ILLINOIS TECH | College of Computing

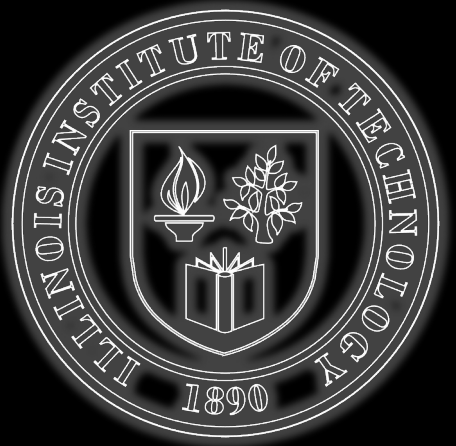
ITMD 536 Software Testing & Maintenance

Nazneen Hashmi

Call or Text: 312-498-8387

IIT email: nhashmi@iit.edu

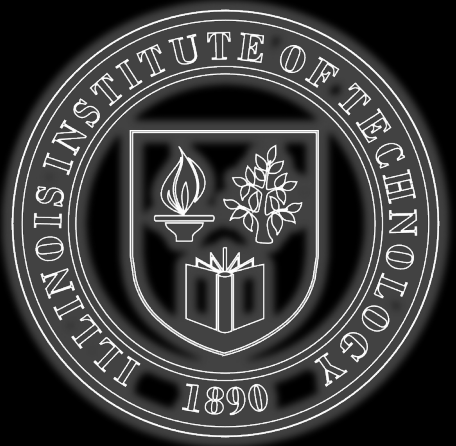
Time 10:00 AM to 12:40 PM



ILLINOIS TECH | College of Computing

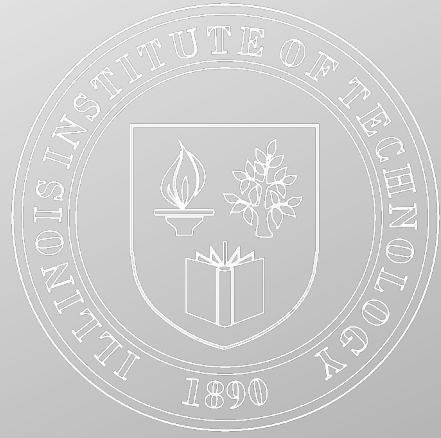
ITMD 536 Software Testing & Maintenance

02 Testing Throughout the Software Life Cycle



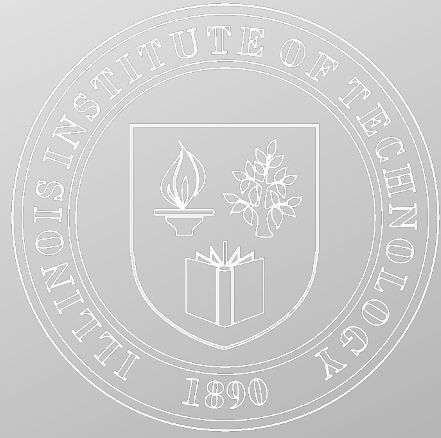
Objectives

- What is beyond Unit Testing?
- What is testing throughout the software life cycle
- What is the difference between Waterfall, V-Model & Agile Testing?
- What are the Test different Levels?
- What are the different Test Types?
- What is Maintenance Testing?



Testing throughout the software life cycle

- Test is not a stand-alone activity
- There are many forms of testing
- There are various models, test levels and types of testing
- Maintenance testing

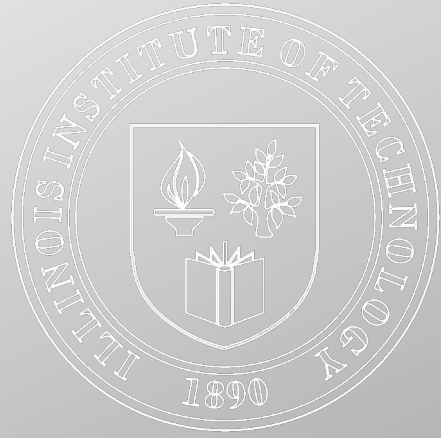


2.1 Software Development Models

- Testing is focused on verification and validation testing
- **Verification:** Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled (ISO)
- **Validation:** Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have fulfilled.

