Ingineria Programării

Cursul 10 – 24 Aprilie

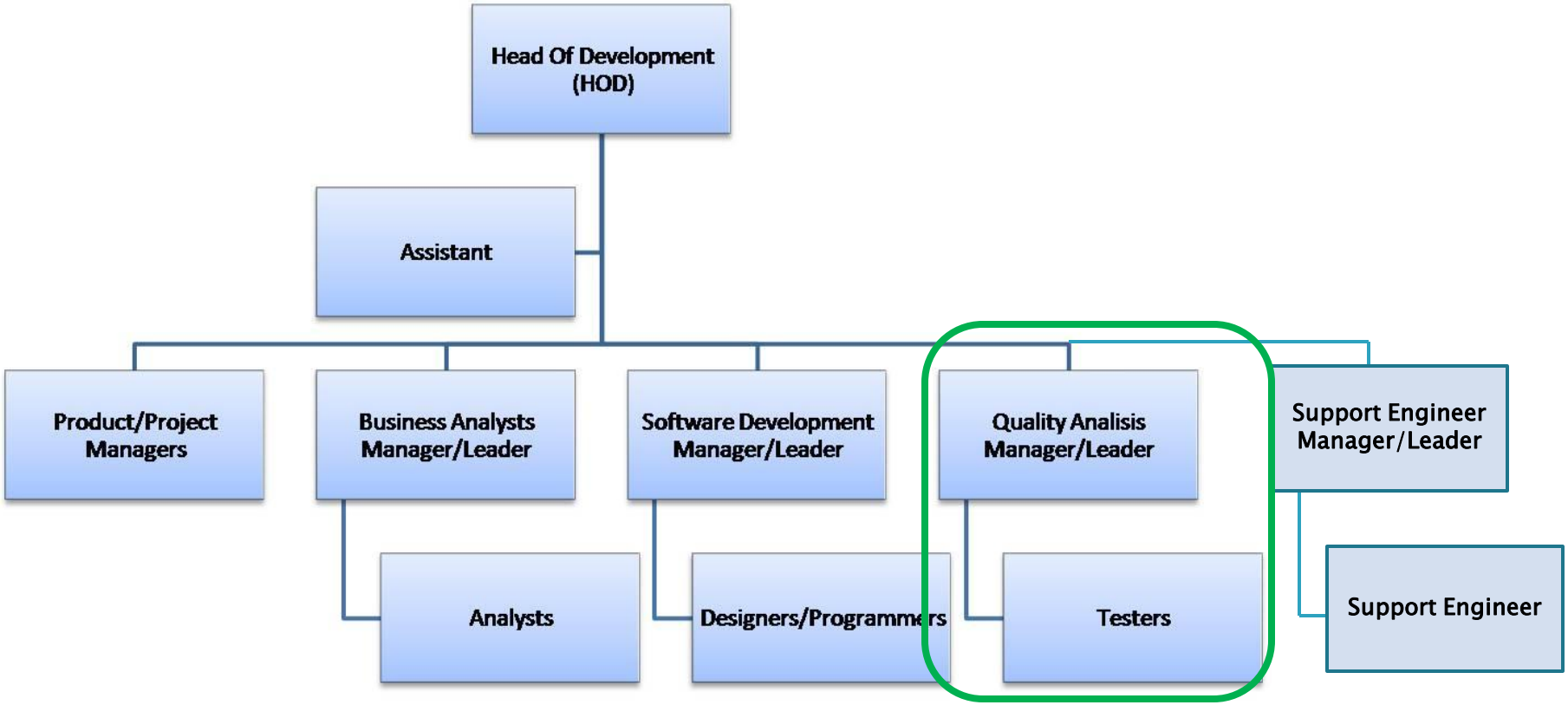
Cuprins

- Din Cursurile trecute…
  - Quality Assurance
  - Test Levels
  - Test Methods
- Quality Assurance
  - Manual Testing
  - Test Automation
  - Software Bug
  - Non functional software testing
  - Measuring software testing
  - Testing artifacts
  - Testing cycle

