

Software Testing

Foundations of Software Engineering

FSE v2023.1

Aleksandr Mikhalev, Fall 2023

Lecture Outline

§1. Why software testing [10 min]

- 1.1. Verification and validation
- 1.2. Incompleteness of testing

§2. Levels of software testing [10 min]

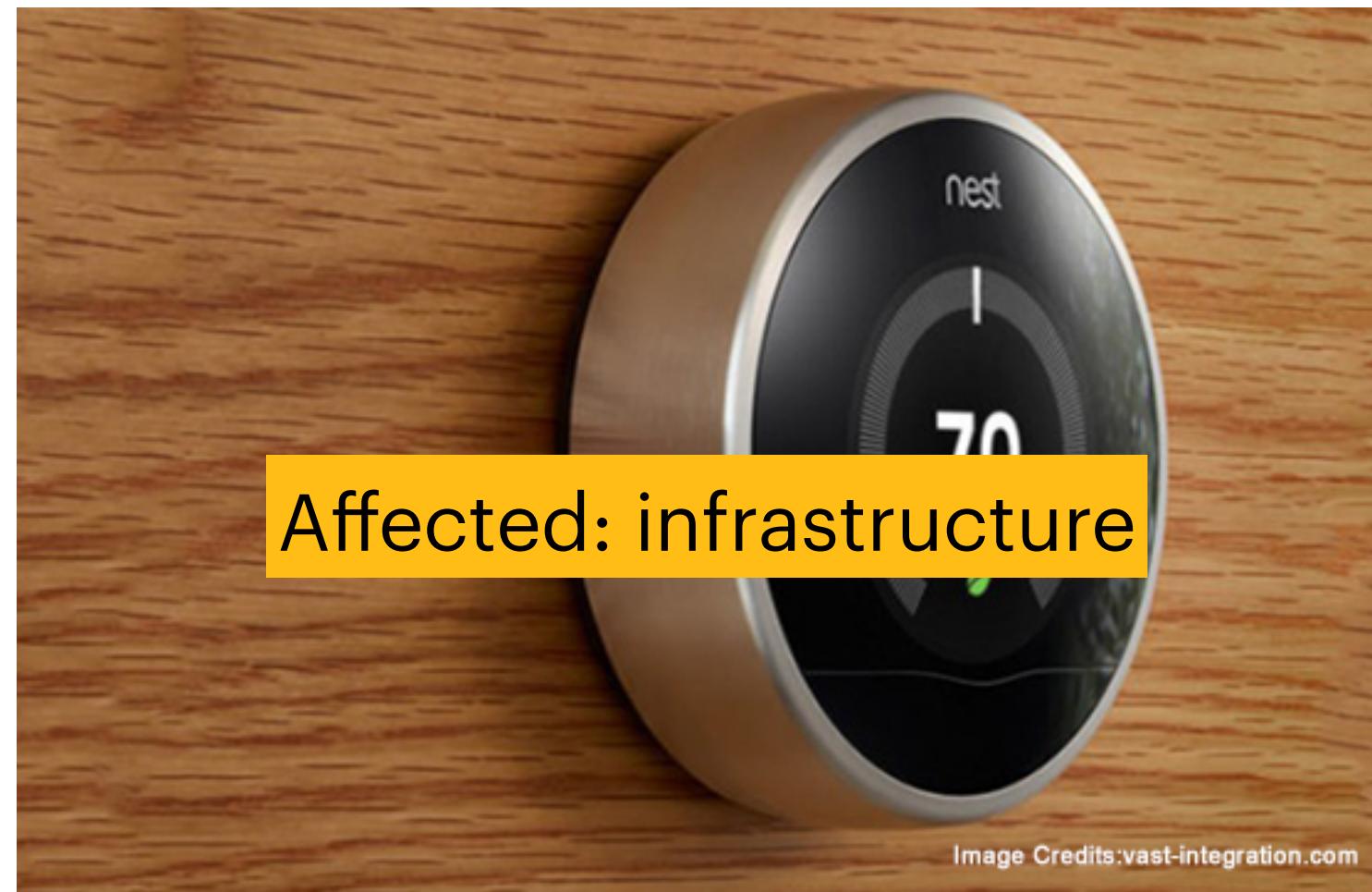
- 2.1. Unit testing, component testing, and system testing
- 2.2. Integration testing and regression testing
- 2.3. User acceptance testing

§3. The software testing process [5 min]

- 3.1. Automated testing

§1. Why software testing?

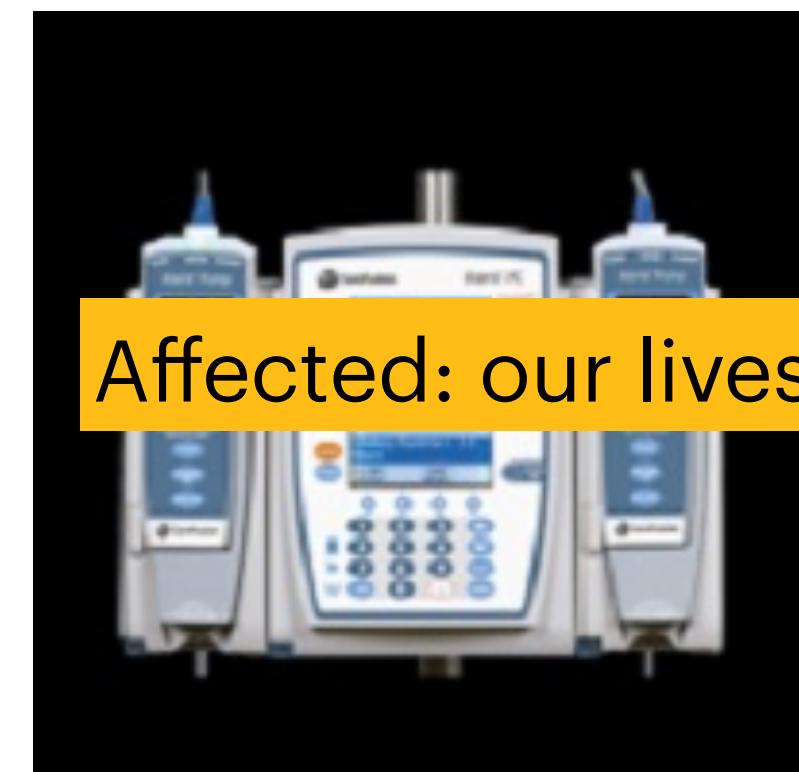
§1. Why software testing?



