

# **Tool Support** for Testing

Software Testing: IN3240 / 4240

# Summary:

## Types of test tools

Tool support for testing | Test tool classification

Tools for ...

Test management / Static testing / Test specification / Execution and logging

Performance and monitoring / Specific testing needs

## Effective use of test tools

Benefits and risks | Special considerations for tools

## Introducing a test tool into an organisation



# Part I: Close-ended questions

# Question 1

Which **tools** help support **static testing**?

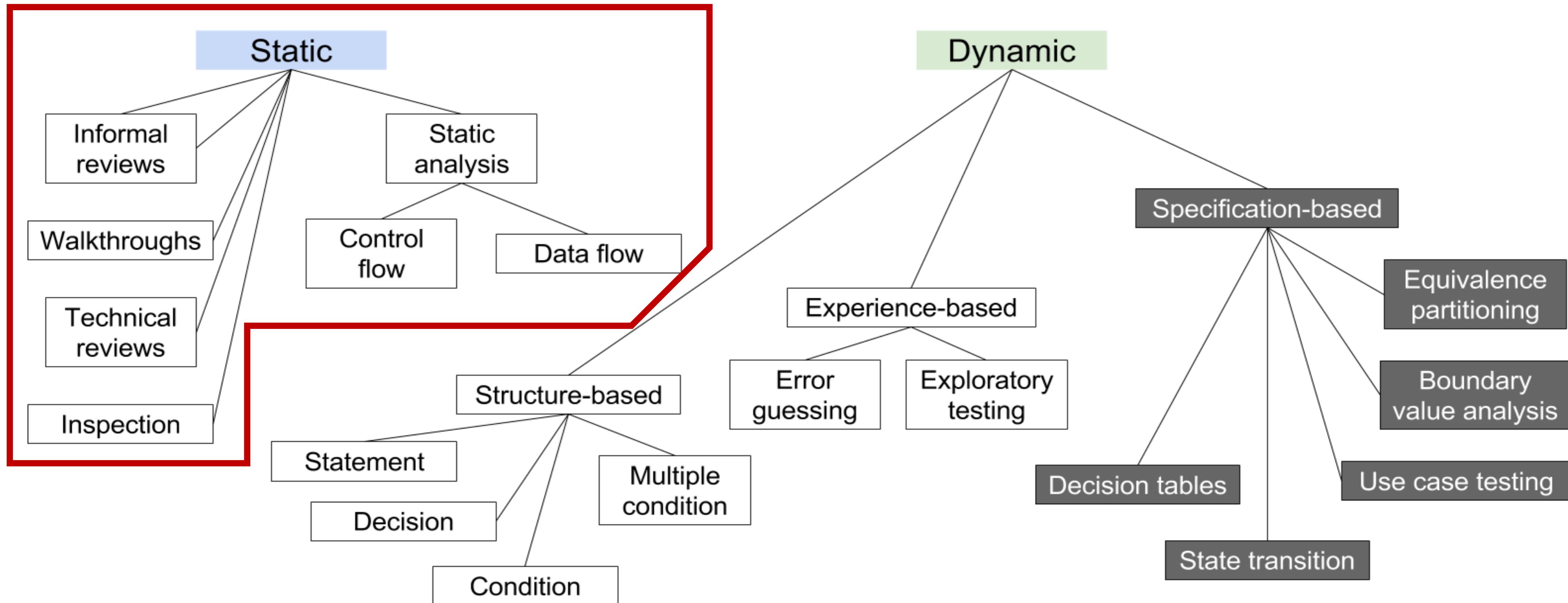
- a. Static analysis tools and test execution tools
- b. Review process support tools, static analysis tools and coverage measurement tools
- c. Dynamic analysis tools and modelling tools
- d. Review process support tools, static analysis tools and modelling tools



# Question 1: Clues

Which **tools** help support **static testing**?

Recall the different **types** of testing



# Question 1: Clues

Which **tools** help support **static testing**?

**Tools** for **static** testing

Tools that **aid** in **improving** the **code** / work product, **without executing** it

Categories

**Review** tools

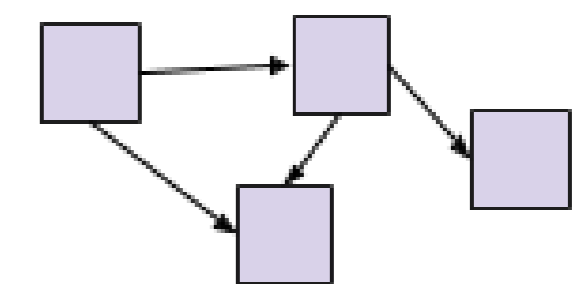
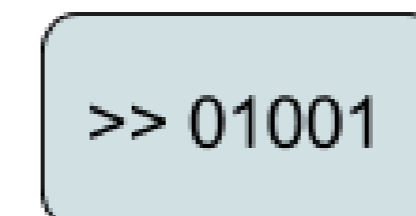
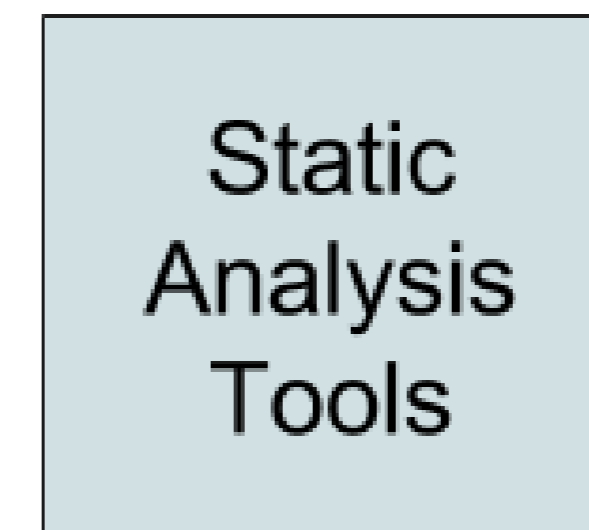
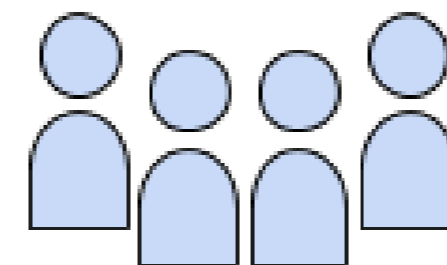
Supports the review process

**Static analysis** tools

Supports code examination

**Modelling** tools

Validate models of system / software



# Question 1: Clues

Which **tools** help support **static testing**?

**Review** process tools

Common **reference** for the **review** processes conducted

Keep **track** of all the **information** from the review process

Store and **communicate** review **comments**, report on **defects** and **effort**

**Monitoring** review status → Passed, passed with corrections, requires re-review

**When** to use?

Suitable for more **formal** review processes

**Geographically dispersed** teams



# Question 1: Clues

Which **tools** help support **static testing**?

**Review** process tool example: **Review Board**

**Collaborative** code **review** tool

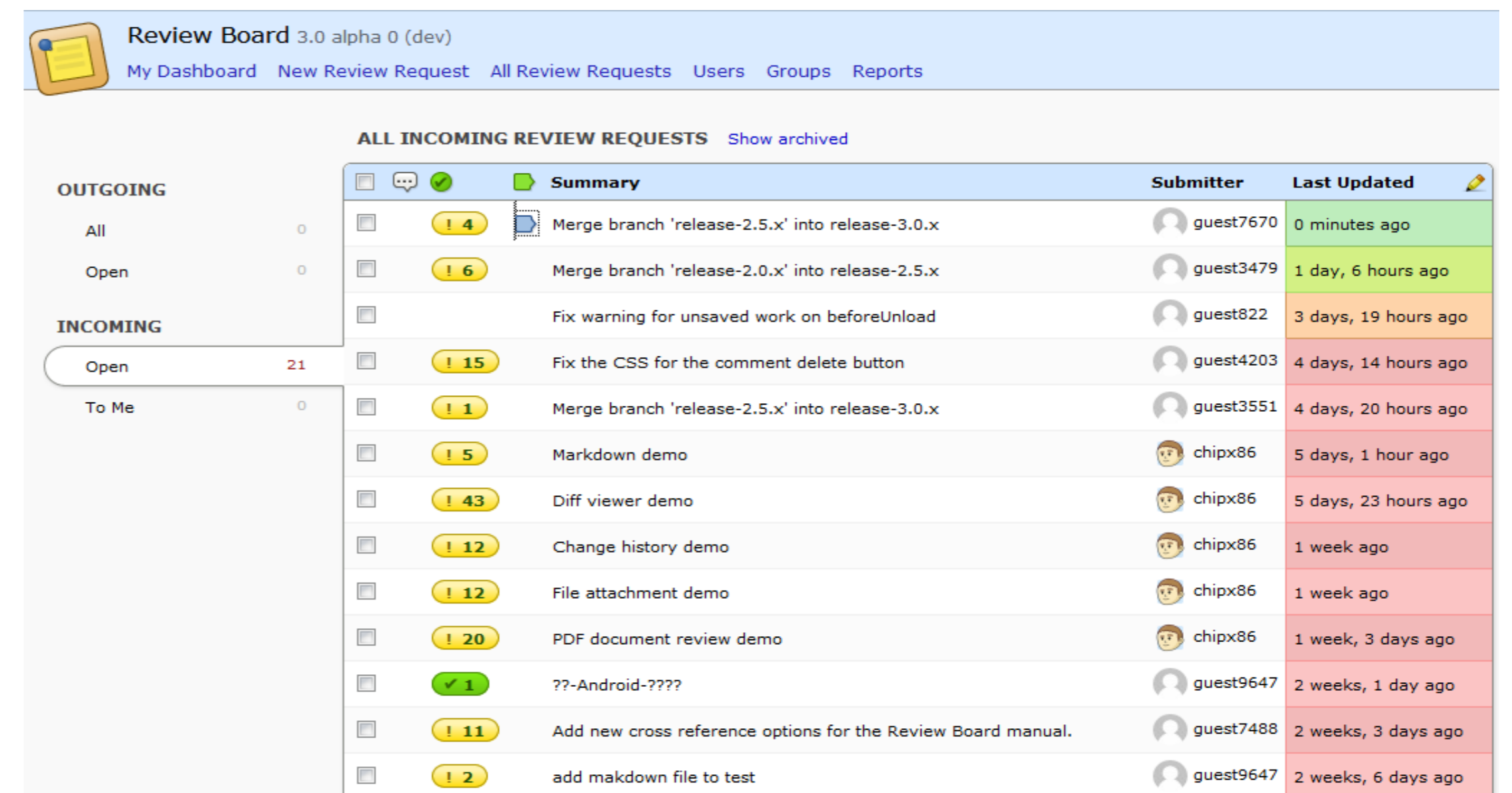
Tracks **changes** to code and documents