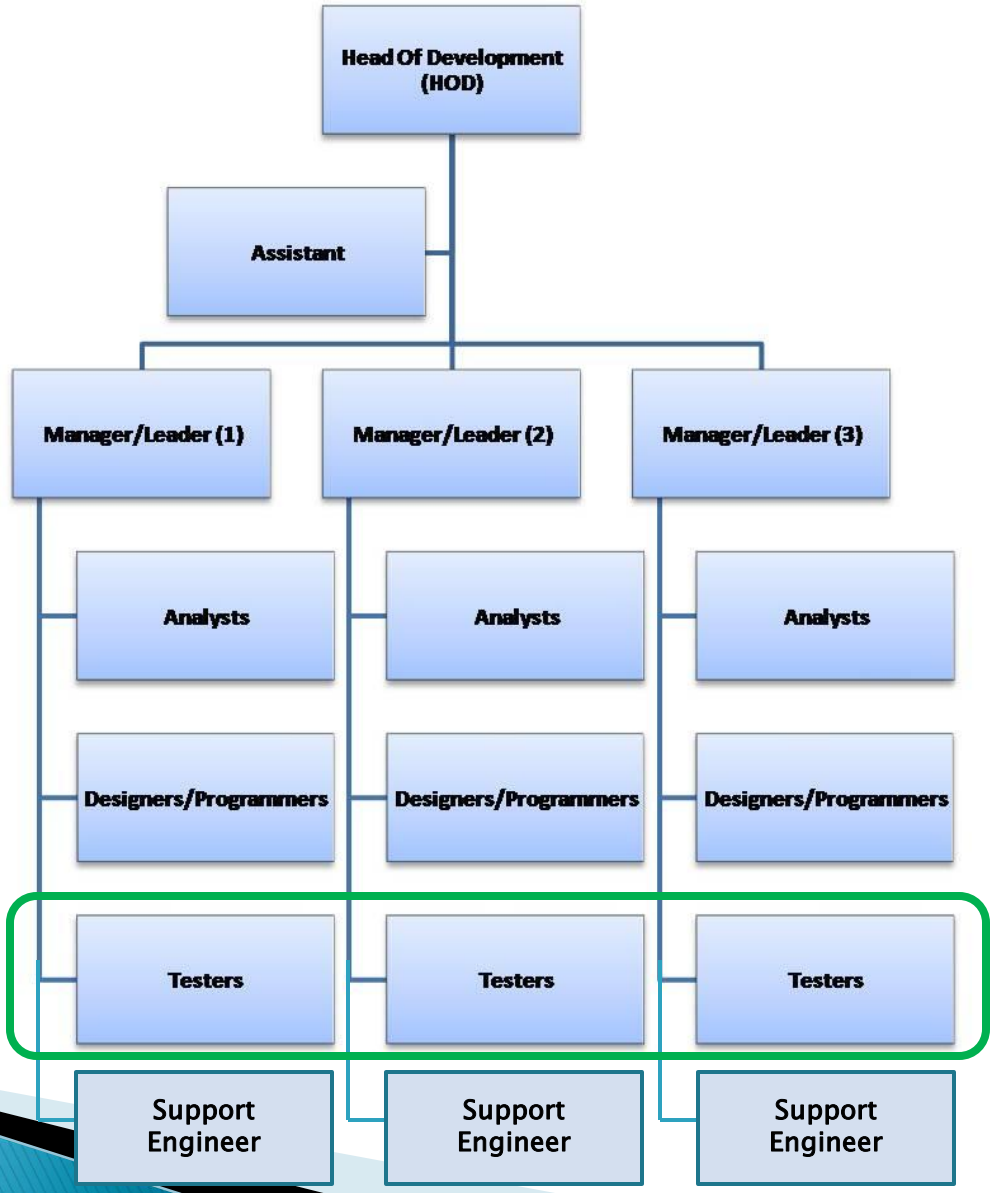
Din Cursurile Trecute

- Quality Assurance: When? Aims?
- Testare Profesională
- Nivele de Test:
  - Unitate, Modul, Integrare, Sistem, Acceptare
- Metode de Testare:
  - White Box, Black Box, Gray Box, GUI, Acceptance, Regression
- Testare Manuala vs Testare Automata

