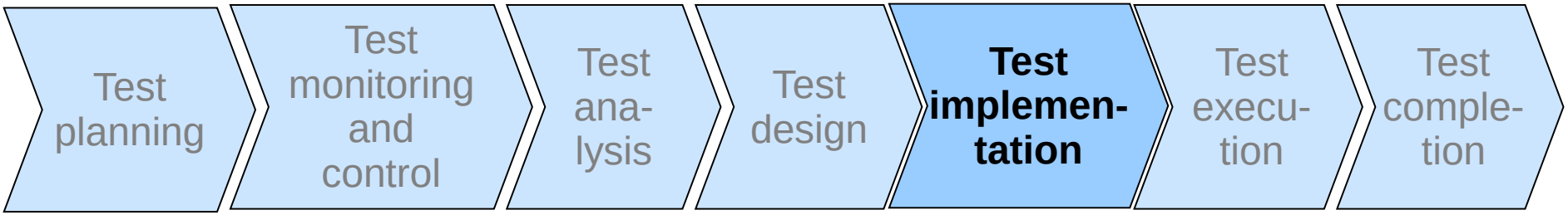
Test Case ID	SDM-0007
Test Case name	Create Student data set
Priority	critical
Description	Creation of a student data set, with name, first name, and enrollment number
Precondition	Enrollment number must already been given
Postcondition	Data are stored in database, table SDM_student
Links	SDM-0005

Test steps

No.	Activity	Test data	Expected result
10	Call SDM system		SDM system opens
20	Call "Add student"		Input masks open
30	Enter last name	Daowang	Data are shown
40	Enter first name	Rid	Data are shown
50	Enter enrollment number	4055-991	Data are shown
60	Press [submit]		System shows: Data are processed