Changes must be **approved**

Feature to **discuss** proposed changes

Shows **difference** in code

**Current** and **proposed** side by side



The screenshot shows the Review Board interface. The top navigation bar includes 'Review Board 3.0 alpha 0 (dev)', 'My Dashboard', 'New Review Request', 'All Review Requests', 'Users', 'Groups', and 'Reports'. The main content area is titled 'ALL INCOMING REVIEW REQUESTS' with a 'Show archived' link. On the left, there are filters for 'OUTGOING' (All: 0, Open: 0) and 'INCOMING' (Open: 21, To Me: 0). The main table lists review requests with columns for 'Summary', 'Submitter', and 'Last Updated'. The first row is highlighted in green, indicating it is the selected item.

Summary	Submitter	Last Updated
Merge branch 'release-2.5.x' into release-3.0.x	guest7670	0 minutes ago
Merge branch 'release-2.0.x' into release-2.5.x	guest3479	1 day, 6 hours ago
Fix warning for unsaved work on beforeUnload	guest822	3 days, 19 hours ago
Fix the CSS for the comment delete button	guest4203	4 days, 14 hours ago
Merge branch 'release-2.5.x' into release-3.0.x	guest3551	4 days, 20 hours ago
Markdown demo	chipx86	5 days, 1 hour ago
Diff viewer demo	chipx86	5 days, 23 hours ago
Change history demo	chipx86	1 week ago
File attachment demo	chipx86	1 week ago
PDF document review demo	chipx86	1 week, 3 days ago
??-Android-????	guest9647	2 weeks, 1 day ago
Add new cross reference options for the Review Board manual.	guest7488	2 weeks, 3 days ago
add makdown file to test	guest9647	2 weeks, 6 days ago



# Question 1: Clues

Which **tools** help support **static testing**?

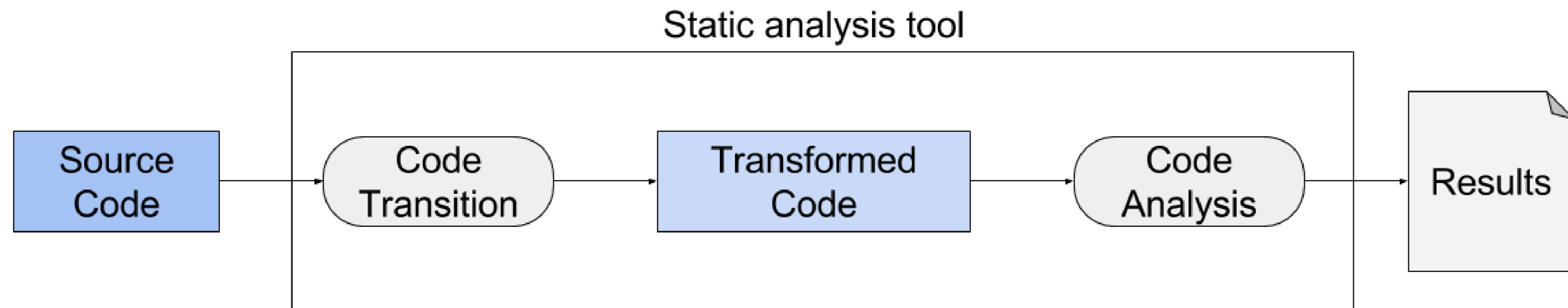
**Static analysis** tools (D)

Mostly used by **developers** → **Component** (unit) testing

**Tool** is **executed** → Code is **not**

The **source code** serves as **input** data to the **tool**

Extension of **compiler** technology



# Question 1: Clues

Which **tools** help support **static testing**?

**Static analysis** tools (D)

Support developers and testers in **finding defects** before dynamic testing

**Purpose**

To better **understand** the code, and find ways of **improving** it

**Common features**

Calculate **metrics** → **Complexity, nesting** levels → Identify areas of **risk**

Enforce coding **standards**

Analyse code **structures** and **dependencies**



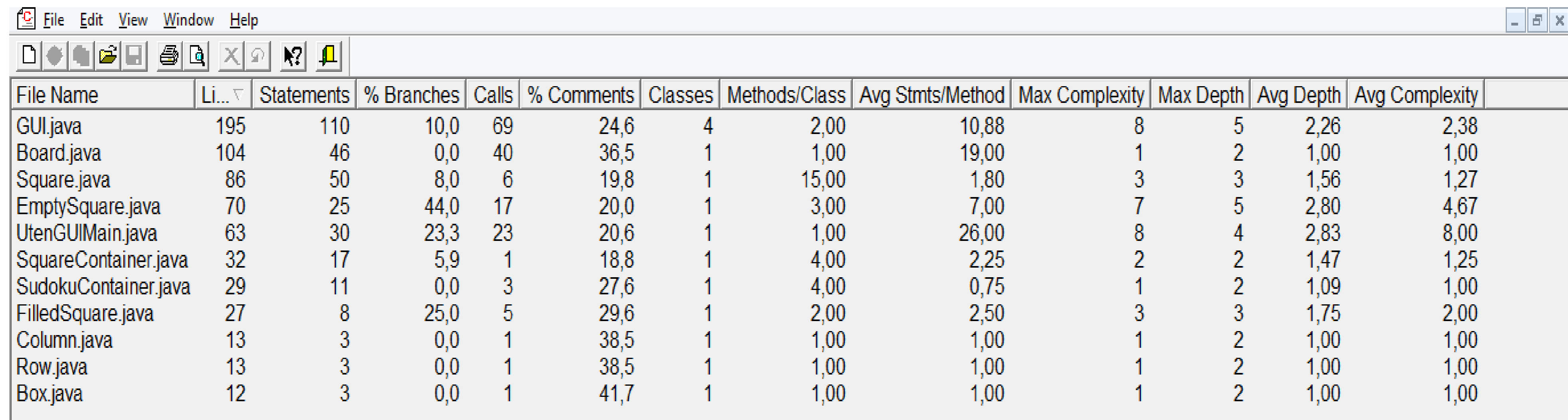
# Question 1: Clues

Which **tools** help support **static testing**?

Static analysis tool example: **Source Monitor**

Collects **metrics** from **source** code files

**Displays** and prints metrics in **tables** and **charts**



The screenshot shows the Source Monitor application window with a menu bar (File, Edit, View, Window, Help) and a toolbar. Below the toolbar is a table displaying metrics for several Java files. The table has 13 columns: File Name, Li... (likely Lines), Statements, % Branches, Calls, % Comments, Classes, Methods/Class, Avg Stmts/Method, Max Complexity, Max Depth, Avg Depth, and Avg Complexity. The data is as follows:

File Name	Li...	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmts/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
GUI.java	195	110	10,0	69	24,6	4	2,00	10,88	8	5	2,26	2,38
Board.java	104	46	0,0	40	36,5	1	1,00	19,00	1	2	1,00	1,00
Square.java	86	50	8,0	6	19,8	1	15,00	1,80	3	3	1,56	1,27
EmptySquare.java	70	25	44,0	17	20,0	1	3,00	7,00	7	5	2,80	4,67
UtenGUIMain.java	63	30	23,3	23	20,6	1	1,00	26,00	8	4	2,83	8,00
SquareContainer.java	32	17	5,9	1	18,8	1	4,00	2,25	2	2	1,47	1,25
SudokuContainer.java	29	11	0,0	3	27,6	1	4,00	0,75	1	2	1,09	1,00
FilledSquare.java	27	8	25,0	5	29,6	1	2,00	2,50	3	3	1,75	2,00
Column.java	13	3	0,0	1	38,5	1	1,00	1,00	1	2	1,00	1,00
Row.java	13	3	0,0	1	38,5	1	1,00	1,00	1	2	1,00	1,00
Box.java	12	3	0,0	1	41,7	1	1,00	1,00	1	2	1,00	1,00

# Question 1: Clues

Which **tools** help support **static testing**?