Bugs in software
updates



Bugs at runtime
(financial services)



Bugs at runtime
(critical software)

Verification and Validation

§1. Why software testing?

1.1. Verification and validation

- **Goal:** make sure that software is meeting the user's needs.
- **Verification:** “Are we building the software *right*? ”
 - Conformity to software requirements specification
- **Validation:** “Are we building *the right product*? ”
 - Conformity to real user requirements

§1. Why software testing?

1.1. Verification and validation

- Understand and validate software requirements
- Verification and validation for:
 - Inspections
 - Design discussions
 - Static analysis
 - Testing
 - Runtime verification



- Study a piece of code to see whether it meets the requirements and works correctly under all circumstances
- Check the whole system
- Accurately assess some system behaviors
- Necessary to “accept” the system by customers
- Document expected system behavior

Incompleteness of software testing

§1. Why software testing?

1.2. Incompleteness of testing

- **Diminishing returns:** finding each next bug is more and more difficult
- **Deadlines:** fix high-profile bugs, let others remain
- **Consequences:** fixing a bug might force you make other according changes
- **Frequency of updates:** fix security issues immediately, serious flaws monthly, minor flaws two times a year, and nice-to-have new features yearly
- **Usefulness:** “Any sufficiently advanced bug is indistinguishable from a feature.”
- **Obsolescence:** don’t fix bugs for features that are expected to go away
- **Not a bug:** train users to avoid misunderstanding

§1. Why software testing?

1.2. Incompleteness of testing

- Only sample a set of possible behaviors from the system's state space
- State space is discrete and discontinuous:
 - No basis for extrapolating from seen cases to unseen ones