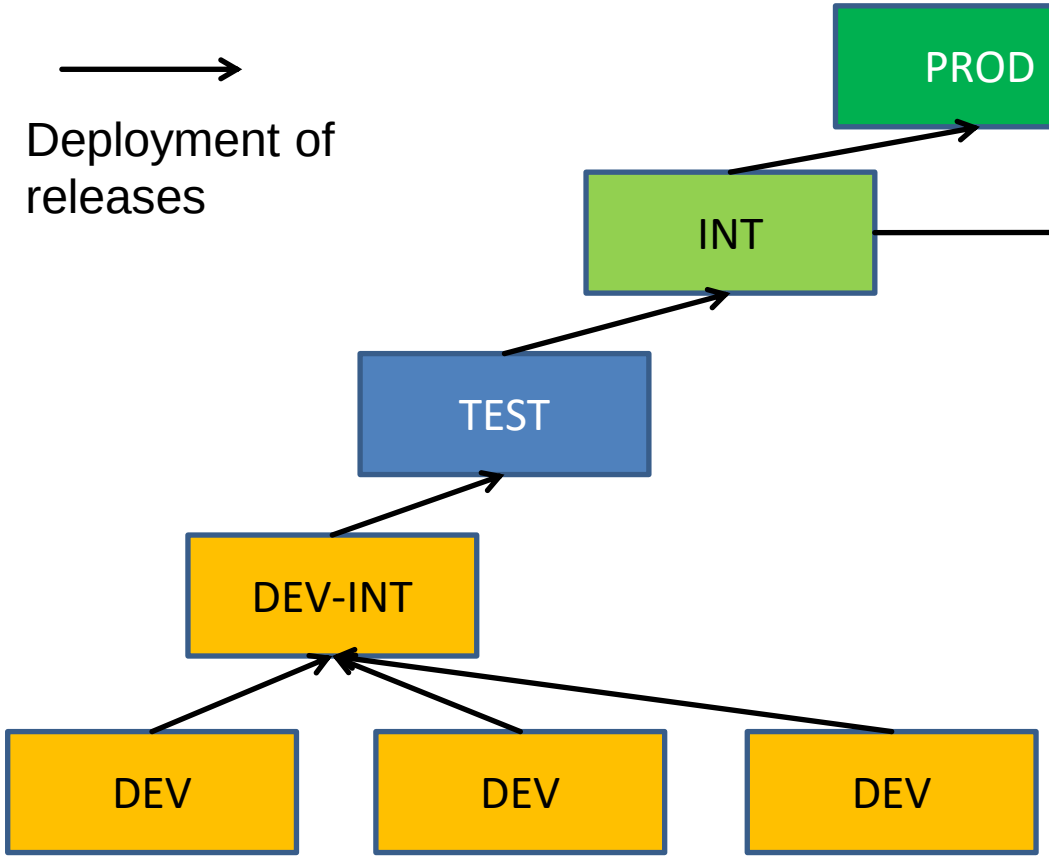


- Tasks
  - Develop and prioritize ***test procedures*** (***Synonym**: test script):* A sequence of test cases in execution order, and any associated actions that may be required to set up the initial preconditions and any wrap up activities post execution.
  - Create automated test scripts in case
  - Create ***test suites***: A set of test scripts or test procedures to be executed in a specific test run.
  - Create ***test execution schedules***: A schedule for the execution of test suites within a test cycle.

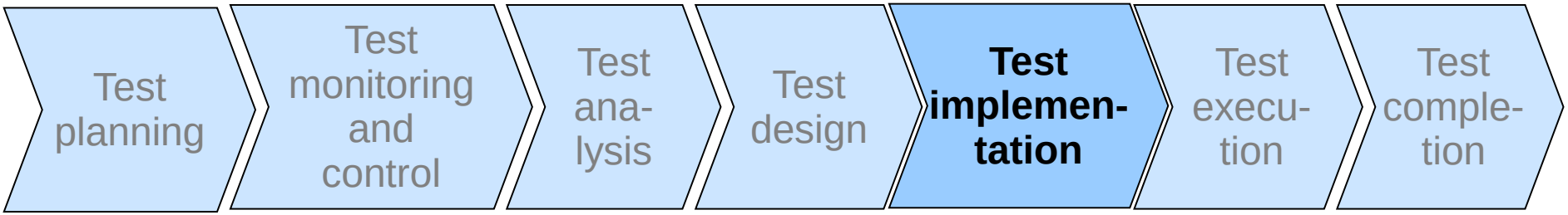


- Tasks
  - Build the test environment
    - Verify that everything needed has been set up correctly
  - Prepare test data and ensure the load in the test environment
  - Verify and update bi-directional traceability between the test basis, test conditions, test cases, test procedures, and test suites

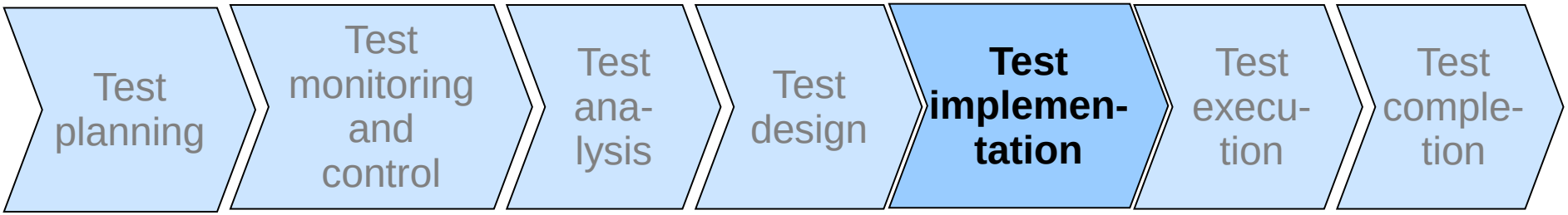
- Test environment – example



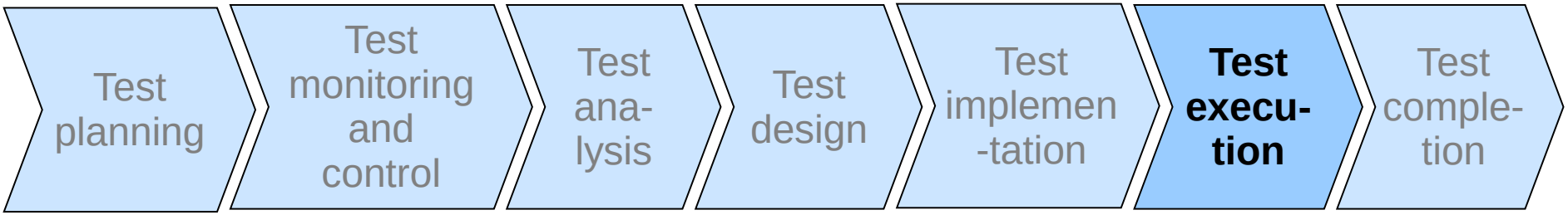
PROD = production environment
INT = Integration environment
INT2 = Integration environment – Copy of PROD
TEST = Test environment
DEV-INT = Development integration environment
DEV = Development environment for developers