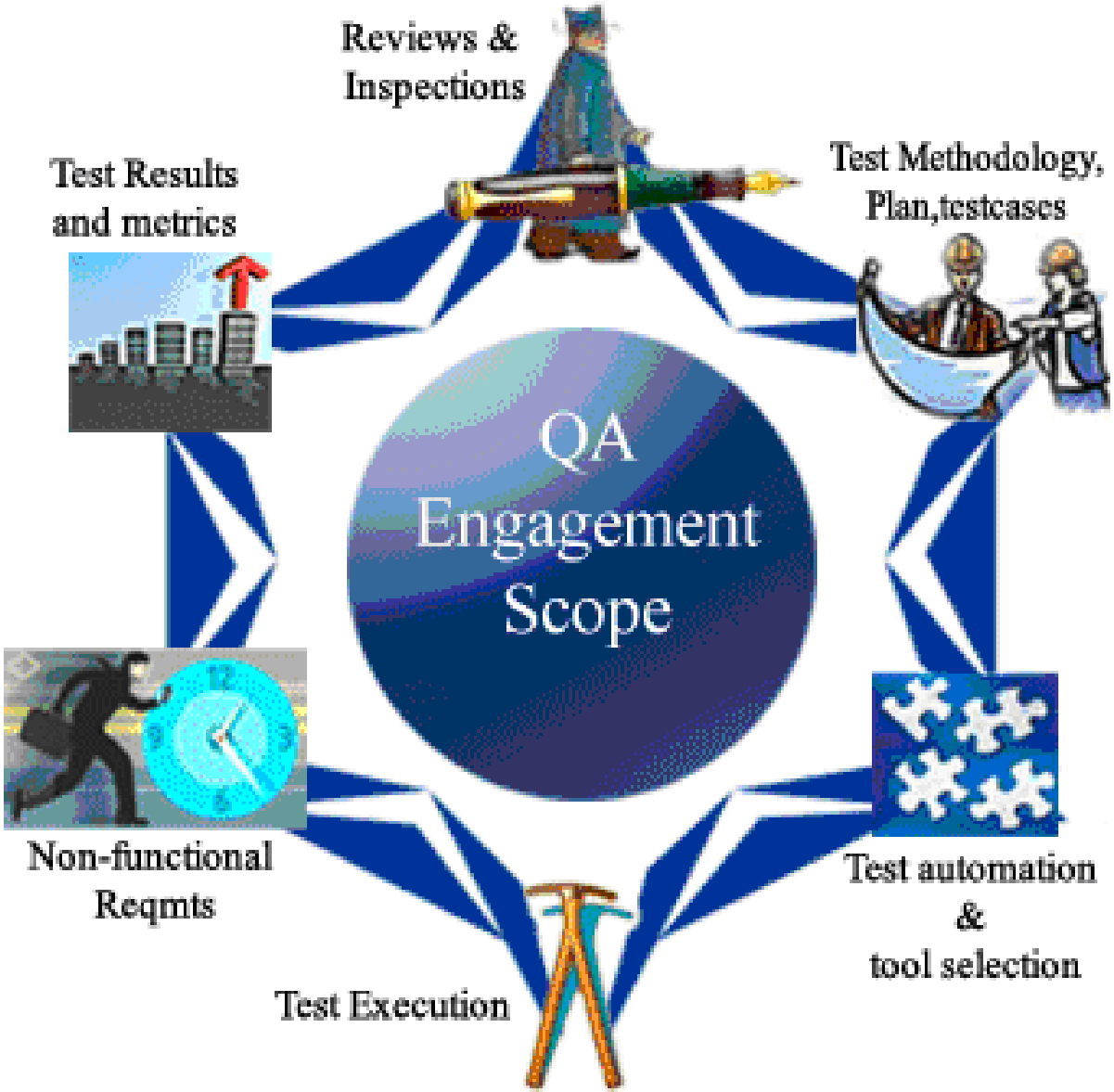
Ierarhia dintr-o firmă de software – Compartimente



Ierarhia dintr-o firmă de software - Proiecte

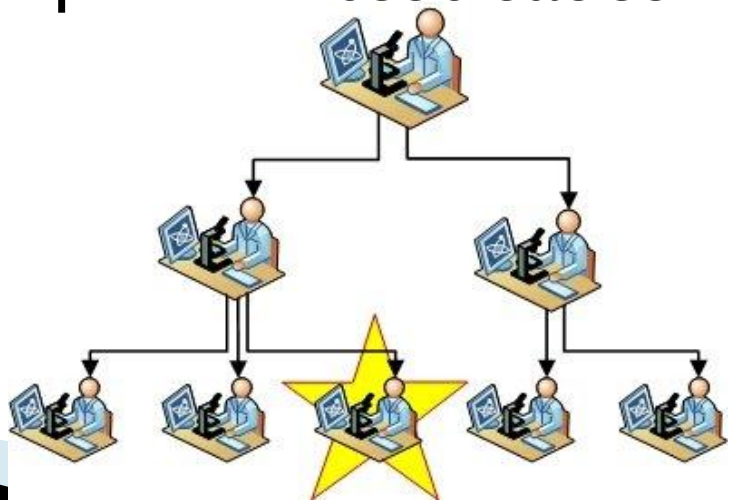


QA Engagement Scope



Manual Testing

- *Manual testing is the process of manually testing software for defects*
- It requires a **tester** to play the role of an end user, and use most of all features of the application to ensure correct behavior
- To ensure completeness of testing, the tester often follows a written **test plan** that leads them through a set of important **test cases**