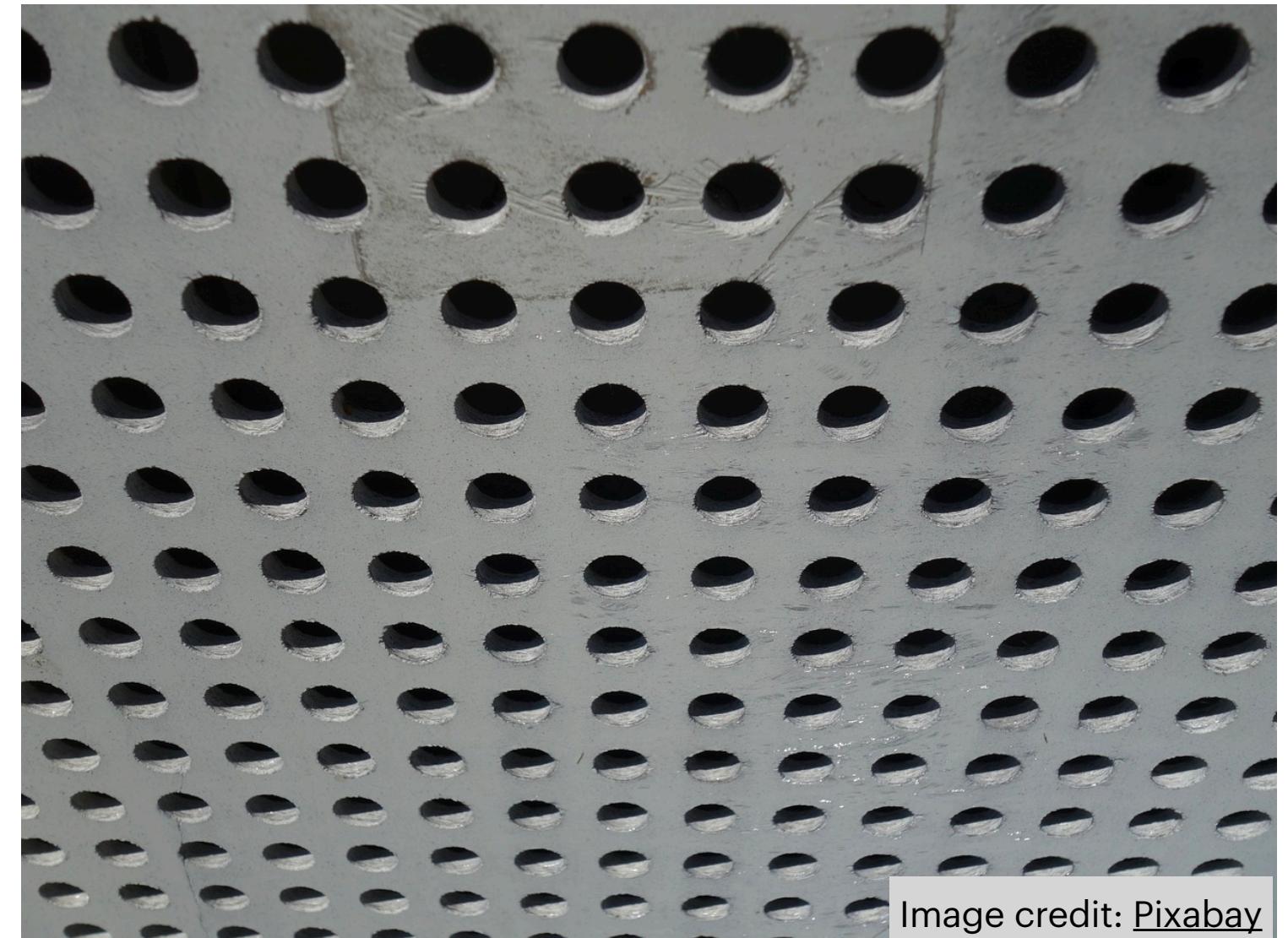
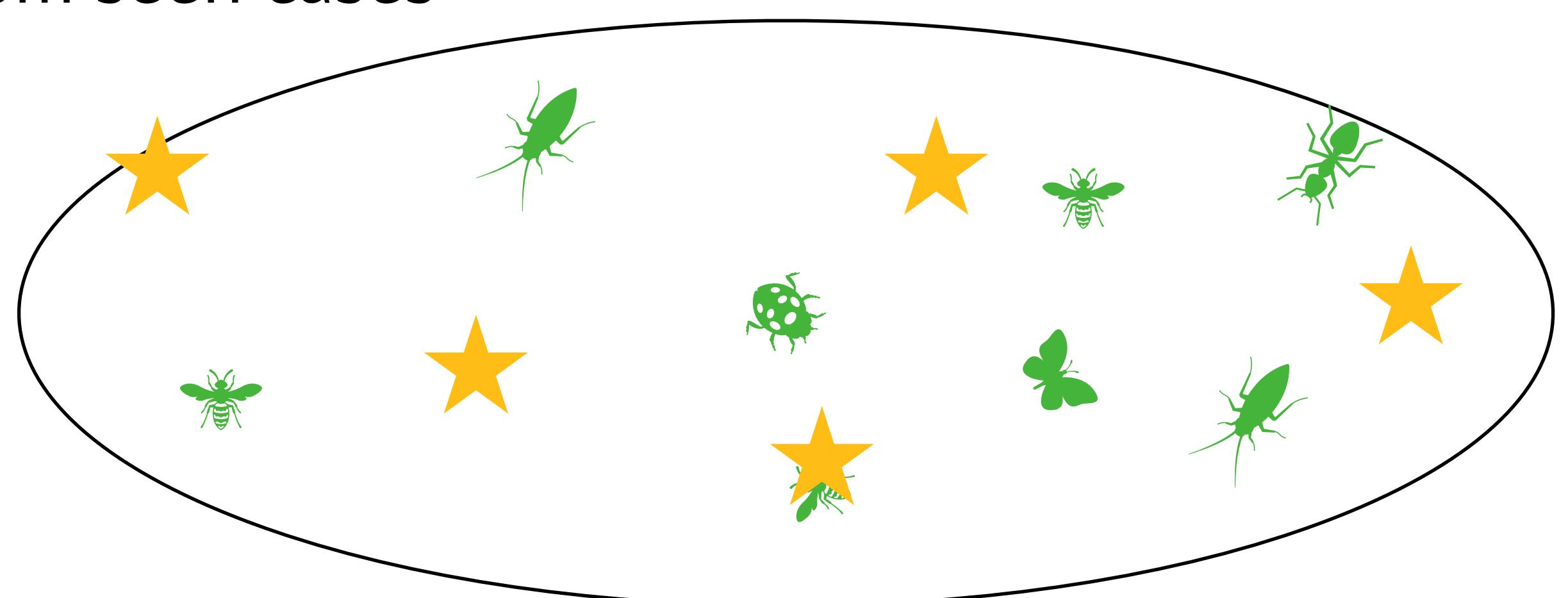


Image credit: [Pixabay](#)