**Modelling tools (D)**

Validate **models** of the system / software

**Purpose**

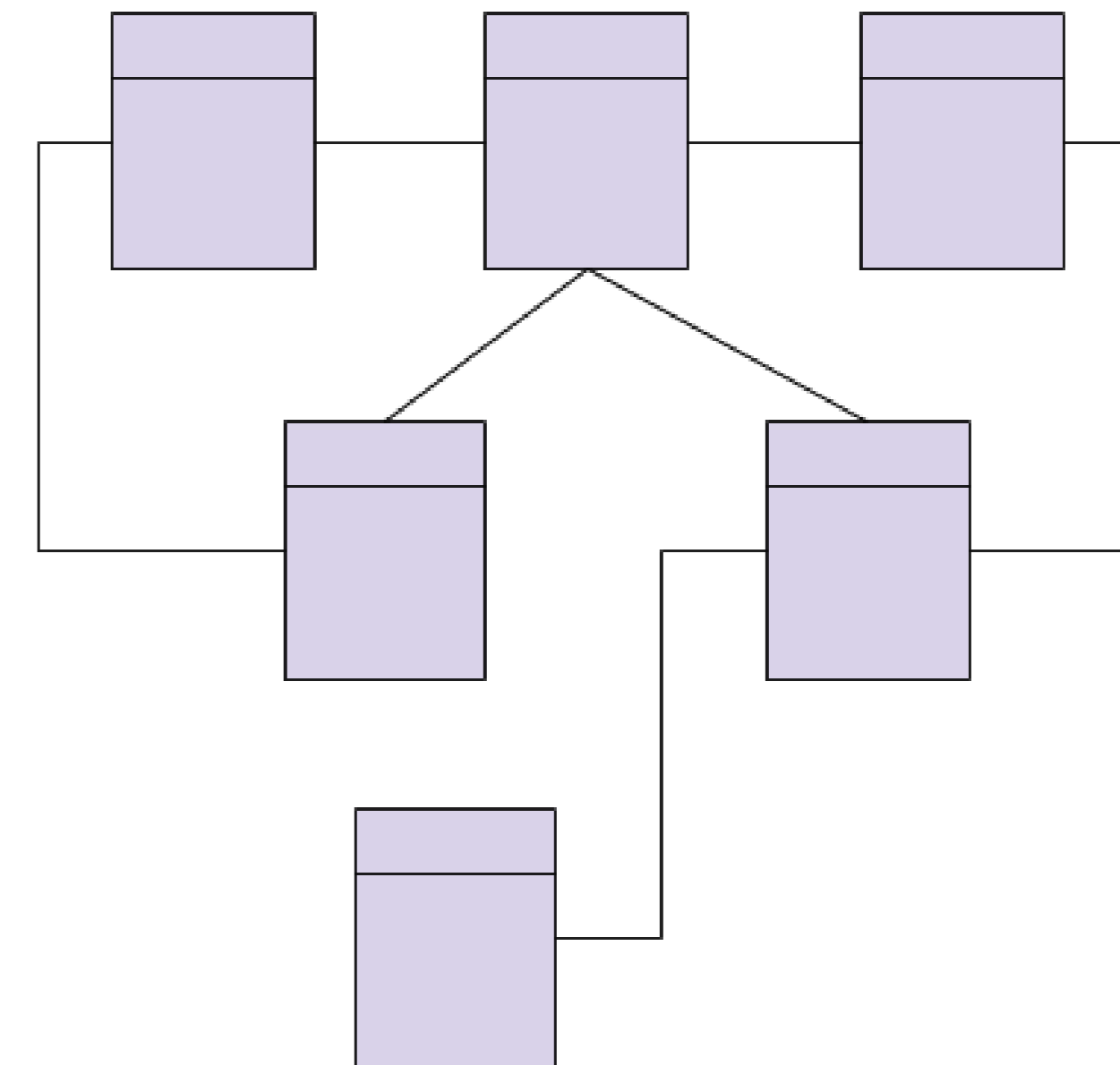
To better **aid** in **designing** the **software**

**Common features and characteristics**

Identify **inconsistencies** and **defects** within the **models**

Identify and prioritise **risk areas**

**Predicting** system **response** and **behaviour** under various situations



# Question 1: Clues

Which **tools** help support **static testing**?

Modelling tool example: **Star UML**

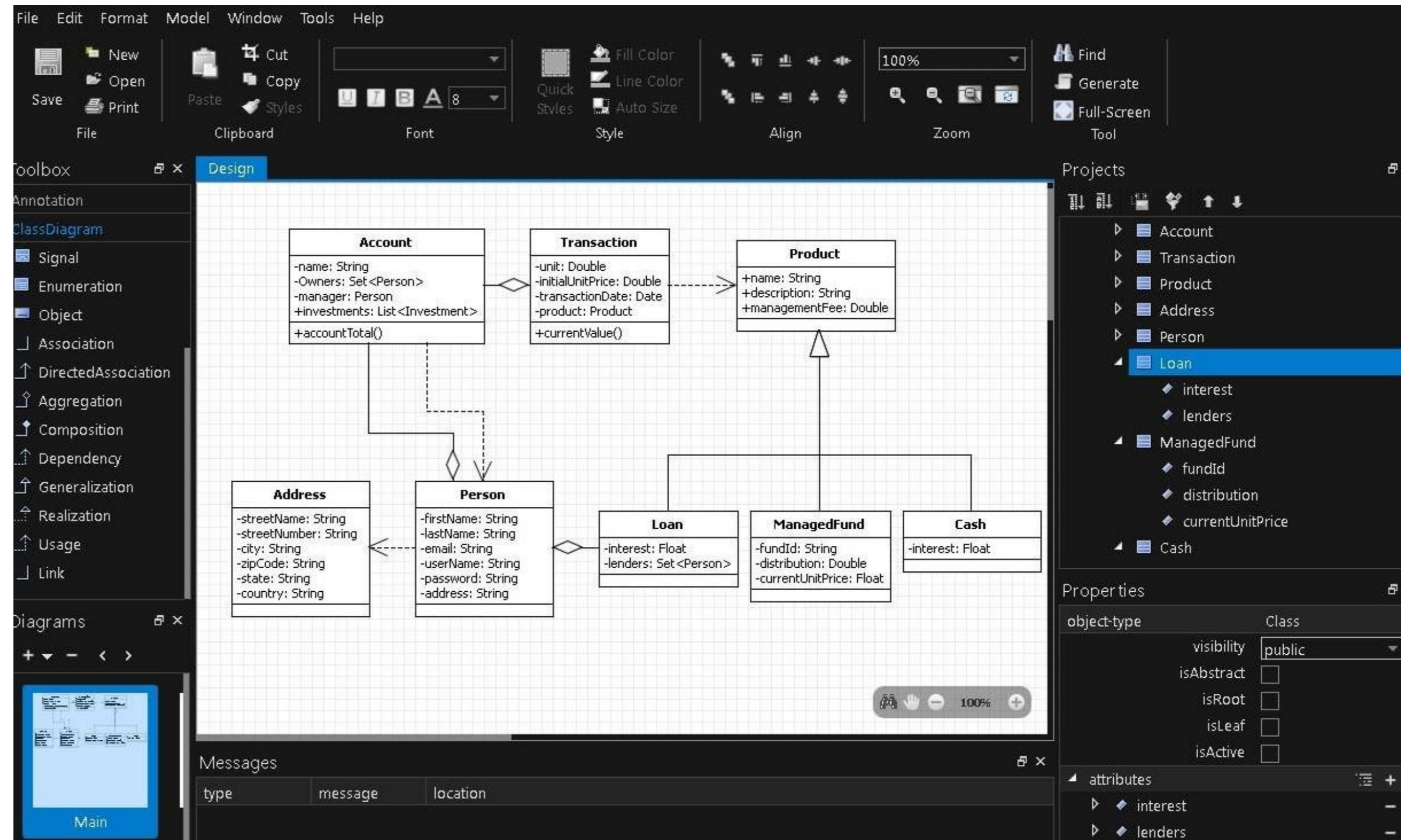
UML tool

Variety of **diagrams**

Class / Domain

Use case

Sequence



# Question 2

Which **test activities** are **supported** by **test harness** or **unit test framework** tools?

- a. Test management and control
- b. Test specification and control
- c. Test execution and control
- d. Performance and monitoring



# Question 2: Clues

Which **test activities** are **supported** by **test harness** or **unit test framework tools**?

Test **harness** and **unit test** framework tools (D)

The two **types** are **similar**

**Support** tools for testing **individual components** or software units

Harness: **Stubs** and **drivers** → Small programs that interact with software

Unit test framework tools → Support for object-oriented software

**When** are these tools used?

During test **execution** and **logging**



# Question 2: Clues

Which **test activities** are supported by test harness or unit test framework tools?

Example: Using **stubs** and **drivers**

Suppose we have a function *calculateAverageGrade()*

>> Calculates the average grade for a student in an academic year

Derives its value based on a function *getSubjectGrade()*

>> Retrieves the grade from a particular subject

We have only finished work on *calculateAverageGrade()*





# Question 2: Clues

Which **test activities** are supported by test harness or unit test framework tools?

Example: Using **stubs** and **drivers**

However, it cannot run without the function *getSubjectGrade()*

>> This function is **still** under **development**

**Solution:**

Create a “**dummy**” function to act in place of *getSubjectGrade()*

>> **Stub**



# Question 2: Clues

Which **test activities** are **supported** by **test harness** or **unit test framework** tools?