- Working products
  - Test cases (Low-level)
  - Test procedures
  - Automated test scripts in case
  - Test suites
  - Test execution schedule
  - Test environment
  - Test conditions defined in test analysis may be further refined

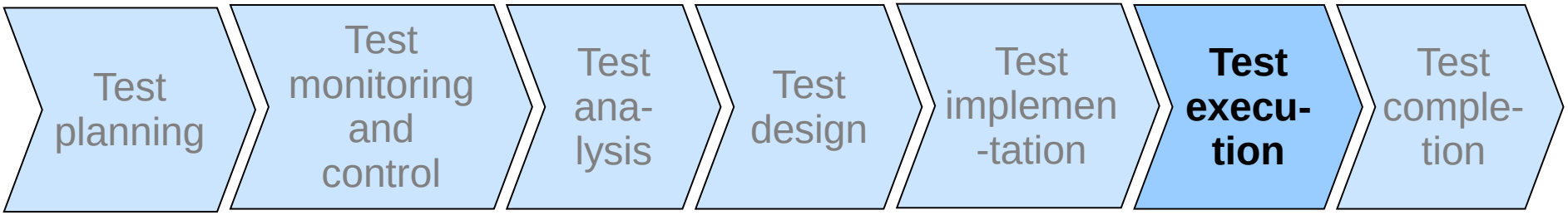


- Working products
  - Test data
    - All data needed for testing
    - Depending on project based on
      - business object data model or
      - physical data model
    - Artificial data or based on real business data, e. g. out of legacy systems → data generator
    - Which test data are included with delivery?
    - Feed of test data
    - Delete test data („naked system“, “database clean-up”)



Test execution: The activity that runs a test on a component or system producing actual results.

- During test execution, test suites are run in accordance with the test execution schedule.

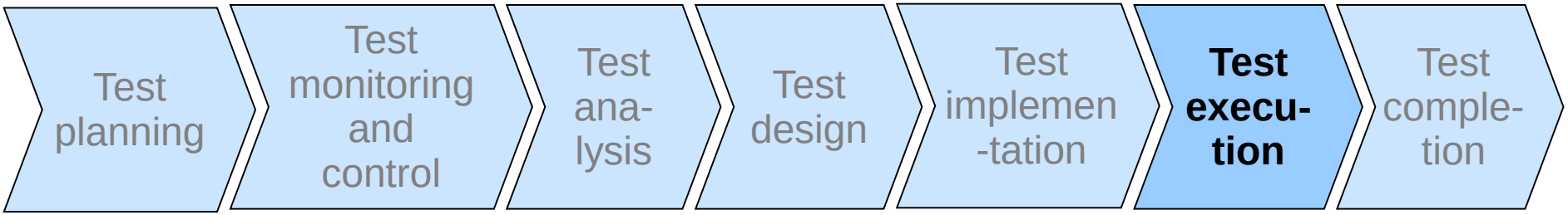


- Tasks
  - Record the IDs and versions of the test item(s) or test object, test tool(s), and testware
  - Execute tests either manually or by using test execution tools
  - Compare actual results with expected results
  - Analyze anomalies to establish their likely causes, e.g.,
    - failures may occur due to defects in the code,
    - but false positives also may occur *("It's not a bug ,it's a feature")*
  - Report defects based on the failures observed
  - Logging the outcome of test execution