Definitions

- **Test Strategy** is developed by the "Project manager" which contains what type of technique to follow and which module to test
- **Test Plan** is developed by the Test Lead, which contains "what to test", "how to test", "when to test", "who to test"
- **Test Scenario** is a name given to test case. It is dealt with by the Test Engineer
- **Test Case** specifies a testable condition to validate functionality. The test cases are deal by Test Engineer

Test Plan

- **A systematic approach to testing a system**
- Contains a detailed understanding of what the eventual workflow will be
- Documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements
- Is usually prepared by or with significant input from Test Engineers

Test Plan Components

- May include one or more of the following:
  - *Design Verification or Compliance test*
  - *Manufacturing or Production test*
  - *Acceptance or Commissioning test*
  - *Service and Repair test*
  - *Regression test*

Test Plan Structure (IEEE 829-1998)

- Test plan identifier
- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Approach
- Item pass/fail criteria
- Suspension criteria
- Test deliverables
- Testing tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Approvals

Test Plan – Life Cycle

Test Case

- **A set of conditions or variables** under which a tester will determine whether an application or software system meets specifications
- **A sequence of steps** to test the correct behavior/functionalities, features of an application
- In order to fully test that all the requirements of an application are met, there must be *at least one test case for each requirement* (*two recommended*)

Test Case Format

- Test case ID
- Test case Description
- Expected Output
- Actual Output
- Pass/Fail
- Remarks
- Test step or order of execution number
- Related requirement(s)
- Depth
- Test category
- Author
- Check boxes for whether the test is automatable and has been automated.

Large Scale Engineering Projects

- Need a systematic approach:
  1. Choose a high level test plan
  2. Write detailed test cases
  3. Assign the test cases to testers, who manually follow the steps and record the results.
  4. Author a test report, detailing the findings of the testers.

- The report is used by managers to determine whether the software can be released

Test Automation

- **A process of writing a computer program to do testing** that would otherwise need to be done manually
- **The use of software to control the execution of tests**, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions
- Commonly, test automation involves automating a manual process already in place that uses a formalized testing process