## Drivers

Calls the **component** to be tested

In other words: A **component** that **calls** the *Tested Unit*

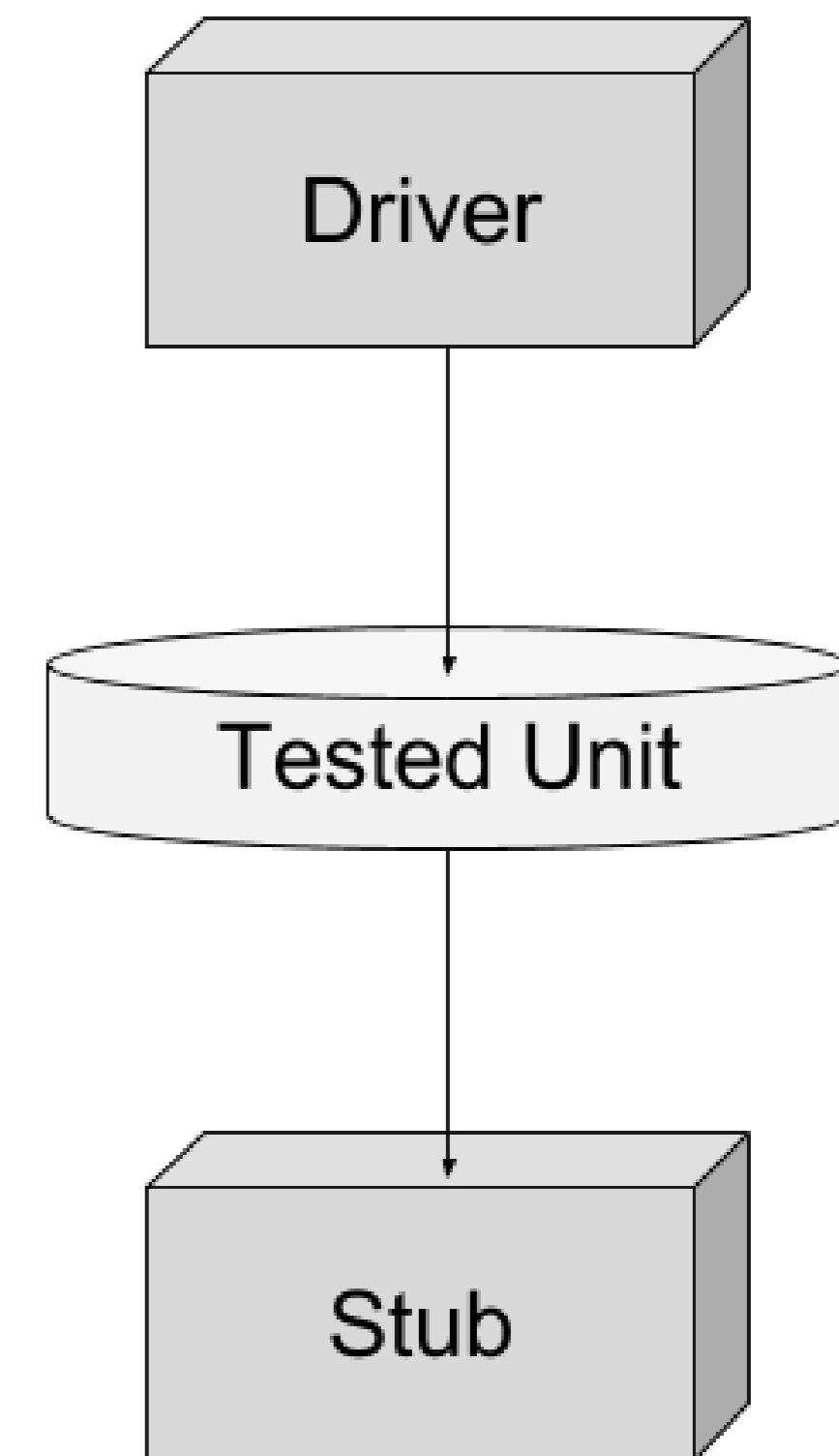
## Stubs

Called *from* the software component to be tested

In other words: A **component** the *Tested Unit* **depends** on

Partial implementation

Fake values



# Question 2: Clues

Which **test activities** are **supported** by **test harness** or **unit test framework tools**?

**Characteristics** of test **harness** and **unit test** framework tools

Supply **inputs** to the software being tested

Receive **outputs** generated by the software being tested

**Execute** a **set** of **tests** within the framework

Record **pass** / **fail** results of each test

**Store** tests

**Coverage** measurement at code level

Provide **support** for debugging



# Question 3

What are the potential **benefits** from **using tools** in general to **support testing**?

- a. Greater quality of code, reduction in the number of testers needed, better objectives for testing
- b. Greater repeatability of tests, reduction in repetitive manual work, objective assessment
- c. Greater responsiveness of users, reduction of tests run, objectives not necessary
- d. Greater quality of code, reduction in paperwork, fewer objectives to the tests

# Question 3: Clues

What are the potential **benefits** from **using tools** in general to **support testing**?

There are some things **humans** can **do better** than a **computer**

>> You see a friend in an unexpected place → You immediately recognise them

Humans are very good at this type of pattern recognition

Complex to write software for facial recognition

There are things **computers** can **do better** / more **quickly** than **humans**

>> Adding up five twenty-digit numbers quickly

A computer can perform this accurately and in “no time”

Unlike humans, computers are not inclined to get tired / find a task to be boring



# Question 3: Clues

What are the potential **benefits** from **using tools** in general to **support testing**?

Let **computers do** what they do **best**

**Reduction** of **repetitive**, manual **work**

Repeat the exact same procedure as previously

No human errors → People are prone to make errors

Greater **consistency** and **reliability**

Can prove more efficient and reliable

Objective **assessment**

**Ease** of **access** to **information** about testing



# Question 4

What is a potential **risk** in **using tools** to **support testing**?

- a. Unrealistic expectations, expecting the tool to do too much
- b. Insufficient reliance on the tool, i.e. still doing manual testing when a test execution tool has been purchased
- c. The tool may find defects that are not there
- d. The tool will repeat exactly the same thing it did the previous time



# Question 4: Clues

What is a potential **risk** in **using tools** to **support testing**?

## Risks

Underestimating **time, cost, effort** → **Introducing** a tool

Underestimating **time, effort** → **Achieve** significant and continuing **benefits** from tool

Unrealistic **expectations** and **over-reliance** on the tool

Expecting tools to be able to do “anything”

Simply purchasing a tool does not guarantee benefit

Must be **chosen carefully**

What are the objectives of the test effort?





# Question 5

Which of the following are **advanced scripting techniques for test execution tools**?

- a. Data-driven and keyword-driven
- b. Data-driven and capture-driven
- c. Capture-driven and keyhole-driven
- d. Playback-driven and keyword-driven



# Question 5: Clues

Which of the following are **advanced scripting techniques** for **test execution tools**?

## Background

As software **systems grow**, **manual** software testing becomes increasingly **difficult**

Aim to **decrease testing time** → **Automation** of tests

## Scripting techniques

Concerns: **What tests** should we **execute**? **How** de we **run** these tests?

Test execution **tool** → Needs to **know what to do** → **SCRIPT**

Tools are **software** → Script must be a **program**



# Question 5: Clues

Which of the following are **advanced scripting techniques** for **test execution** tools?

**Levels** of scripting

Linear  
Scripts

Structured  
Scripts

Shared  
Scripts

Data-Driven  
Scripts

Keyword-driven  
Scripts



# Question 5: Clues

Which of the following are **advanced scripting techniques for test execution tools**?

## Data-driven scripting

Separate test **scripts** and instructions from **data** (input / expected results)

Single test **script** to run with **different** data

### Advantages

Similar tests can be added very quickly

Time-saving → Reduction of repetitive manual work

### Disadvantages

Changes to either data file or script requires alteration of both



# Question 5: Clues

Which of the following are **advanced scripting techniques** for **test execution tools**?

**Example:** Simple Login Form

Test with **different** combinations of **username** and **password**

## Simple Login Form

Username :  Password :

**Problem:** Necessary to write three scripts for three different combinations?

1. Go to login page
2. Type username "Hansen"
3. Type password "oslo123"
4. Click "Login" button

1. Go to login page
2. Type username "Olsen"
3. Type password "bergen456"
4. Click "Login" button

1. Go to login page
2. Type username "Viljan"
3. Type password "dong789"
4. Click "Login" button

# Question 5: Clues

Which of the following are **advanced scripting techniques** for **test execution tools**?

**Example:** Simple Login Form

This **test** approach is **time-consuming**

**Solution:** Separate test script from data (username, password) → No hard-coding

*One* script retrieves different combinations of username and password

1. Go to page
2. Type username "**file.nextUsername()**"
3. Type password "**file.nextPassword()**"
4. Click "Login" button

Username	Password
Hansen	oslo123
Olsen	bergen456
Jensen	harstad789

# Question 5: Clues

Which of the following are **advanced scripting techniques** for **test execution tools**?

## Keyword-driven scripting

**Keywords** symbolising **actions** (functionality)

“One level up” from data-driven scripting

Can write tests using keywords

*“What to test, rather than how to test it”*

Keyword	Script
Login	script1
CH_password	script2
Logout	script3

[script1]  
1. Go to page  
2. Type username “file.nextUsername()”  
3. Type username “file.nextPassword()”  
4. Click “Login” button

[script2]  
1. Click on user avatar  
2. Click “Change password”  
3. Type current password  
4. Type new password  
5. Click “Confirm” button

[script3]  
1. Click on user avatar  
2. Click “Logout” button



# Question 6

Which of the following would **NOT** be **done** as part of **selecting** a **tool** for an **organisation**?