**“Testing can only show the presence of errors,
not their absence”**

Edsger Dijkstra, 1972

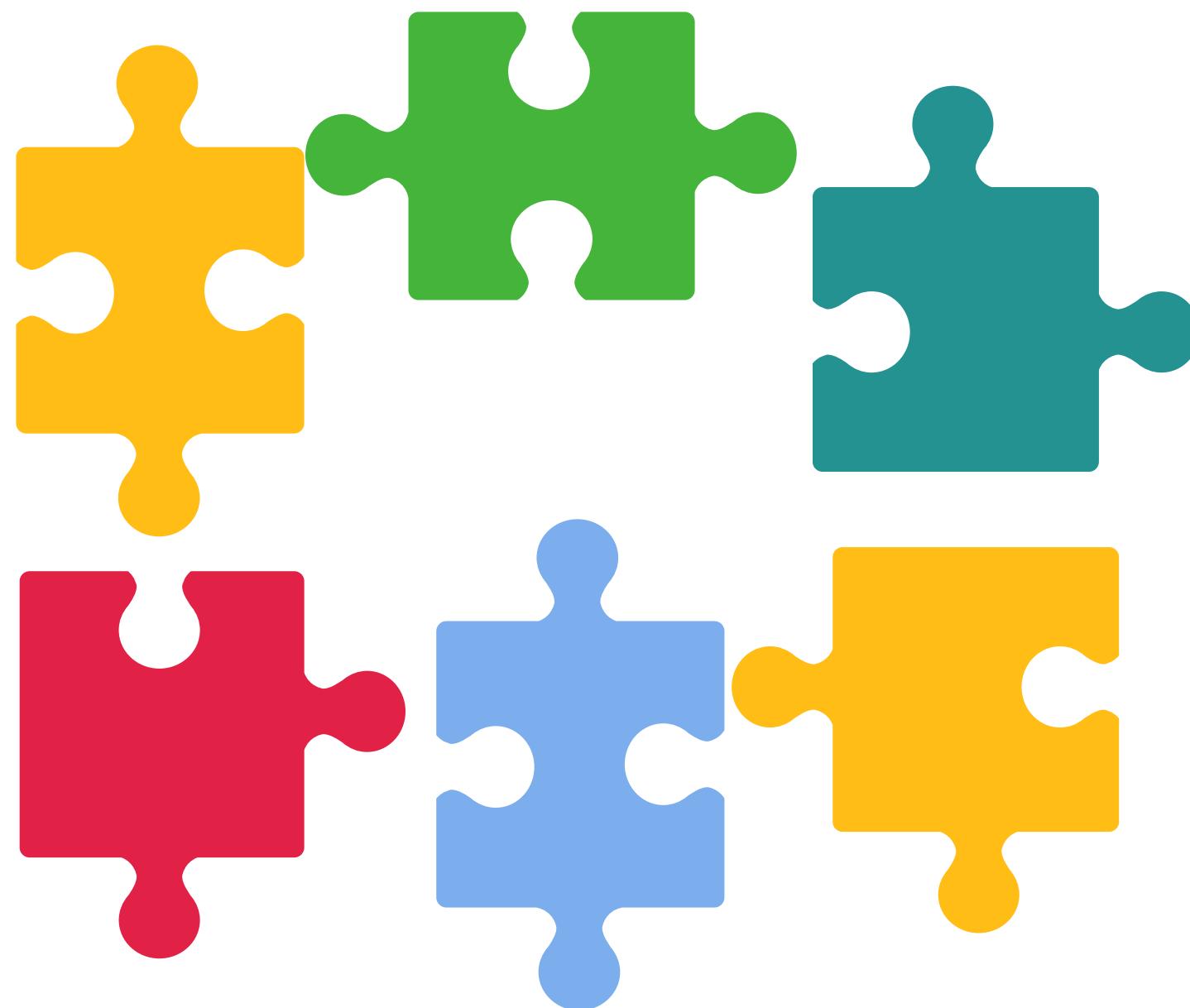
§2. Levels of software testing

Unit, Component, and System Testing

§2. Levels of software testing

2.1. Unit testing

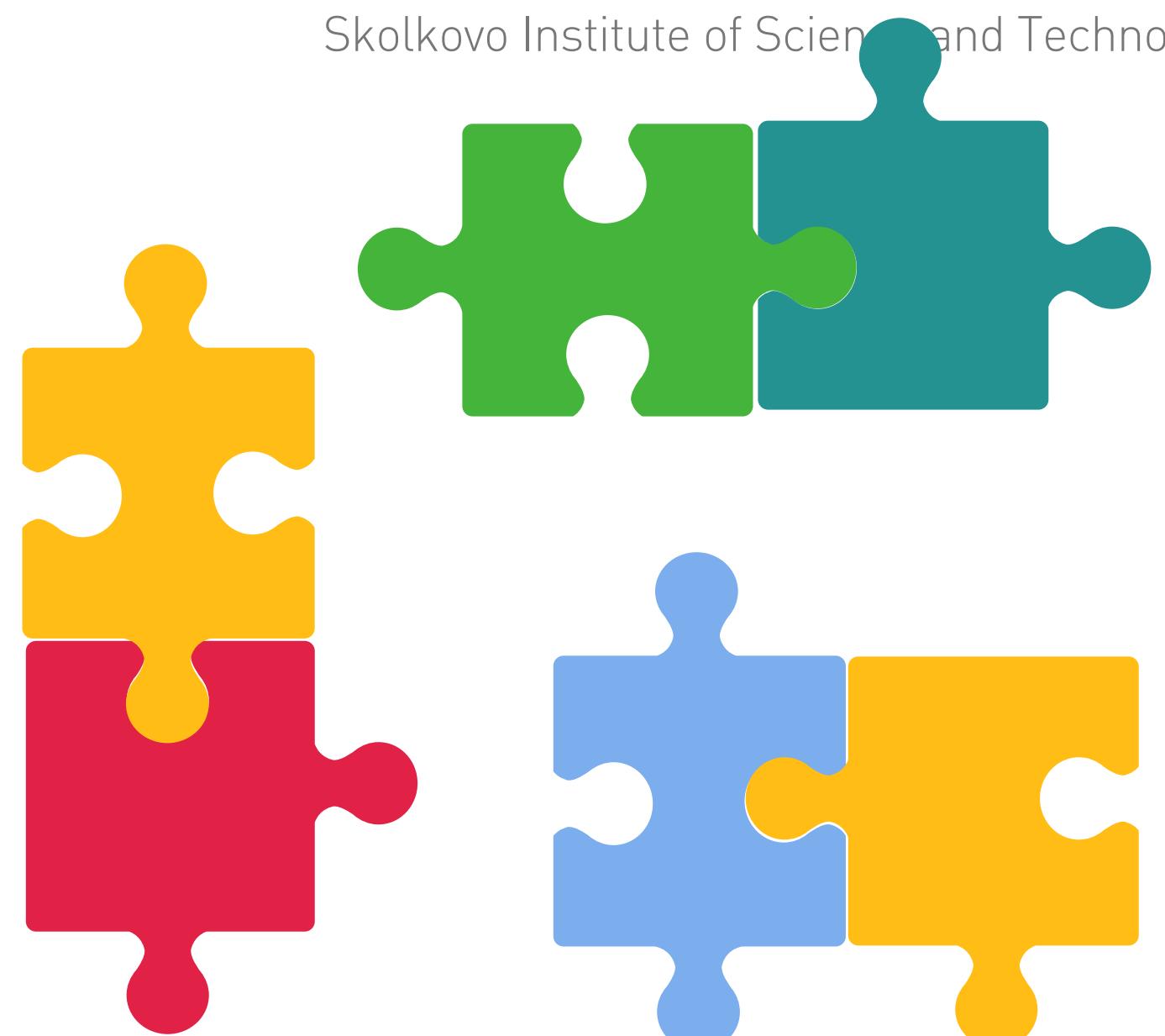
- **Targets:** individual functions or methods
- **How to test:** calls to these routines with different input parameters
- **What to test:**
 - test all operations associated with the object;
 - set and check the value of all attributes associated with the object
 - put the object into all possible states