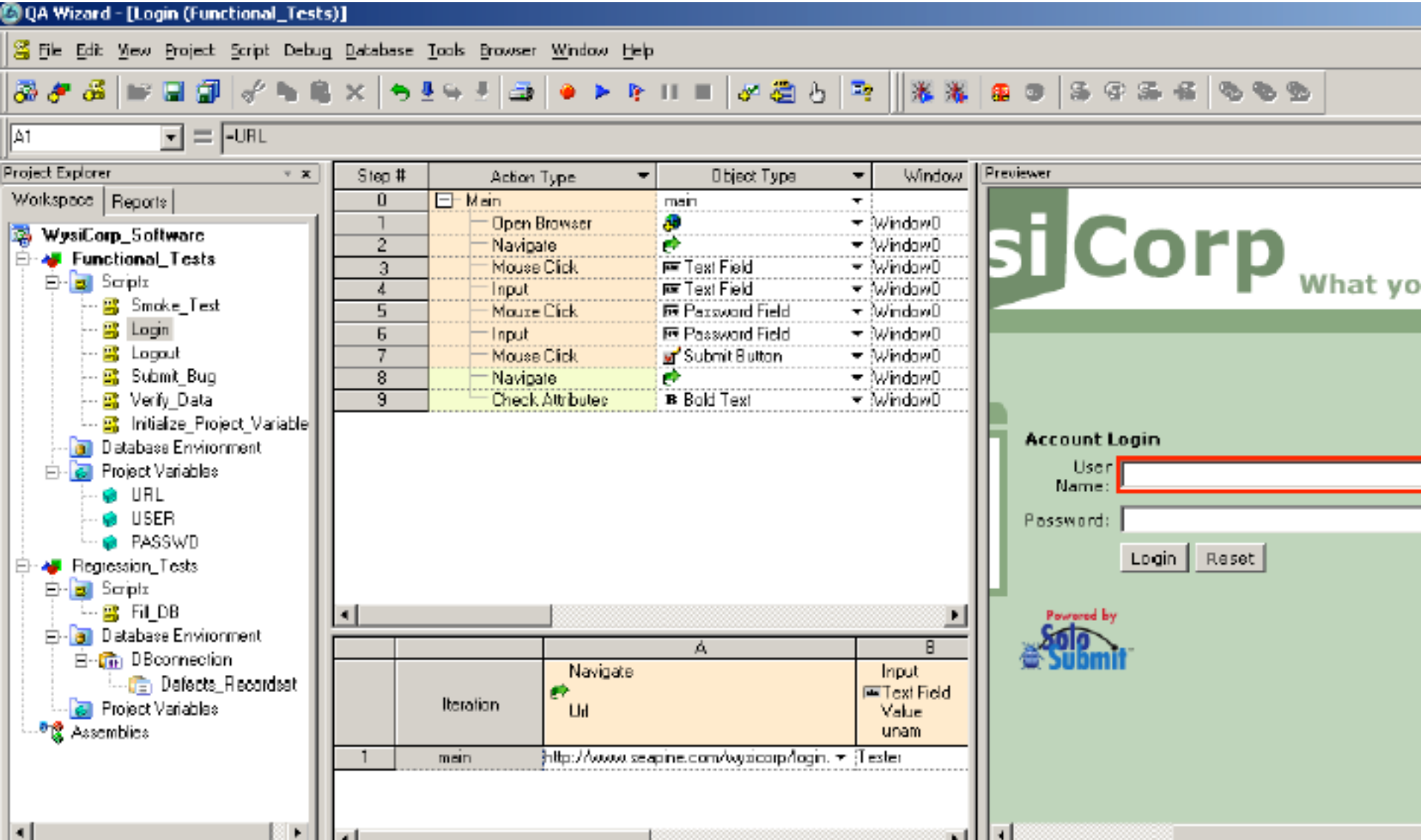
Test Automation – Approaches

- **Graphical user interface testing**. A testing framework generates user interface events such as keystrokes and mouse clicks, and **observes the changes** that result in the user interface, to validate that the observable behavior of the program is correct
- **Code-driven testing**. The public (usually) interface to classes, modules, or libraries are tested with a variety of input arguments to validate that the results that are returned are correct

TA – What to test

- Testing tools can help automate tasks such as *product installation, test data creation, GUI interaction, problem detection, defect logging*, etc.
- Important points when thinking about TA:
  - Platform and OS independence
  - Data driven capability (Input Data, Output Data, Meta Data)
  - Customizable Reporting (DB Access, crystal reports)
  - Email Notifications
  - Easy debugging and logging
  - Version control friendly
  - Extensible & Customizable
  - Support distributed execution environment
  - Distributed application support

Test Automation – Example



Software Bug

- **A software bug is an error, flaw, mistake, failure, or fault in a computer program**
- Most bugs arise from mistakes and errors made by people (in program or in its design), and a few are caused by compilers
- Reports detailing bugs in a program are commonly known as **bug reports**, **fault reports, problem reports, trouble reports, change requests**, and so forth.