- a. Assess the organisational maturity, strengths and weaknesses
- b. Roll out the tool to as many users as possible within the organisation
- c. Evaluate the tool features against clear requirements and objective criteria
- d. Identify internal requirements for coaching and mentoring in the use of the tool





# Question 6: Clues

Which of the following would **NOT** be **done** as part of **selecting** a **tool** for an **organisation**?

Tools **aid** in the **testing** process

Tools **do not ensure** the **test effort** alone

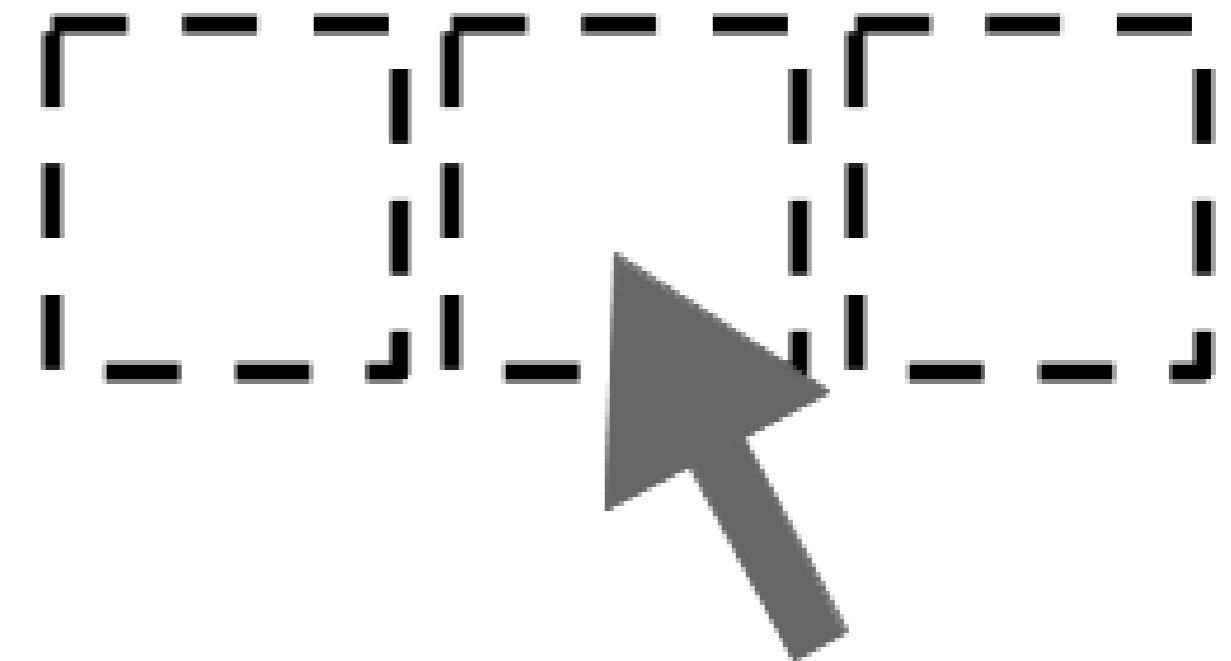
For a **tool** to provide **benefit**, it must:

**Match** the **need** within the **organisation**

Be **carefully selected** to meet the objectives

Recall test **principle 6**: Testing is **context-dependent**

The chosen **tool** must **match** the given **context**



# Question 6: Clues

Which of the following would **NOT** be **done** as part of **selecting** a **tool** for an **organisation**?

Considerations for **selecting** a test **tool**

Assessment of the **organisational maturity**

Identify **areas** within the **organisation** where **tool** support will **improve** the **test effort**

**Evaluate** the **tools** against clear **requirements** / objectives

**Proof-of-concept**

Ensure the product works as desired / Actually meets the requirements

Evaluate the **vendor** (or open-source network) → Training, support, etc.

Identify **planning** and **internal implementation** → Coaching, mentoring, etc.



# Question 6: Clues

Which of the following would **NOT** be **done** as part of **selecting** a **tool** for an **organisation**?

Success factors for selecting a test tool

Incremental roll-out → As opposed to “big bang”

Provide adequate support and training for users of the tool

Define guidelines for using the tool

Monitor use of the tool and the benefits achieved / Ask for feedback

Beware of:

Selecting tool based on “hype” / unrealistic expectations



# Question 7

Which of the following is a **goal** for a **pilot phase of introducing** a new **tool** to an **organisation**?

- a. Decide which tool to acquire
- b. Decide the main objectives and requirements for this type of tool
- c. Evaluate the vendor including training, support, and commercial aspects
- d. Decide on standard ways of using, managing, storing, and maintaining the tool and the test assets



# Question 7: Clues

Which of the following is a **goal** for a **pilot phase** of **introducing** a new **tool** to an **organisation**?

**Proof-of-concept**

Demonstration to **verify** that some **concept** (tool) has **potential** within the **context**

**Pilot project** → Tool has already been chosen

Use the tool on a **small scale**

**Learn** more about the tool and **explore ways** of **using** it

Explore various **settings, functionality**

Decide on **standard ways** of **using** the tool

Naming conventions, creation of libraries, maintenance of test assets, etc.



# Question 8

**Pair** the following **testing tools** with the main **activity** they support

Tools for test execution and logging	Used for traceability of tests, test results and incidents. Used to connect tests with their originating documents, such as requirements specifications.
Tools for static testing	Enable tests to be executed automatically using stored inputs and expected outcomes.
Tools for performance and monitoring	Used for testing the structure and dependencies of the code. Used to measure code coverage with tests.
Tools for test management	They simulate a load on: - An application / A database / A system environment

# Question 9

Test **comparators** are used when the executed **test generates a lot of output**. In order to **validate** the output against an **oracle**, one **needs to use a test tool**.

E.g. Send SMS with less than 10 special characters to 20.000 users

- a. True
- b. False



# Question 9: Clues

Test **comparators** are used when the executed **test generates a lot of output**.

Testing is **more** than providing **inputs**

Need to **check** if software **produces** the **correct** result

**Compare actual** outcomes to **expected** results

Two ways of comparing results

**Dynamic** comparison → Comparison done **during** test execution

**Post-execution** comparison → Comparison performed **after** test has finished

Software under test is no longer executing





# Question 9: Clues

Test **comparators** are used when the executed **test generates a lot of output.**

## Dynamic comparison

Best done by test **execution tools**

**Useful** when actual results do not match expected results in the **middle** of a **test**

Tool may be **programmed** to take **recovery actions** / go to a **different** set of **tests**

## Example

Good for comparing wording of an error message

Does the pop-up message match the correct wording for that error message?



# Question 9: Clues

Test **comparators** are used when the executed **test generates a lot of output**.

**Post-execution** comparison

Done by a **separate**, standalone **tool** (not test execution tool)

>> **Test comparator** / test comparison tool

**Best** for comparing **large amounts of data**

Example

Comparing the contents of an entire file

Does the produced file match the expected contents of that file?

Comparing a large set of records from a database to the expected contents

A decorative graphic at the bottom of the slide consisting of a series of overlapping, right-pointing chevrons. The chevrons are light gray, with the final one on the right being a darker shade of gray.

# Question 10

A potential \_\_\_\_\_ of **using** a test **tool** is the **reduced** repetitive **manual work**.

E.g. When running regression tests, re-entering the same input data, etc.



# Question 11

Which of the following are **benefits** and which are **risks** of **using tools to support testing**?

1. Over-reliance on the tool
2. Greater consistency and repeatability
3. Objective assessment
4. Unrealistic expectations
5. Underestimating the effort required to maintain the tool
6. Ease of access to information about tests or testing
7. Repetitive work is reduced

# Question 11

Which of the following are **benefits** and which are **risks** of **using tools to support testing**?

- a. Benefits: 3, 4, 6 and 7. Risks: 1, 2, and 5
- b. Benefits: 1, 2, 3 and 7. Risks: 4, 5, and 6
- c. Benefits: 2, 3, 6 and 7. Risks: 1, 4, and 5
- d. Benefits: 2, 3, 5 and 6. Risks: 1, 4, and 7

# Question 11: Clues

Which of the following are **benefits** and which are **risks** of **using tools to support testing**?