§2. Levels of software testing

2.1. Component testing

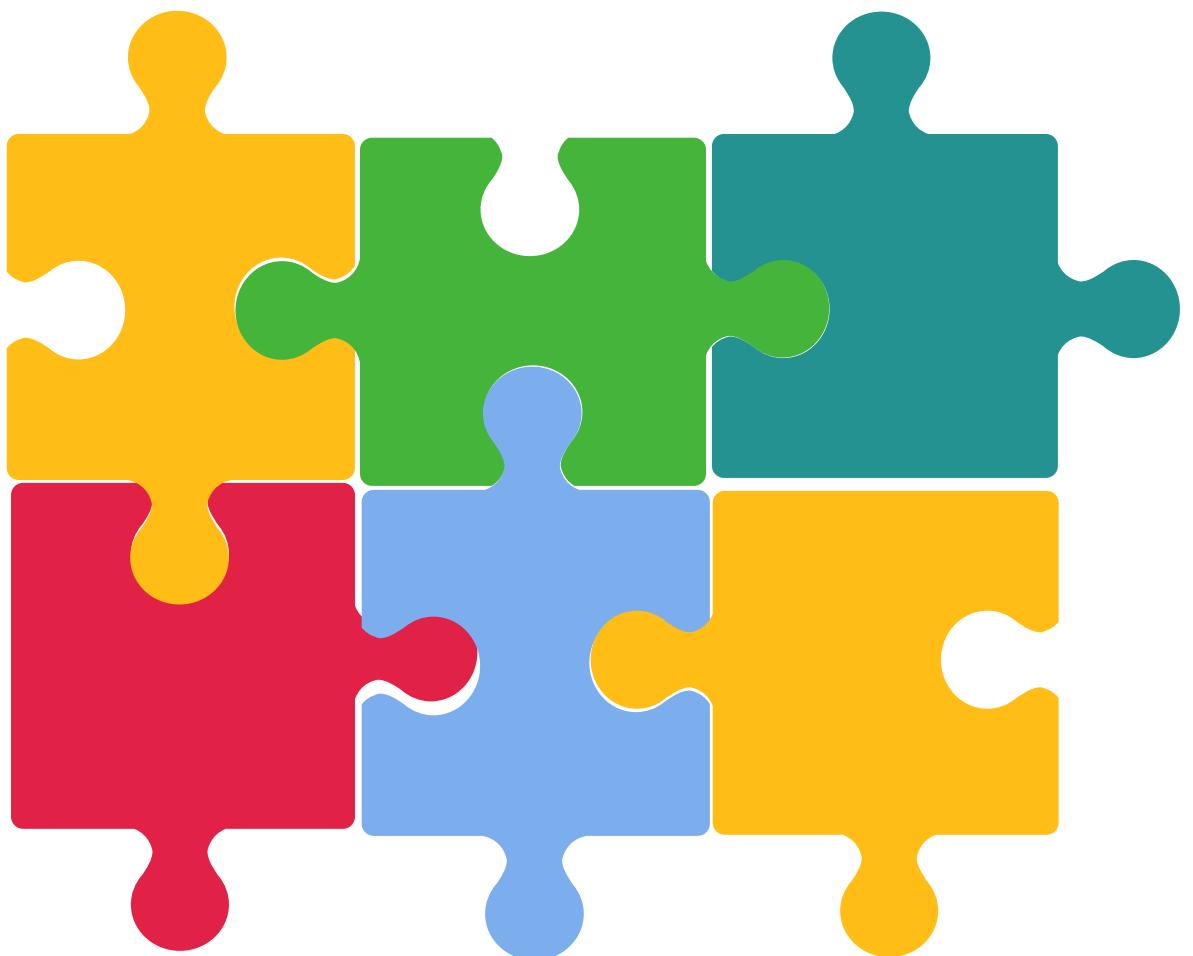
- **Targets:** interactions and interfaces between components
- **How to test:** calling the interface of the composite component
- **What to test:**
 - each call to an external component
 - test the interface with null pointer parameters
 - design tests that deliberately cause the component to fail
 - stress testing in message passing systems
 - design tests that vary the order in which these shared memory components are activated



§2. Levels of software testing

2.1. System testing

- **Targets:** testing the interactions between the components and objects that make up a system
- **How to test:** use case-based testing
- **What to test:**
 - all functions accessible through menus
 - combinations of functions accessed from the same menu
 - functions taking user input (correct or incorrect)