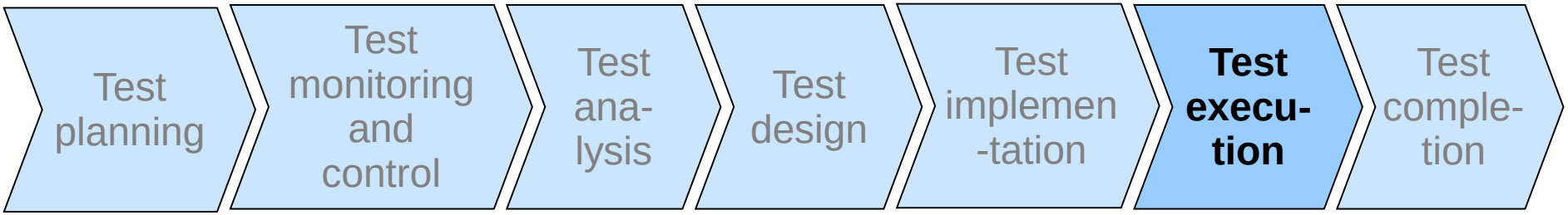
1.2

5.6

Unexecuted | blocked | In progress | pass | fail

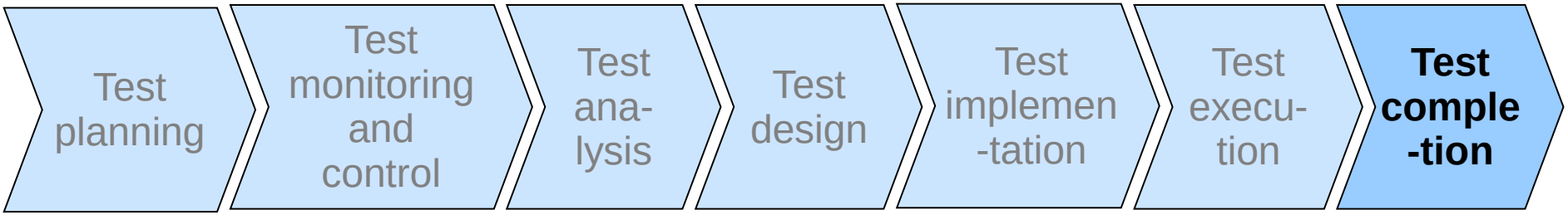


- Tasks
  - Repeat test activities either as a result of action taken for an anomaly, or as part of the planned testing, e.g.,
    - execution of a corrected test,
    - confirmation testing, and/or
    - regression testing
  - verify and update bi-directional traceability between the test basis, test conditions, test cases, test procedures, and test results.



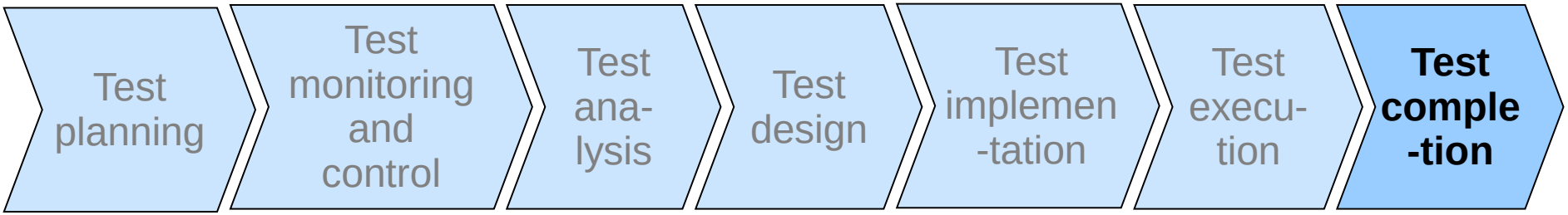
- Working products
  - Documentation of the status of individual test cases or test procedures (e.g., ready to run, pass, fail, blocked, deliberately skipped, etc.)
  - Defect reports
  - Documentation about which test item(s), test object(s), test tools, and testware were involved in the testing
- Goal: After completion of the test execution the status of each element of the test basis is reported via bi-directional traceability to give transparency if coverage criteria have been met: like for example:
  - requirements that passed their tests,
  - requirements that failed their tests and/or have defects associated with them,
  - requirements that have pending tests.

5.6

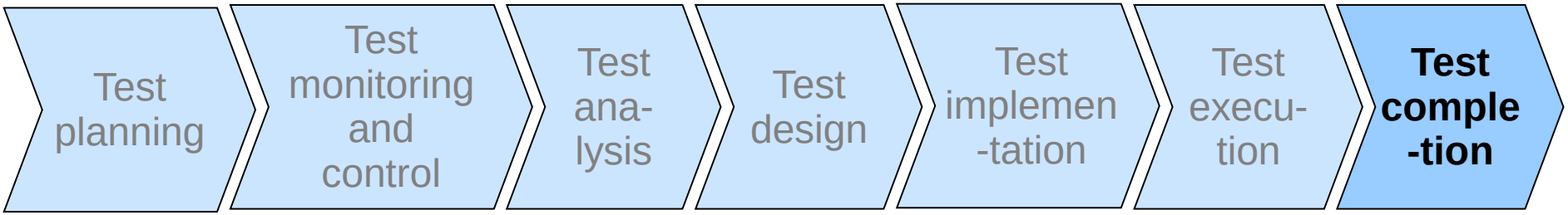


Test completion: The activity that makes testware available for later use, leaves test environments in a satisfactory condition and communicates the results of testing to relevant stakeholders.