Greater **consistency** and **repeatability**

**People** tend to do the same **tasks** in a slightly **different** way

**Distractions** affect human **performance**

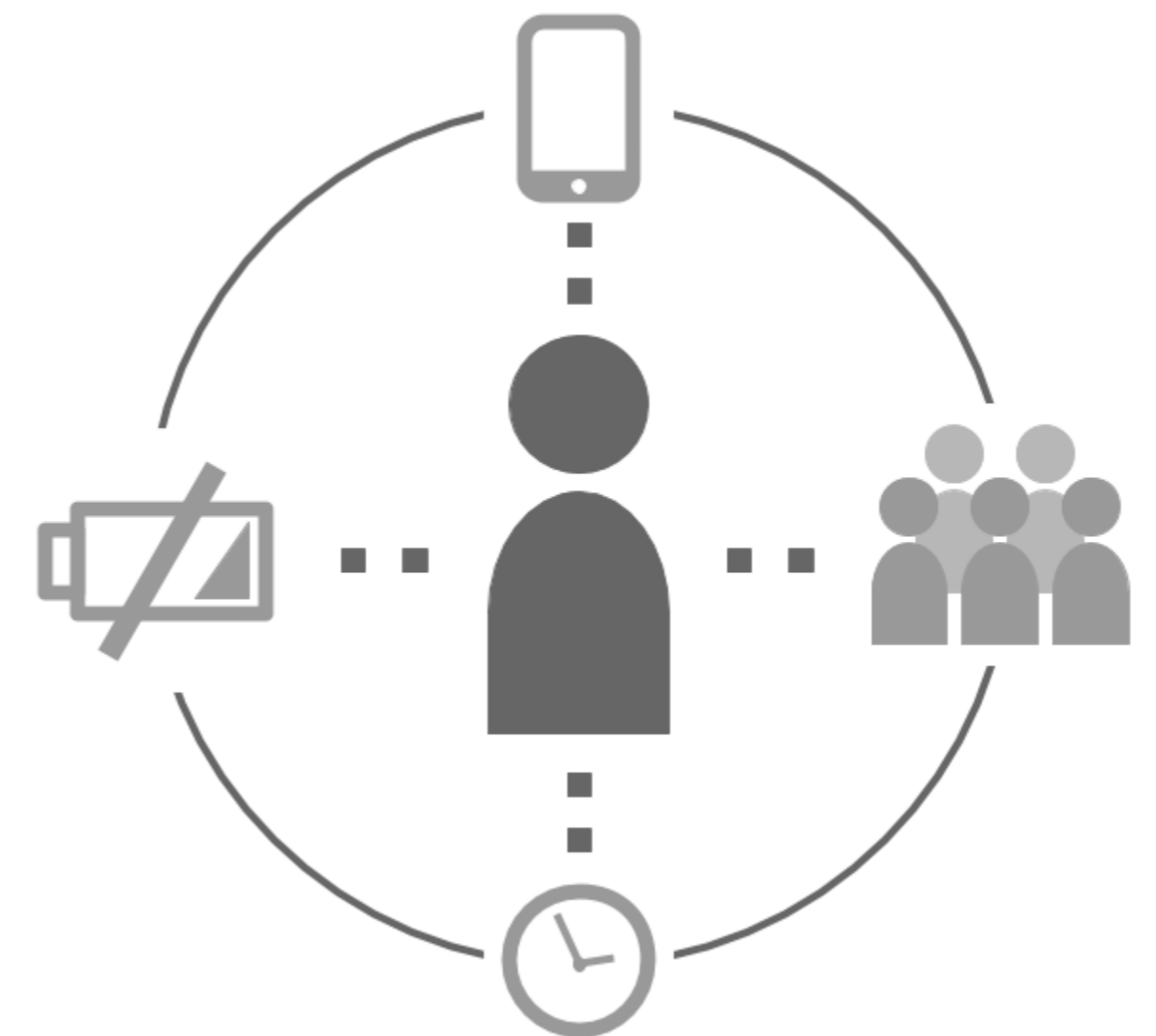
Doing more than one task simultaneously

Interruptions by peers / co-workers

Fatigue and personal issues

External pressures

**Tools** will **reproduce** the **exact** same **procedure** as **previously**



# Question 11: Clues

Which of the following are **benefits** and which are **risks** of **using tools to support testing**?

**Objective** assessment

**Humans** are prone to make **errors**

**Subjective** preconceived notions and **bias** toward **verification**

Testing tools on the other hand ...

**Objective** “preconceived notions”

**Assessment** → **Repeatable** and **consistently** calculated

Cyclomatic complexity, nesting levels

Coverage, system behaviour, incident statistics



# Question 11: Clues

Which of the following are **benefits** and which are **risks** of **using tools to support testing?**

**Ease of access to information** about the tests or test effort

Information presented **visually**

**Easier** for the **human** mind to **understand**

Chart, graphs > Long list of numbers

**Special purpose** tools provide **features directly**

Statistics and graphs

Incident rates

Performance





# Question 12

Which **test activities** are **supported** by **test data preparation tools**?

- a. Test management and control
- b. Test specification and control
- c. Test execution and control
- d. Performance and monitoring



# Question 12: Clues

Which **test activities** are **supported** by **test data preparation** tools?

**Tests** should **reflect realistic** (correct) **scenarios**

Systems are often **required** to handle **significant load** / interactions

Inadequate / **insufficient** testing **compromises** system **quality**

**Setting up test data** → Significant **effort**

Extensive range or volume of data needed

**Creating this data** can be very **resource-consuming**

Test **data preparation** tools **help** us manage this effort



# Question 12: Clues

Which **test activities** are **supported** by **test data preparation** tools?

Common **features** of **test data preparation** tools

Data can be **selected** from an **existing** database

Data can be **created**, **generated**, and **altered** for use in **tests**

Construct a **large number** of similar records → Volume tests

**When to use?**

During test **specification** and **control** → Test data **management** is **difficult**

**Ensure** the system under **test** is being tested *realistically*

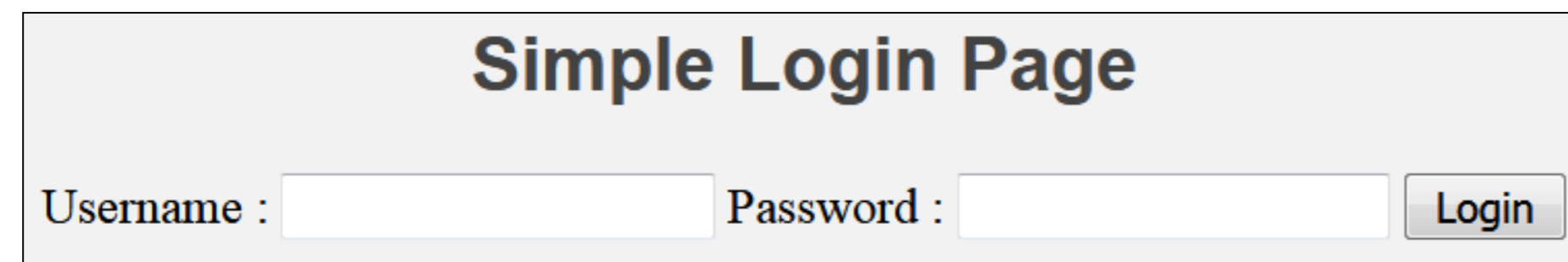
Useful for **performance** and **reliability** testing



# Question 12: Clues

Which **test activities** are **supported** by **test data preparation tools**?

**Example:** Simple login site



A screenshot of a simple login page. It features a title "Simple Login Page" at the top. Below the title, there are two input fields: "Username :" followed by a text box, and "Password :" followed by a text box. To the right of the password field is a "Login" button.

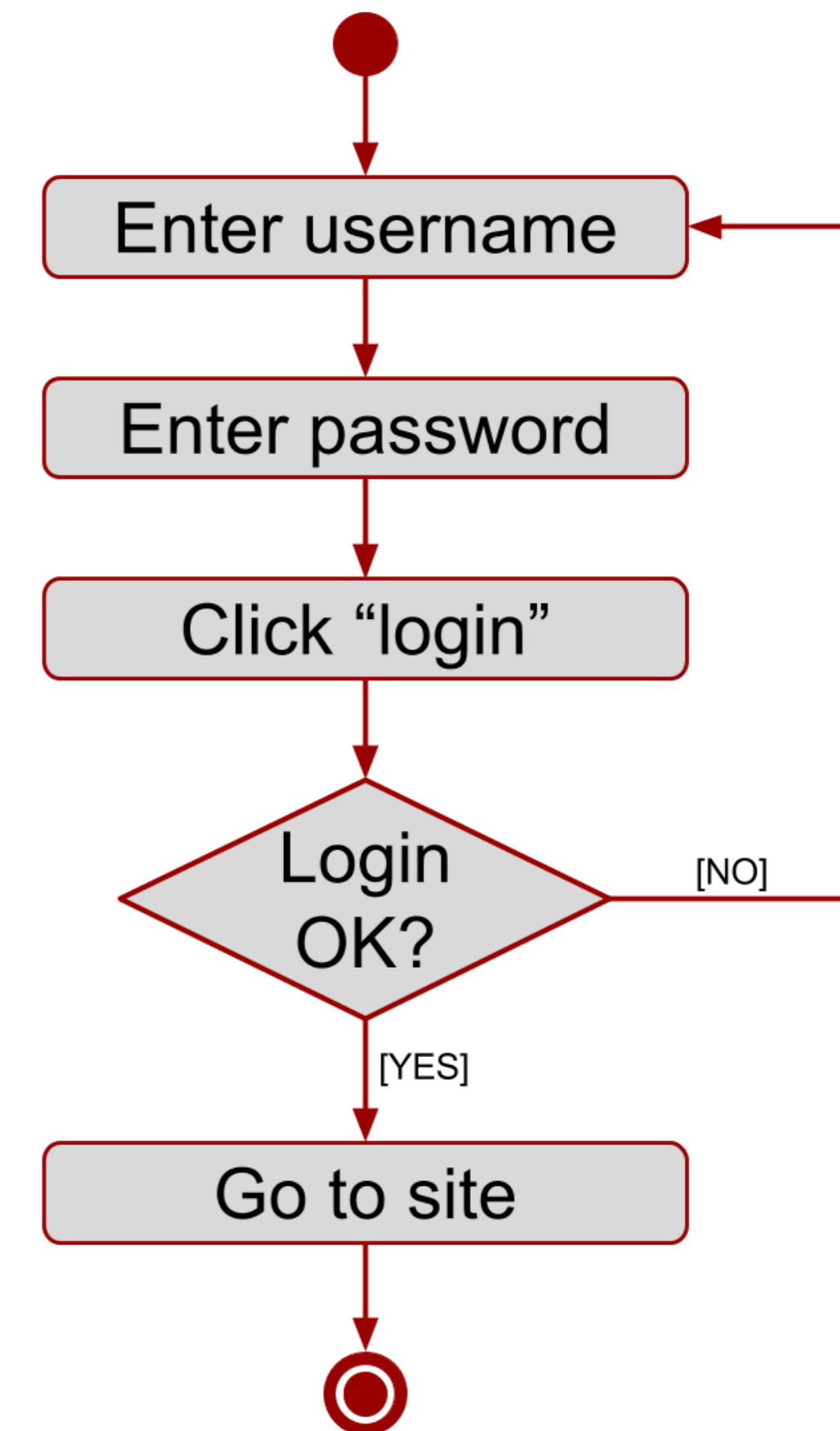
**Data need for testing website**

List of various **usernames**

List of various **passwords**

Database of existing **users**

**We do not want to create all this by hand!**



# Question 13

Consider the following **types of tools**:

1. Test management tools
2. Static analysis tools
3. Modelling tools
4. Dynamic analysis tools
5. Performance testing tools



# Question 13

Which of the following **tools** are most **likely** to be **used** by **developers**?