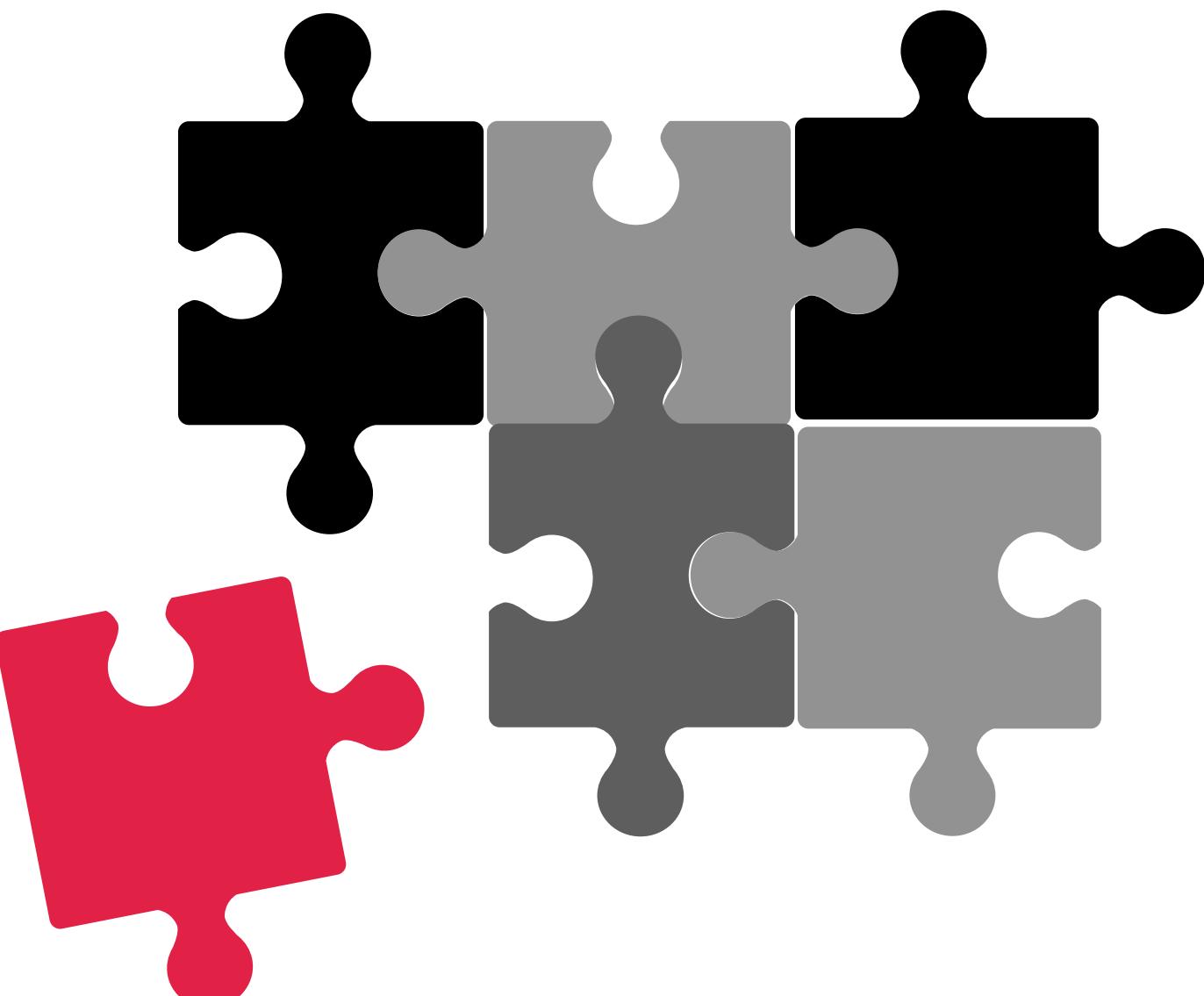


Integration and Regression Testing

§2. Levels of software testing

2.2. Integration testing and regression testing

- **Integration testing:** verify that the newly introduced code interacts well with the existing codebase
- **Regression testing:** discover bugs introduced by merging the new code into the codebase
- **Continuous integration / testing:** run integration/regression tests every time an increment is complete (in the limit: every commit to VCS)