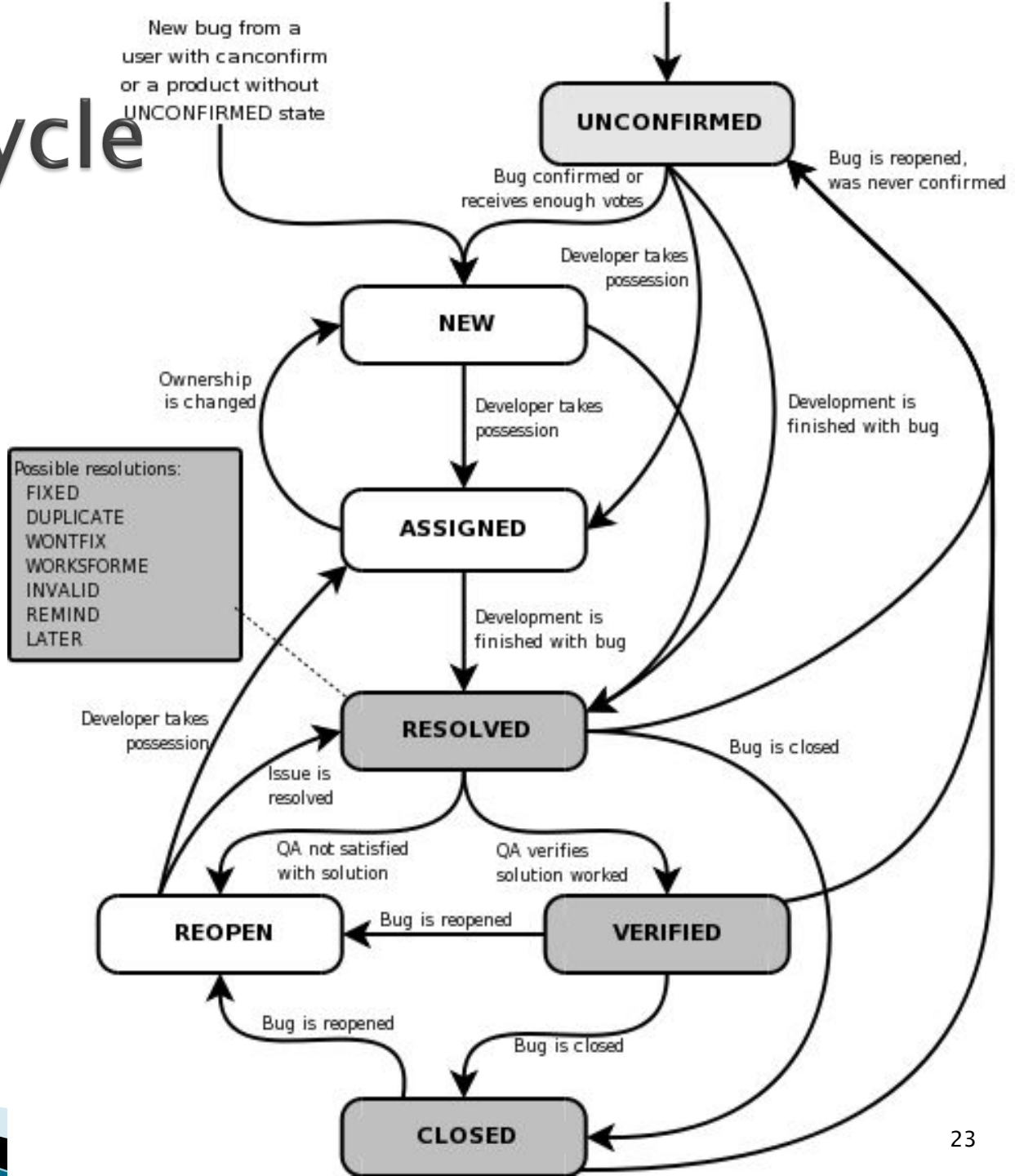
Bugs Effects

- Bugs can have only a subtle effect on the program's functionality
- More serious bugs may cause the program to crash or freeze leading to a denial of service. Others qualify as security bugs
- Bugs in the code controlling the Therac-25 radiation therapy machine were directly responsible for some patient deaths in the 1980s
- In 2002, a study commissioned by the US DCNIST concluded that software bugs, or errors cost the US economy an estimated $59 billion annually (0.6% of gross domestic product)

Bug Prevention

- Programming style
- Programming techniques
- Development methodologies
- Programming language support
- Code analysis
- Instrumentation

Bug Life Cycle



Non Functional Software Testing

- Verifies that the software functions properly even when it receives invalid or unexpected inputs
- Example: software fault injection (fuzzy form)
- Methods:
  - **Performance testing** or **Load Testing** checks to see if the software can handle large quantities of data or users (software scalability).
  - **Usability testing** checks if the user interface is easy to use and understand.
  - **Security testing** is essential for software which processes confidential data and to prevent system intrusion by hackers.
  - **Internationalization and localization** is needed to test these aspects of software, for which a pseudo localization method can be used.

Software Performance Testing

- **Types**
  - **load testing** – can be the expected concurrent number of users on the application (database is monitored)
  - **stress testing** – is used to break the application (2 x users, extreme load) (application's robustness)
  - **endurance testing** – if the application can sustain the continuous expected load (for memory leaks)
  - **spike testing** – spiking the number of users and understanding the behavior of the application whether it will go down or will it be able to handle dramatic changes in load