- a. Static analysis tools, modelling tools, and dynamic analysis tools
- b. Test management tools, dynamic analysis tools, and performance testing tools
- c. Test management tools, static analysis tools, and performance testing tools
- d. Modelling tools, dynamic analysis tools, and performance testing tools



# Question 13: Clues

Which of the following **tools** are most **likely** to be **used** by **developers**?

**Developers** are primarily concerned with *building* and creating

Want to **produce** something of **value** and **quality**

Concerned with code **quality**, **design**, **performance**, etc.

As such, **developers** are **likely** to use tools that **help** meet the **objectives**

**Static** analysis tools → Examines work products without execution

**Modelling** tools → Validate models of system / software

**Dynamic** analysis tools → Require code to be running



# Question 13: Clues

Which of the following **tools** are most **likely** to be **used** by **developers**?

## Static analysis tools

Examination of **code without executing** it

Can additionally perform static **analysis** on **requirements**

## Advantages and contribution to quality

**Ensure** and enforce **coding standards**

Greater **analysis** of **structures** and **dependencies**

Can be used **before dynamic** testing





# Question 13: Clues

Which of the following **tools** are most **likely** to be **used** by **developers**?

## Modelling tools

Validate **models** of system / software

Check **consistency** of **data objects** in a database

Reveal **inconsistencies** and **defects**

## Advantages and contribution to quality

**Ensure** system is built / **designed** in the **right** (most sensible) way

Can be used **before dynamic** testing



# Question 13: Clues

Which of the following **tools** are most **likely** to be **used** by **developers**?

## Dynamic analysis tools

Require **code** to be **executed** during tests

Analyse **what** is **happening** → Behind the scenes

Analogy: **Test driving** a car (engine must be running), not simply sitting in it (static)

## Advantages and contribution to quality

Helps detect memory leaks, time dependencies, and pointer arithmetic errors

Can be used **before dynamic** testing



# Question 14

Which **success factors** are **required** for good **tool support** within an **organisation**?

- a. Acquiring the best tool and ensuring that all testers use it
- b. Adapting processes to fit with the use of the tool and monitoring tool use and benefits
- c. Setting ambitious objectives for tool benefits and aggressive deadlines for achieving them
- d. Adopting practices from other successful organisations and ensuring that initial ways of using the tool are maintained

# Question 14: Clues

Which **success factors** are **required** for good **tool support** within an **organisation**?

Introducing and using a tool can be a **complex** task

Internal **resistance**, scepticism

**Lack of skill**

**Necessary** to have a good **framework** for **optimal** use of the tool

How can we benefit from using this tool?

After all, introducing a tool constitutes a **change**

How can we best **manage** this **process**?

Recall, **success** is **not guaranteed**



# Question 14: Clues

Which **success factors** are **required** for good **tool support** within an **organisation**?

**Learning** from using the tool

Cannot expect the **tool** do **solve all** testing-related **problems**

**Adapt processes** to fit with the **use** of the tool

Continuous(!) **improvement** of testware and tool **artefacts**

**Mapping** how the tool is being used

**Monitor usage**

Assess **benefits** and **challenges** experienced when using the tool

Provide adequate **training**, coaching, and mentoring



# Question 15

What kind of **interface** can be **used** to **automate** tests?

- a. API – Application programming interface
- b. GUI – Graphical user interface
- c. Both API and GUI
- d. None of the above



# Question 15: Clues

What kind of **interface** can be **used** to **automate** tests?

**API** – Application Programming Interface

Set of **routines**, **definitions**, **protocols** and **tools** → Building **software**

Methods for **communication** between **software** components

APIs in test automation

**Selenium Java API**

Classes to use Selenium with Java

**JSONPlaceholder**

Provides a variety of fake data for testing applications



# Question 15: Clues

What kind of **interface** can be **used** to **automate** tests?

**GUI** – Graphical User Interface

Interface for the **interaction** between **users** and electronic **devices**

**Information** presented through **graphical** icons / visual indicators

GUI in test automation

**Selenium IDE**

Record and play test cases

Can be exported in various formats (C#, Java, Python, Ruby)

For more information, see *W08 – Tool Support for Testing*





# Question 15: Clues

What kind of **interface** can be **used** to **automate** tests?

**Example:** Testing a simple **login** page

Address: <http://inf3121-login-example.bitballoon.com/>

### Simple Login Page

Username :  Password :

**How to use:** Please provide a username and password.

**Valid username and password combinations**

Username	Password
beckham	football
federer	tennis
deniro	acting
jobs	apple

# Question 15: Clues

What kind of **interface** can be **used** to **automate** tests?

Login Procedure (API)

1. **Open** site
2. Type **username**
3. Type **password**
4. Click **“login”** button
5. **Verify** successful login
6. Go **back** to start

```
@Test
public void testBeckham() throws Exception {
    driver.get(baseUrl + "/");

    driver.findElement(By.id("username")).clear();
    driver.findElement(By.id("username")).sendKeys("beckham");

    driver.findElement(By.id("password")).clear();
    driver.findElement(By.id("password")).sendKeys("football");

    driver.findElement(By.id("submit")).click();

    try {
        assertEquals("SUCCESS", driver.getTitle());
    } catch (Error e) {
        verificationErrors.append(e.toString());
    }

    driver.findElement(By.cssSelector("button")).click();

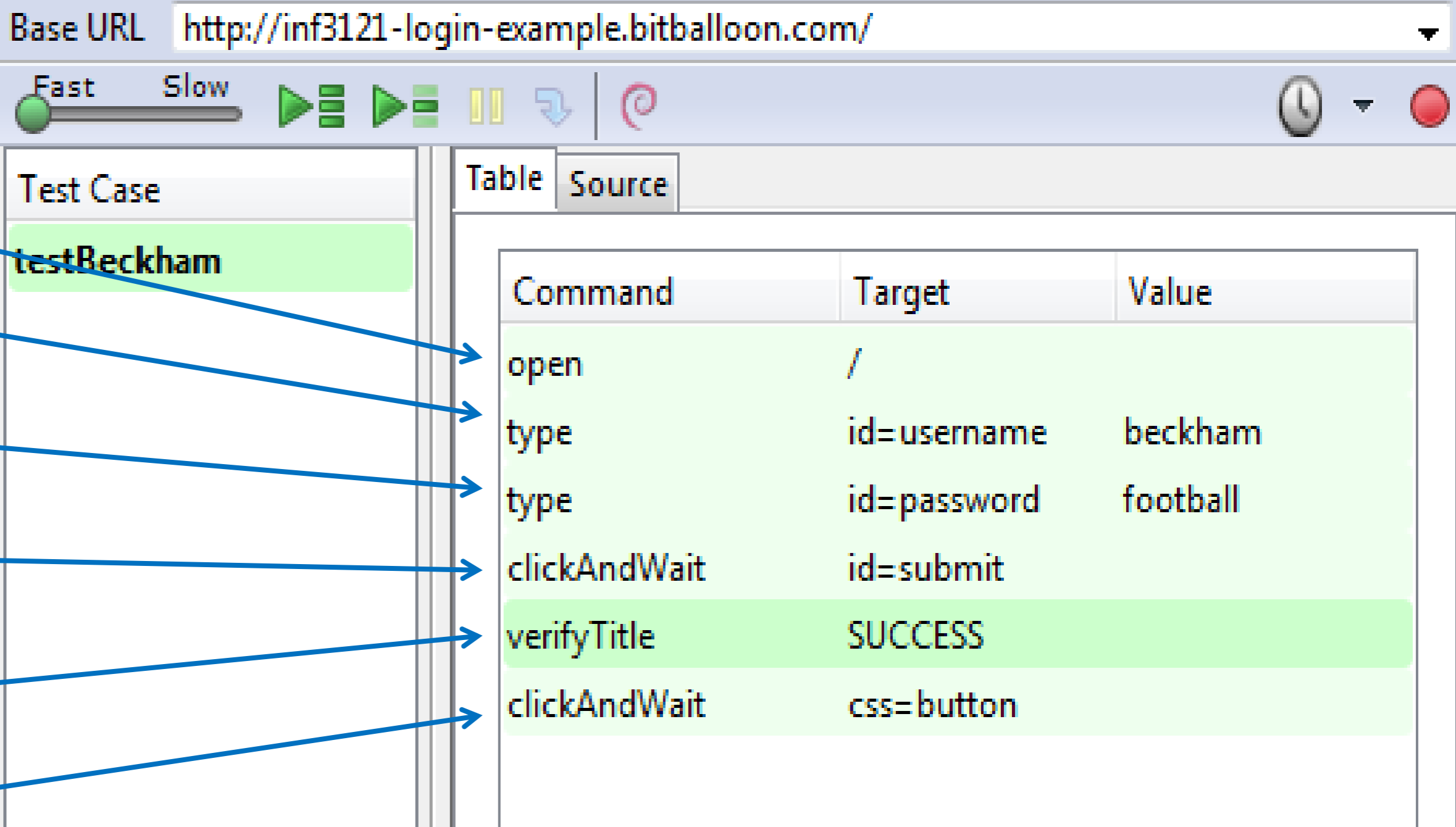
    //driver.quit();
}
```

# Question 15: Clues

What kind of **interface** can be **used** to **automate** tests?