- Test completion activities occur at project milestones such as when
  - a software system is released,
  - a test project is completed (or cancelled),
  - an Agile project iteration is finished,
  - a test level is completed, or
  - a maintenance release has been completed



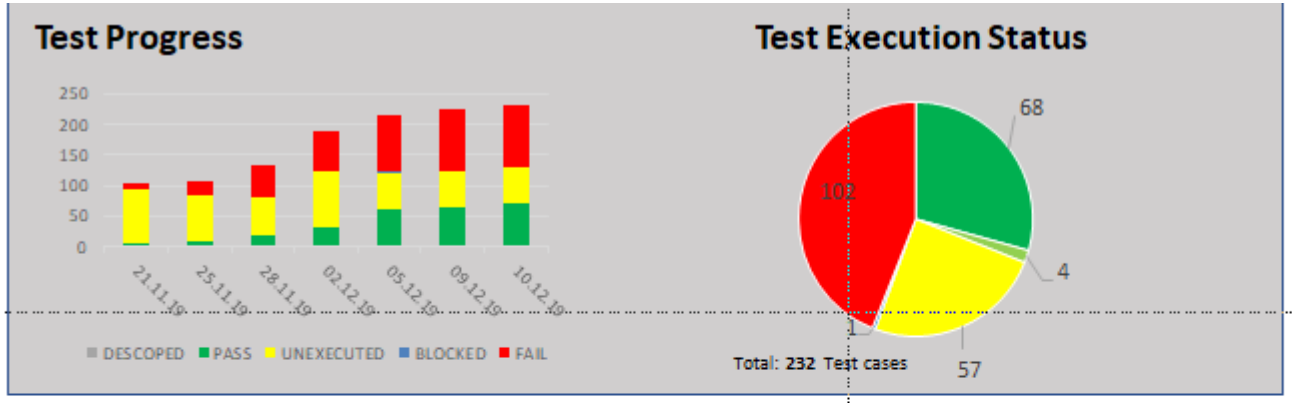
- Tasks
  - Create a test summary report for stakeholders
  - Lessons learned for future iterations, releases, and projects
  - Hand over the testware to the maintenance teams, other project teams, and/or other stakeholders who could benefit from its use
  - Close defects reports
  - Enter change requests or product backlog items for any defects that remain unresolved
  - Finalize and archive the test environment, the test data, the test infrastructure, and other testware for later reuse
  - Use the information gathered to improve test process maturity