Performance Testing Analysis

Usability testing

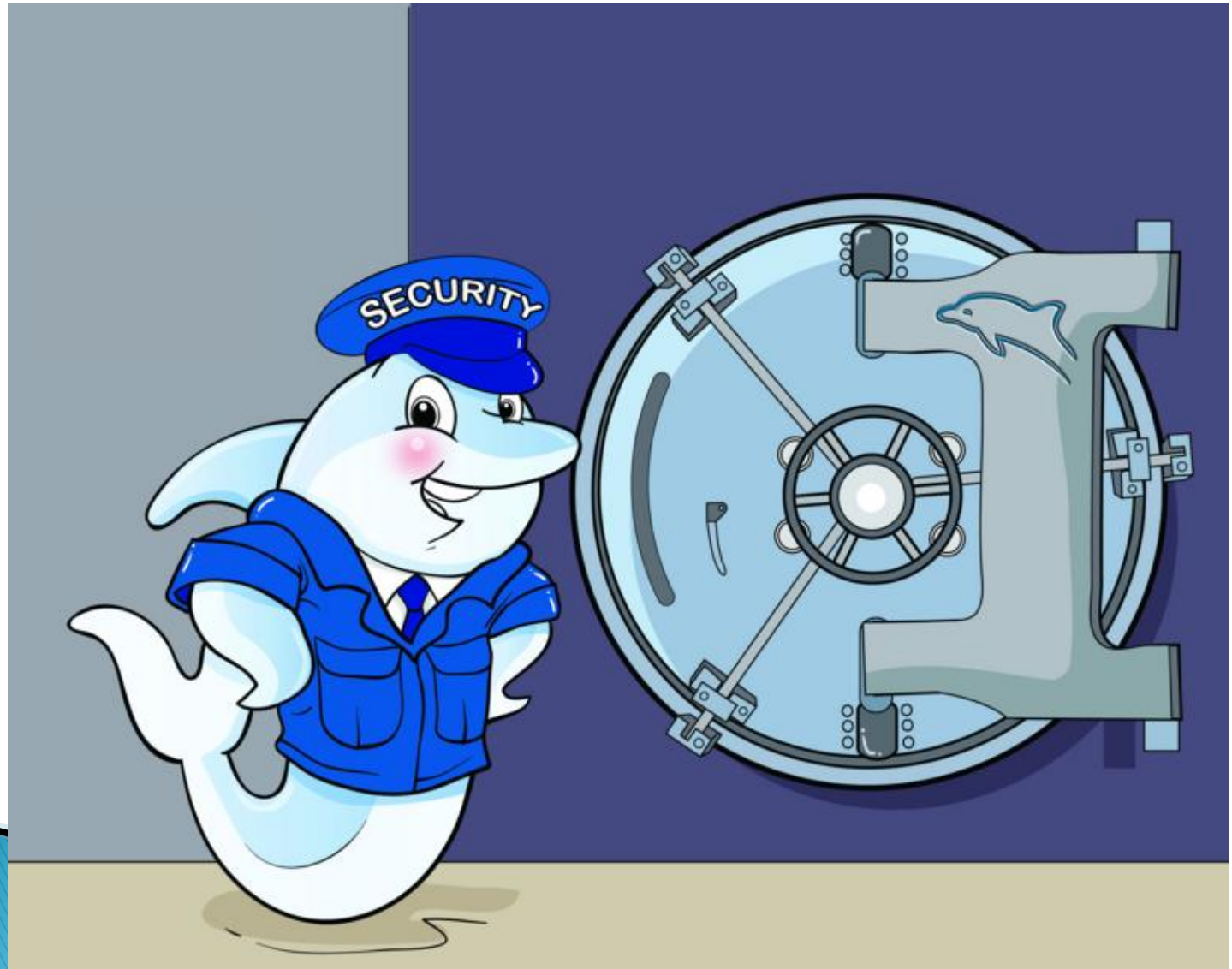
- A technique used to evaluate a product by testing it on users
- Usability testing focuses on measuring a human-made product's capacity to meet its intended purpose.
- **Examples** of products that commonly benefit from usability testing are web sites or web applications, computer interfaces, documents, or devices
- **Goals**
  - *Performance* – How much time, steps?
  - *Accuracy* – How many mistakes/fatal did people make?
  - *Recall* – How much does the person remember afterwards or after periods of non-use?
  - *Emotional response* – How does the person feel about the tasks completed? Is the person confident, stressed? Would the user recommend this system to a friend?

Usability testing steps

Security testing

- The Process to determine that an Information System protects data and maintains functionality as intended.
- The six basic security concepts that need to be covered by security testing are:
  - **Confidentiality**,
  - **Integrity** – information which it receives has not been altered in transit or by other than the originator of the information
  - **Authentication** – validity of a transmission, message, or originator,
  - **Authorization** – determining that a requester is allowed to receive a service or perform an operation,
  - **Availability** – Assuring information and communications services will be ready for use when expected,
  - **Non-repudiation** – prevent the later denial that an action happened, or a communication that took place

Security logo



Internationalization and localization

- Means of adapting computer software to different languages and regional differences
- **Internationalization** is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes.
- **Localization** is the process of adapting software for a specific region or language by adding locale-specific components and translating text.

Measuring software testing

- Usually, quality is constrained to such topics as **correctness, completeness, security**
- Can also include capability, reliability, efficiency, portability, maintainability, compatibility, and usability
- There are a number of common software measures, often called "metrics", which are used to measure the state of the software or the adequacy of the testing.