Login Procedure (GUI)

1. **Open** site
2. Type **username**
3. Type **password**
4. Click "**login**" button
5. **Verify** successful login
6. Go **back** to start



The screenshot shows the Selenium IDE interface. The Base URL is `http://inf3121-login-example.bitballoon.com/`. The test case is named `testBeckham`. The table below lists the commands and their targets/values:

Command	Target	Value
open	/	
type	id=username	beckham
type	id=password	football
clickAndWait	id=submit	
verifyTitle	SUCCESS	
clickAndWait	css=button	

# Question 15: Clues

What kind of **interface** can be **used** to **automate** tests?

Can use **both API** and **GUI** to **automate** testing

```
@Test
public void testBeckham() throws Exception {
    driver.get(baseUrl + "/");

    driver.findElement(By.id("username")).clear();
    driver.findElement(By.id("username")).sendKeys("beckham");

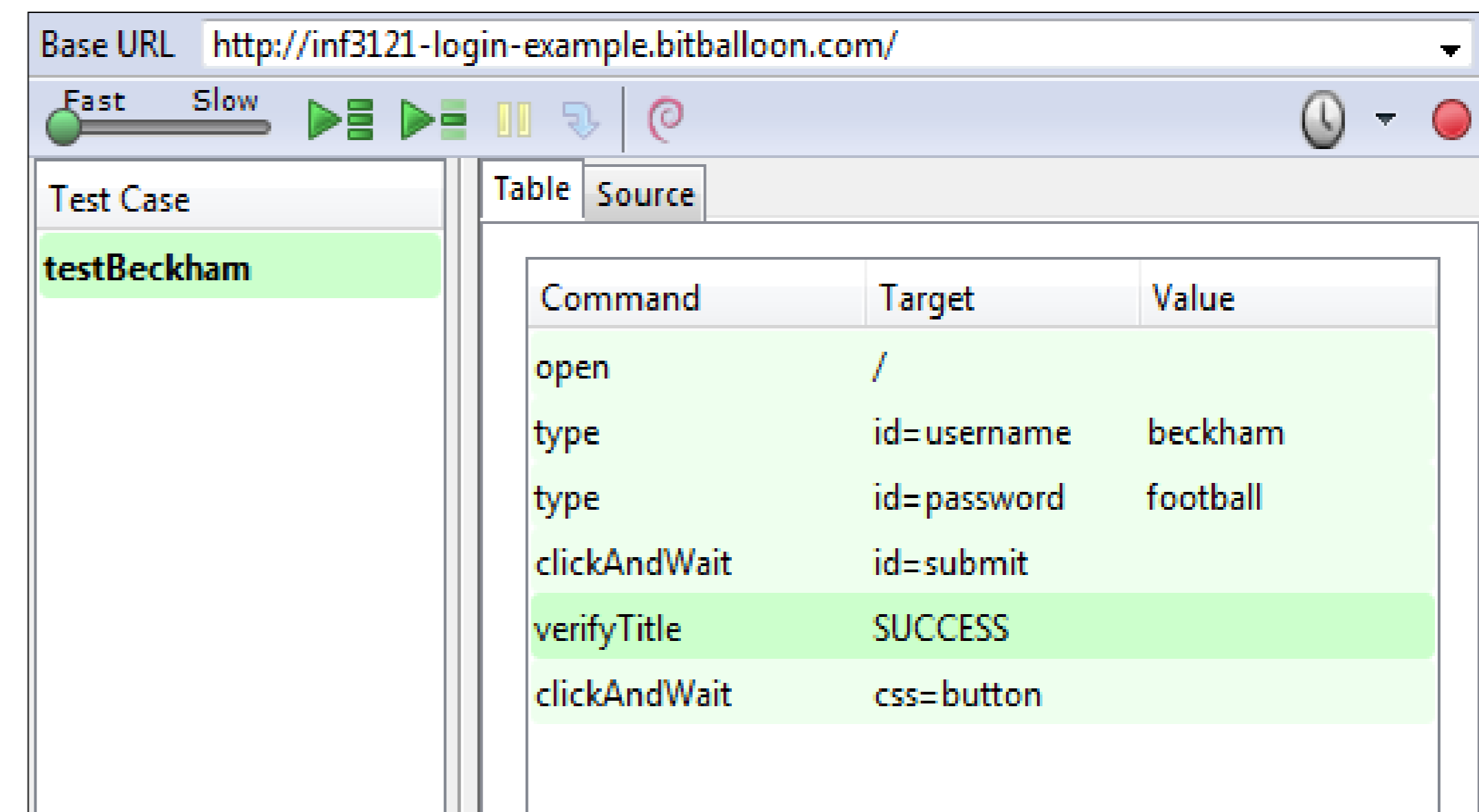
    driver.findElement(By.id("password")).clear();
    driver.findElement(By.id("password")).sendKeys("football");

    driver.findElement(By.id("submit")).click();

    try {
        assertEquals("SUCCESS", driver.getTitle());
    } catch (Error e) {
        verificationErrors.append(e.toString());
    }

    driver.findElement(By.cssSelector("button")).click();

    //driver.quit();
}
```



The screenshot shows the Selenium IDE interface. The Base URL is set to `http://inf3121-login-example.bitballoon.com/`. The Test Case list contains one entry, `testBeckham`. The Source table shows the following commands and targets:

Command	Target	Value
open	/	
type	id=username	beckham
type	id=password	football
clickAndWait	id=submit	
verifyTitle	SUCCESS	
clickAndWait	css=button	

# Question 16

Which of the following are **advantages of test automation**?

- a. Tests run faster and can be more complex
- b. Tests are run by machines and the results are interpreted by humans
- c. Data sets used in testing can be very simple
- d. The results of running the tests is always the same



# Question 16: Clues

Which of the following are **advantages of test automation?**

Run faster

More complex

Repeatability of tests

Objective assessment of results

More efficient use of tester

Use of external APIs and use of GUIs



# Question 17

Which of the following is a **limitation of test automation?**

- a. Tests can be very simple
- b. Tests need to be complex in order to be considered for automation
- c. One cannot automate all tests
- d. Data sets used in testing are not stored, therefore tests are not always reproducible



# Question 17: Clues

Which of the following is a **limitation** of **test automation**?

Computers can only do so much ...

You *cannot automate all* tests

## CAPTCHA

Completely Automated Public Turing test to tell Computers and Humans Apart

If you **could** automate it → CAPTCHA **poorly** implemented

## Environment- / Production-dependant scenarios

E.g. payment gateway **timeouts** → Depend on throughput capability of network

**Human factors** → Gestures, reactions, thought processes





# Question 18

**Pair** the following **approaches** to **automated testing** with their corresponding **description**:

Capture and Replay	The test inputs are extracted or generated with scripts. To automate testing, we reuse one main script together with this data to implement a number of tests.
Data-driven approach	The automated test scripts are built by putting together reusable smaller scripts, name keywords.
Keyword-driven approach	Tools are used to capture interactions with the system under test (SUT) while performing the sequence of actions as defined by a test procedure.

# Question 19

Which of the following **factors** must be **considered** when **transitioning** from **manual** to **automated** testing? Several answers might be correct

1. Frequency of use of the tested feature
2. The upcoming release date
3. How complex it is to automate the test
4. The current cyclomatic complexity of the code

# Question 19: Clues

Which of the following **factors** must be **considered** when **transitioning** from **manual** to **automated** testing?

## Frequency

Automation still takes **time**

If the tested **feature** is **rarely** used, **automating** could be *more* **time-consuming**

Opt to automate **repetitive** tests

## Complexity

Automation requires **skill**

**Unskilled** testers may **slow** down the **testing process** if required to automate

Assess the skill level of potential testers



# Question 20

A test **manager** does **not need** to take into account **re-educating** the team when **preparing** to go from **manual** to **automated** testing

- a. True
- b. False



# Question 20: Clues

There is **no need** to **assess re-education** of the team when **preparing** to go from **manual** to **automated** testing

Typical tasks of a test **manager**

Responsible for **project management** of the test effort

**Directs, controls, administers, plans, regulates** the test effort and objects

Decide **what** should be **automated**, to what **degree**, and **how**

Introduce suitable **metrics** for progress **monitoring** and **quality** assessment

Make **decisions** about the **implementation** of the test **environment**

**Select tools** to support testing

**Organise** any **training** (re-education) and mentoring for **testers**



# Part II: Exercises and Open-ended questions

# Exercise 1: Benefits and Limitations

Discuss the **advantages** and **limitations** of **automated** testing.