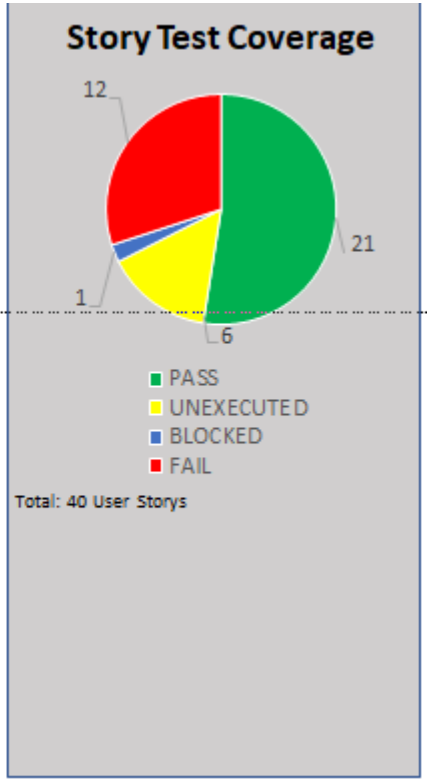


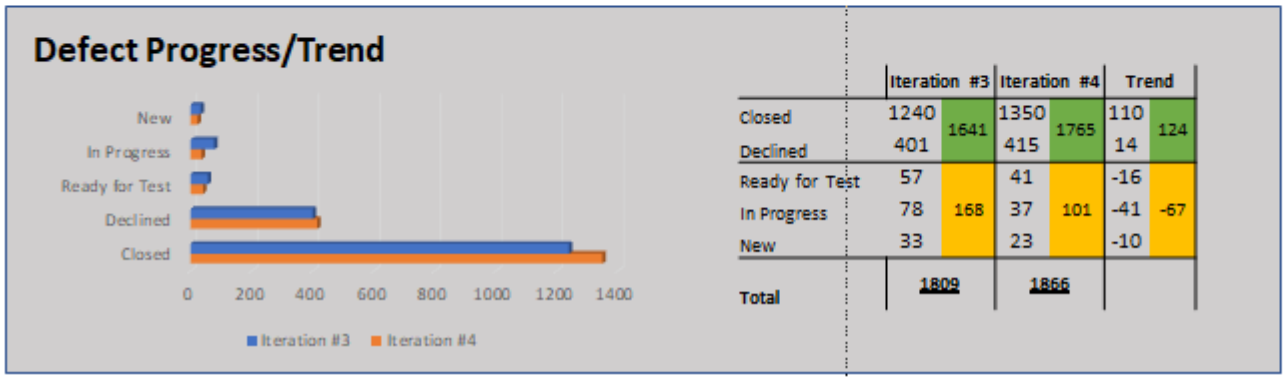
- Working products
  - test summary reports,
  - action items for improvement of subsequent projects or iterations,
  - change requests or product backlog items,
  - finalized testware.

- Example for a test report

Iteration 29 SDM test dashboard



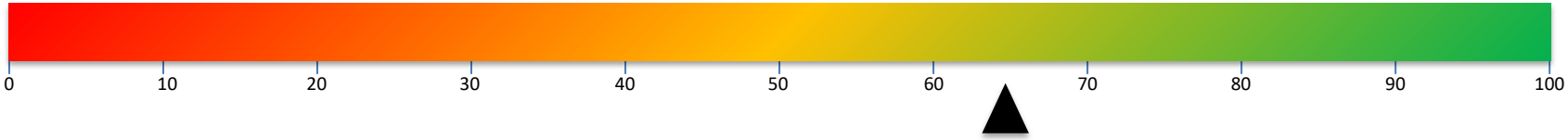




	Iteration #3		Iteration #4		Trend	
Closed	1240	1641	1350	1765	110	124
Declined	401		415		14	
Ready for Test	57	168	41	101	-16	-67
In Progress	78		37		-41	
New	33		23		-10	
Total	1809		1866			

- Example for a test report

Iteration 29 SDM quality characteristics



Overall rating: Iteration 29: Satisfactory (65 %)
Iteration 28: Satisfactory (65 %)

Functional:
1. Loading data
Several loads stuck, results in 6 open defects; **measures:** defects will be fixed/retested, additional story will be implemented
2. Export
19 defects, 1 critical; **measures:** defects will be fixed/retested
…

Non-functional:
1. Usability
Several issues reported by end-user,; **measures:** usability workshop
2. Security
13 defects, 2 critical; **measures:** critical defects will be fixed/retested
3. Supportability:
Operation processes are established
4. Efficiency:
Performance test could not be performed ; **measures:** task force established
…

Traceability between the Test Basis and Test Work Products

Traceability: The degree to which a relationship can be established between two or more work products.

- Main goal: evaluation of test coverage
- For effective test monitoring and control: establish and maintain traceability throughout the test process between
  - each element of the test basis and
  - the various test work products associated with that element

Traceability between the Test Basis and Test Work Products

- Good traceability supports
  - Analyzing the impact of changes
  - Making testing auditable
  - Meeting IT governance criteria
  - Improving the understandability of test reports, for example showing requirements and their test result
  - Communication of test results to stakeholders
  - Providing information to assess product quality, process capability, and project progress against business goals

Traceability between the Test Basis and Test Work Products

Example: Mobile application

- Test basis contains
  - a list of requirements
    → Each requirement is an element
  - a list of supported mobile devices
    → Each supported device is an element
- Coverage criteria
  - require at least one test case for each element of the test basis.
  - can act effectively as key performance indicators (KPIs)
  - show achievement of software test objectives
- Once executed, the results of these tests tell stakeholders if
  - specified requirements are fulfilled
  - failures were observed on supported devices