Testing artifacts

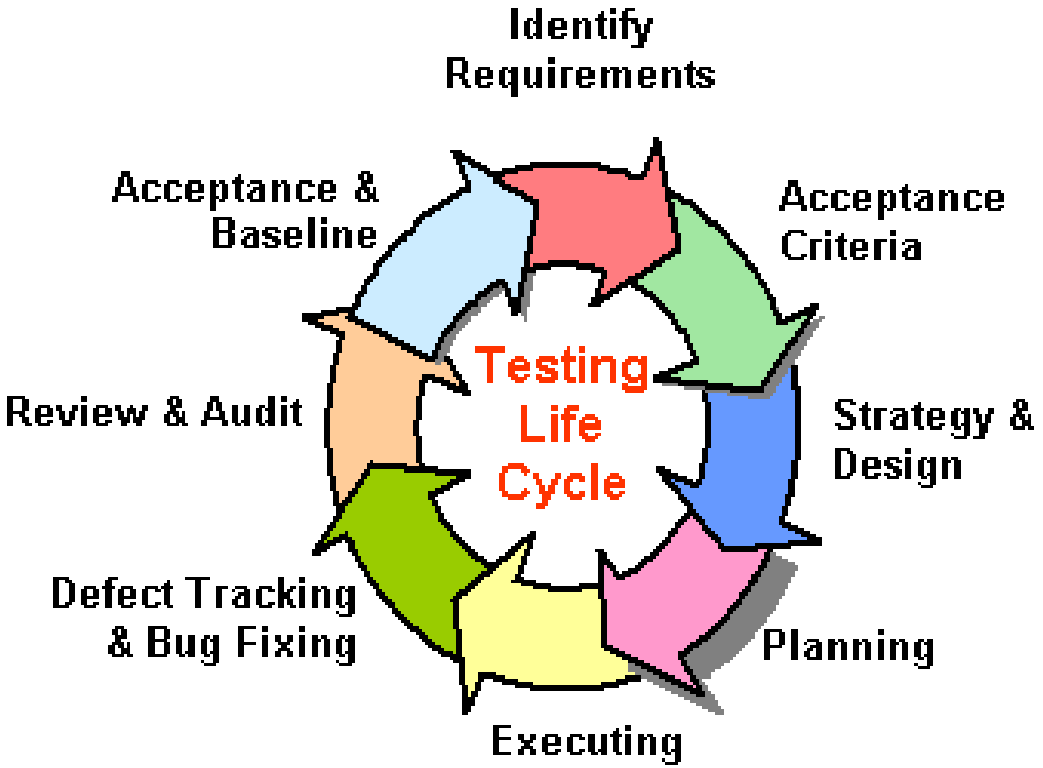
- **Test case** – consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output, expected result, and actual result
- **Test script** – the combination of a test case, test procedure, and test data
- **Test data** – multiple sets of values or data are used to test the functionality of a particular feature

Testing artifacts (2)

- **Test suite** – a collection of test cases
- **Test plan** – A test specification
- **Test harness** – The software, tools, samples of data input and output, and configurations

Testing cycle

- There is a typical cycle for testing: Requirements Analysis, Test Planning, Test Development, Test Reporting, Test Result Analysis, Retesting the Resolved Defects, Regression Testing, Test Closure



Correctness

- Correctness of an algorithm is asserted when it is said that **the algorithm is correct with respect to a specification**
- **Functional correctness** refers to the input-output behavior of the algorithm (i.e., for each input it produces the correct output)
- A distinction is made between **total correctness**, which additionally requires that the algorithm terminates, and **partial correctness**, which simply requires that *if* an answer is returned it will be correct.

Studiu de Caz 1

- Ce trebuie în general testat pentru tipurile de câmpuri de mai jos (test, regression, acceptance):
  - Alfanumeric
  - Characters
  - Numeric
  - Date

Studiu de Caz 2

- Realizaţi scenarii de test pentru înregistrarea utilizatorilor
- UserID, Password, ConfirmPassword

- Test, Regression, Acceptance
- Testare automată

Studiu de Caz 3

- Ce apare în plus pentru modulul de căutare când se folosesc câmpurile de tip dată From Date, To Date?

Links

- Software Bug: http://en.wikipedia.org/wiki/Software_bug
- http://en.wikipedia.org/wiki/Manual_testing
- http://en.wikipedia.org/wiki/Test_automation
- HP: BTO Software – Download Center https://h10078.www1.hp.com/cda/hpdc/display/main/search_results.jsp?zn=bto&cp=54_4012_100__