CI  **CD**

User Acceptance Testing

§2. Levels of software testing

2.3. User Acceptance Testing

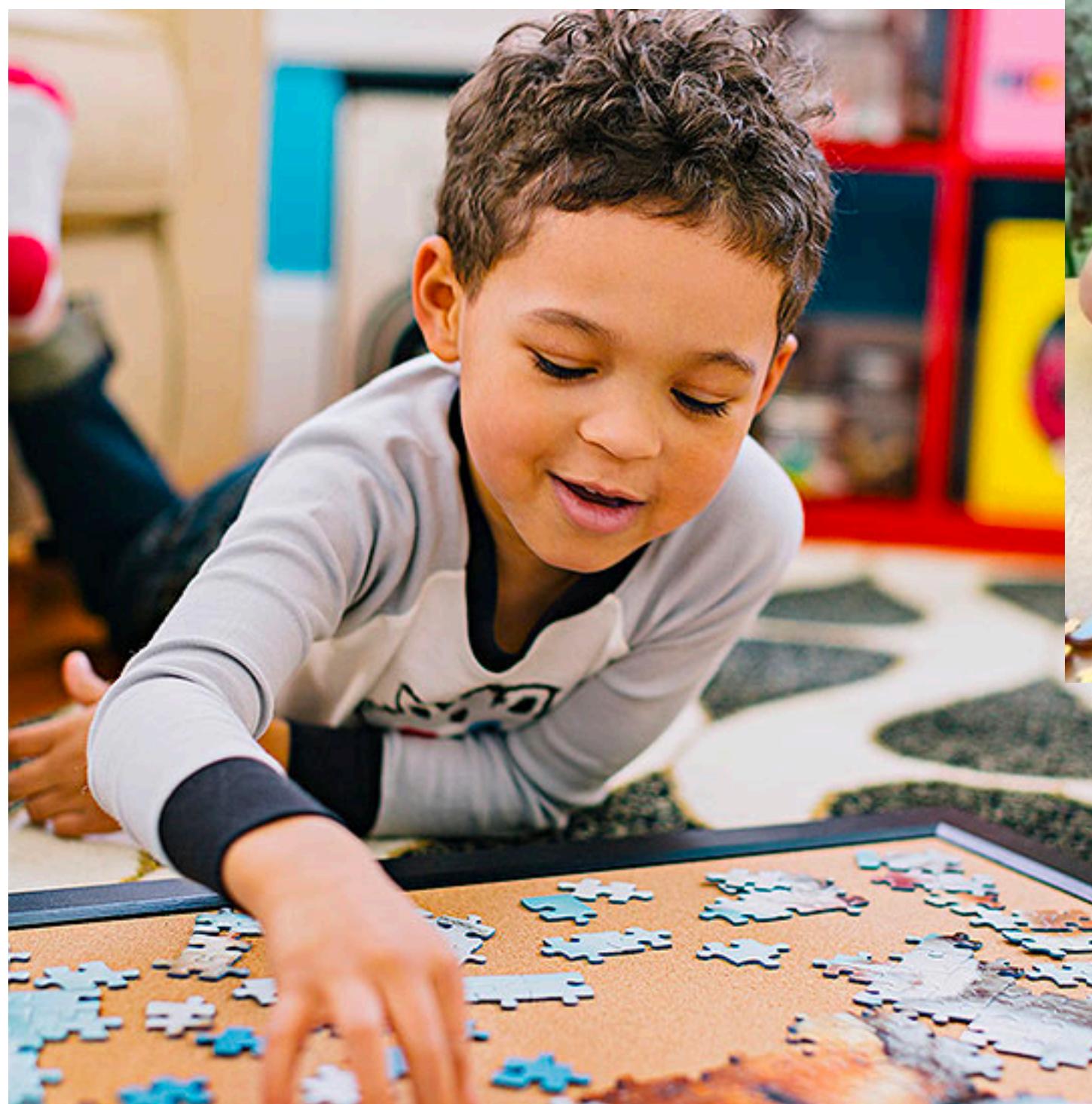


Image Credit: [Parents.com](#)

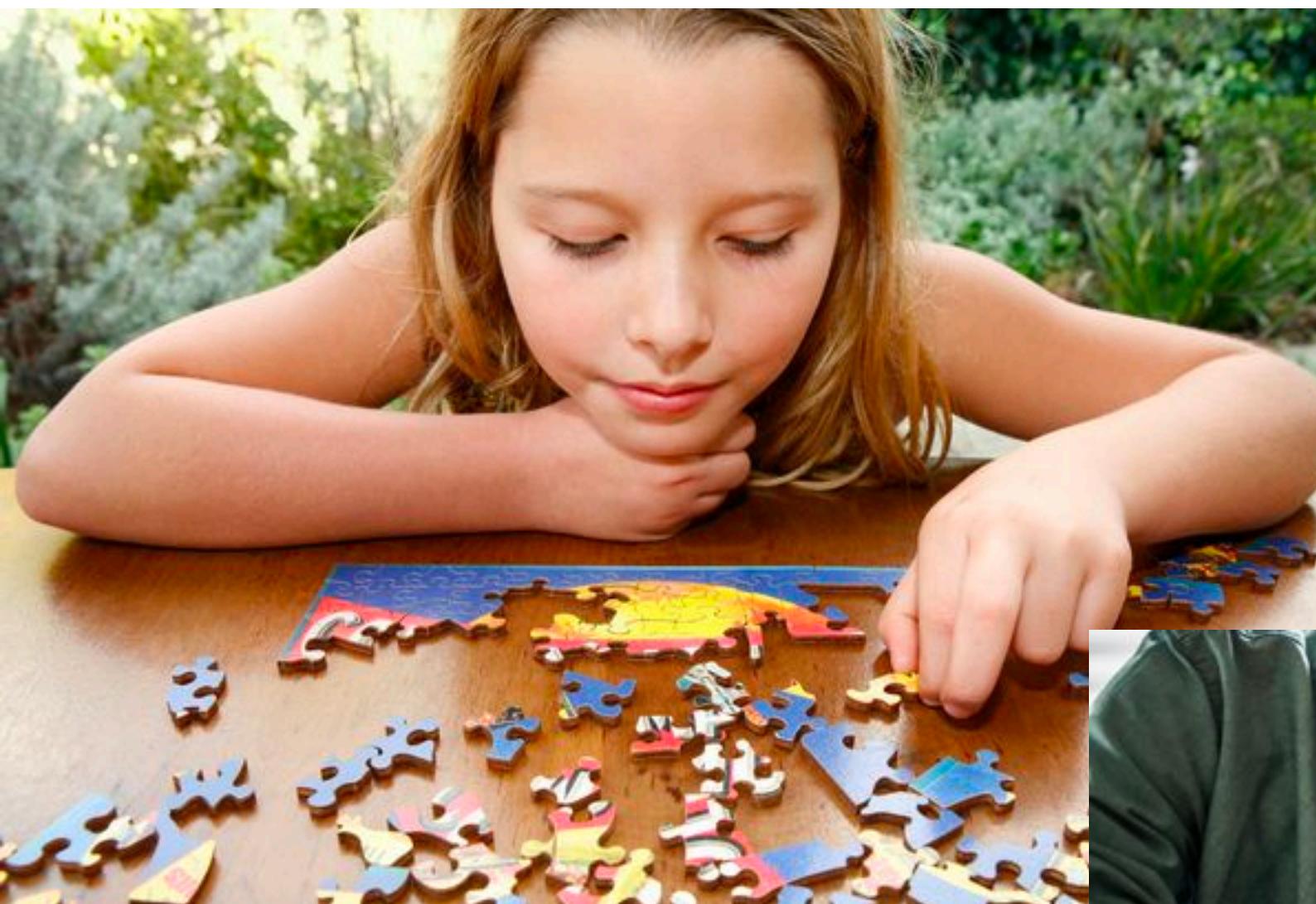


Image Credit: [Daily Mirror](#)