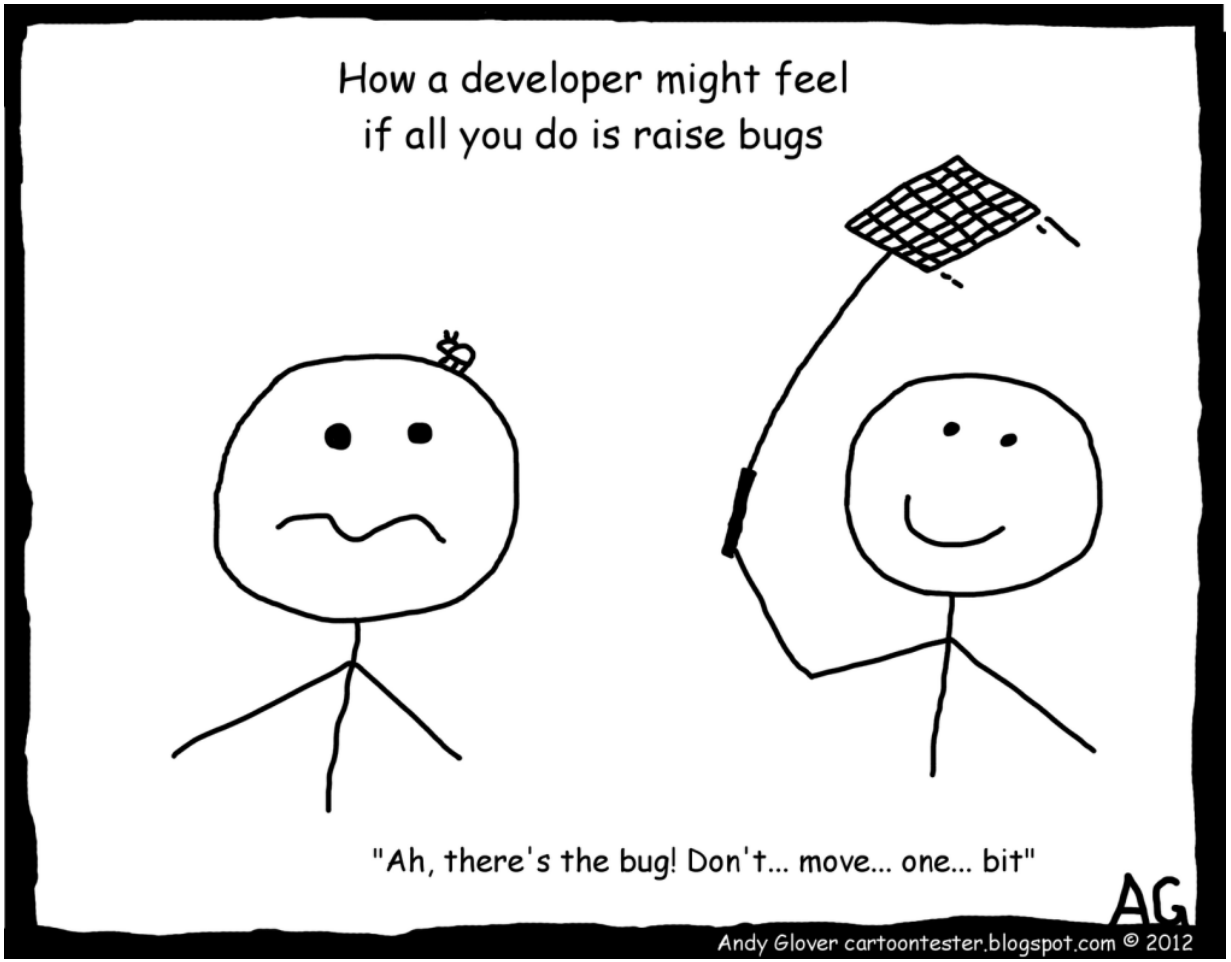
1.1

Summary

- Many different test processes are possible
- A test process consists of the following main groups of activities:
  - Test planning
  - Test monitoring and control
  - Test analysis
  - Test design
  - Test implementation
  - Test execution
  - Test completion
- Traceability supports effective test monitoring and control as well as reporting to stakeholders with linking each element of the test basis with various test products

Contents

Human Psychology and Testing



Human Psychology and Testing

- Communication problems may occur, particularly if testers are seen only as messengers of unwanted news about defects.
- However, there are several ways to improve communication and relationships between testers and others ...

Human Psychology and Testing

- Start with collaboration rather than battles. Common goal of everyone: Better quality systems
- Communicate findings on the product in a neutral, fact-focused way, e. g. reproducible defect descriptions
- Write objective and factual incident reports and review findings.
- Do not criticize the person who created it.
- Try to understand how the other person feels and why they react as they do.
- Confirm that the other person has understood what you have said and vice versa.

Tester’s and Developer’s Mindsets

- Errare humanum est (English: To err is human)
  … who admits?
- Development = constructive
  Testing = ?
- Is it good for a developer to test his own program?
  What do you think?

Tester's and Developer's Mindsets

Developer

- Successful if no defects found
- "Testing" to show program is working
- Blinkered in own work

Tester

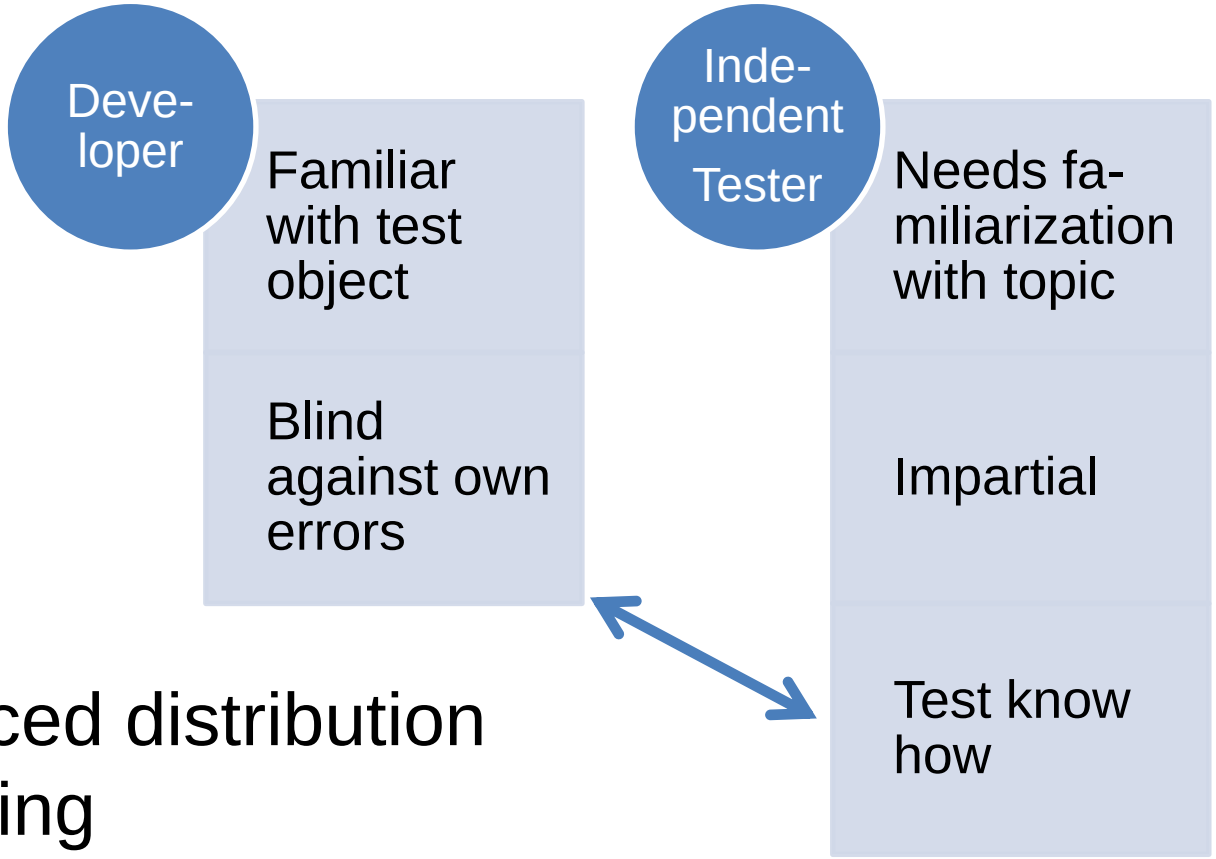
- Successful if defects are found
- Testing to evaluate
- „Bean counter"

Tester’s and Developer’s Mindsets

- The mindset to be used while developing software is different from that used while testing and reviewing.
- With the right mindset:
  Developers are able to test their own code.
- A certain degree of independence (avoiding the author bias) often makes the tester more effective at finding defects and failures.
- Independence is not a replacement for familiarity

Tester's and Developer's Mindsets

- Who should test?



Idea:
Balanced distribution of testing

Summary

- Tester need good communication skills
  - Common goal: Good quality!
  - Working results should be communicated in a neutral, fact-focused way
- Mindsets of testers and developers concerning the test object are different
  A higher level of product quality could be achieved in bringing them together