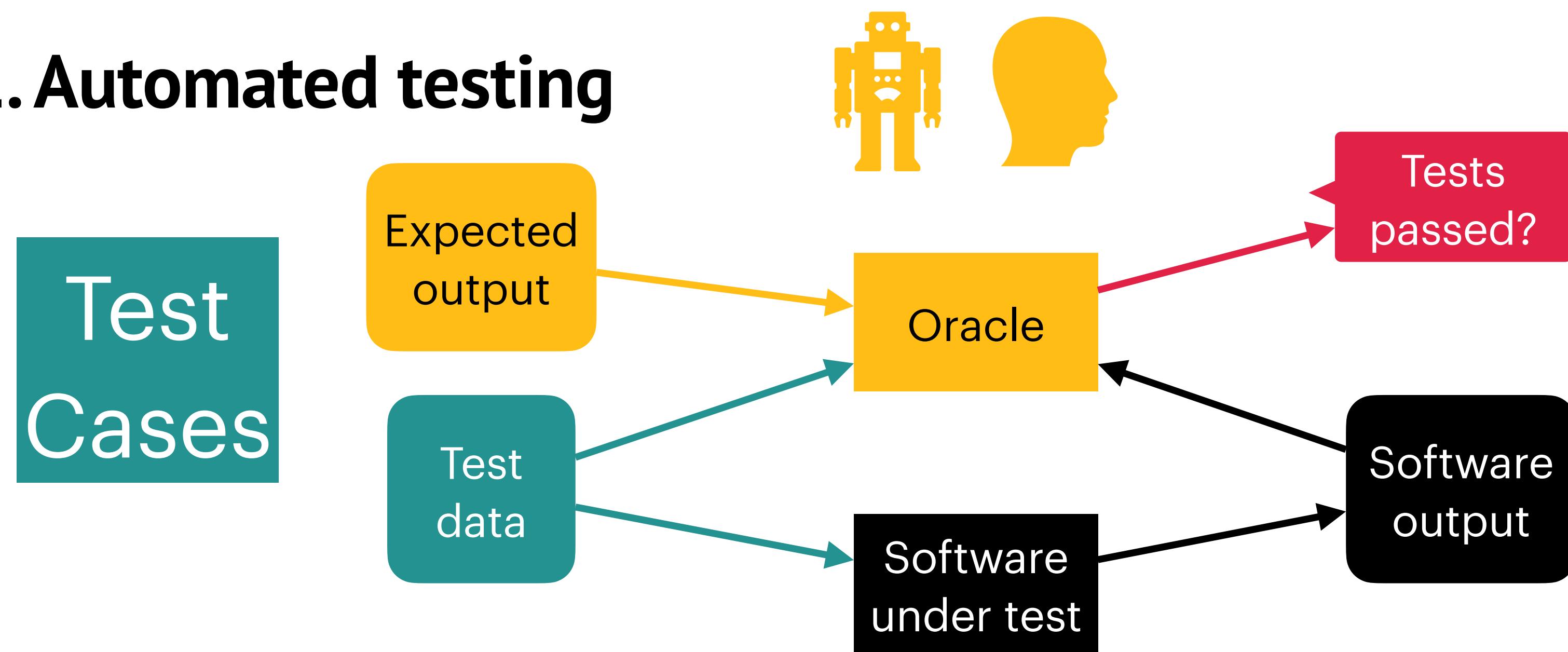
Image Credit: [womansday.com](#)

§3. The software testing process

Automated Testing

§3. The software testing process

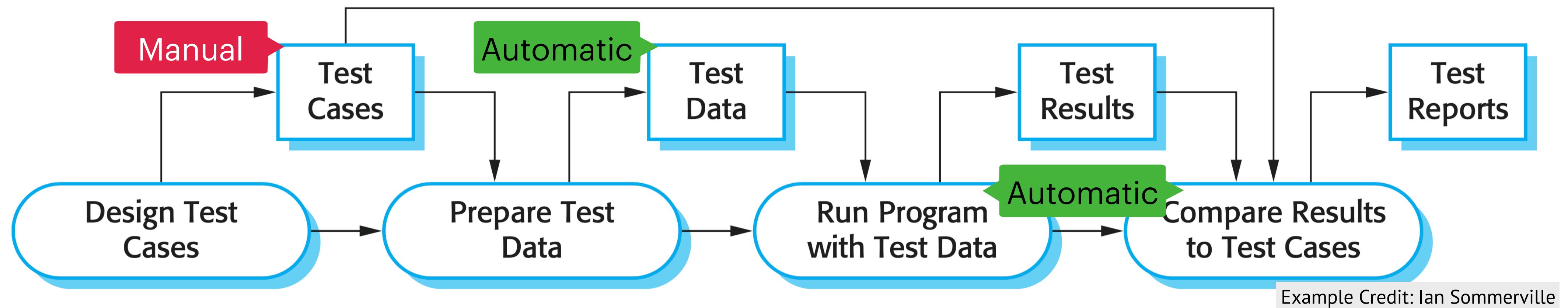
3.1. Automated testing



- **Software under test:** a module or unit of code where we can exercise some behaviour
- **Test data:** the information on which to act
- **Output data:** behaviour of the program given the input data
- **Oracle:** an entity that checks whether the software has provided the correct result

§3. The software testing process

3.1. Automated testing



- **Test cases:** specifications test inputs / system outputs
- **Test data:** inputs devised to test a system
- **Test runner:** compare expected results with predicted results

References

1. R. Stephens (2015). Beginning software engineering. John Wiley & Sons.
2. I. Sommerville (2020) *Software engineering 9th Edition*.