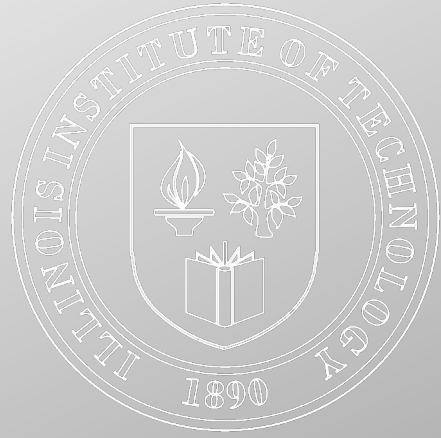
2.1 Software Development Models

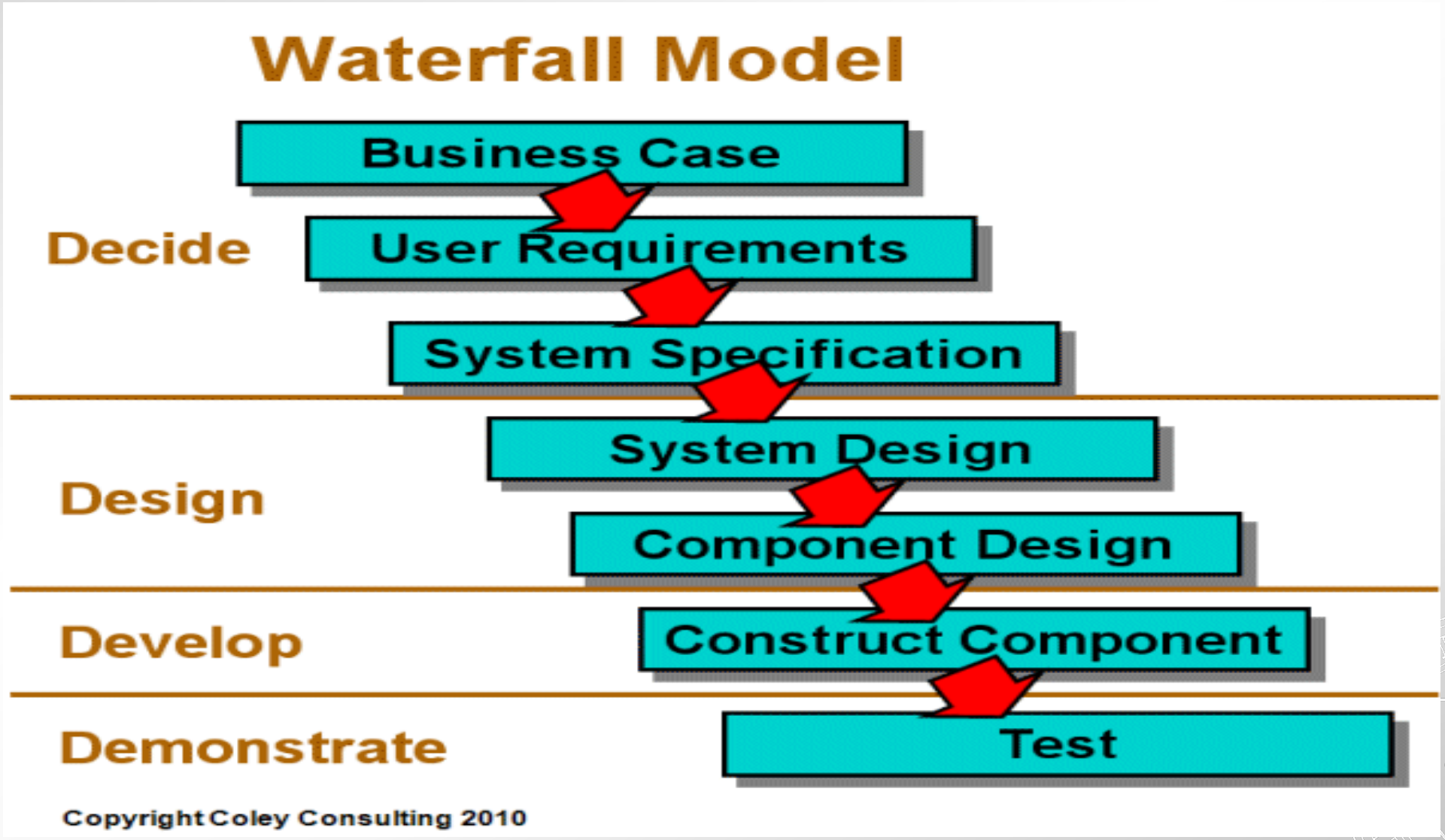
- **Waterfall Testing** is the top-down development and design by functional decomposition.



Waterfall Model

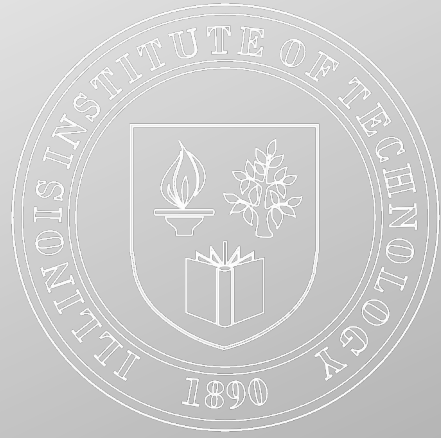
- Need, wish, policy, law
- User Requirements,
- System Requirements,
- Global Design,
- Detail Design,
- Implementation
- Testing




Waterfall Model
Business Case
Decide
User Requirements
System Specification
System Design
Design
Component Design
Develop
Construct Component
Demonstrate
Test
Copyright Coley Consulting 2010

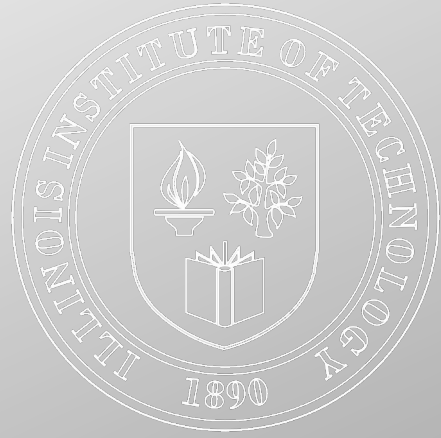
V-Model Testing

- **V-Model Testing** - A framework to describe the software development lifecycle activities from requirements specification to maintenance. The V-model illustrates how testing activities can be integrated into each phase of the software development lifecycle.



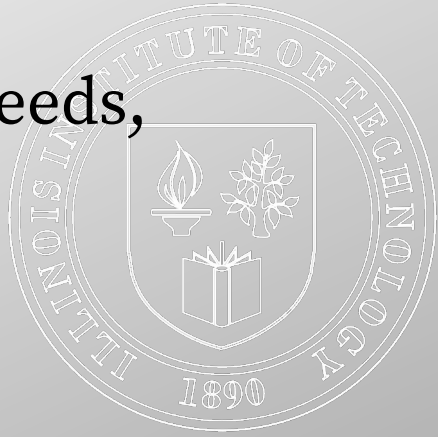
V-Model Test Level

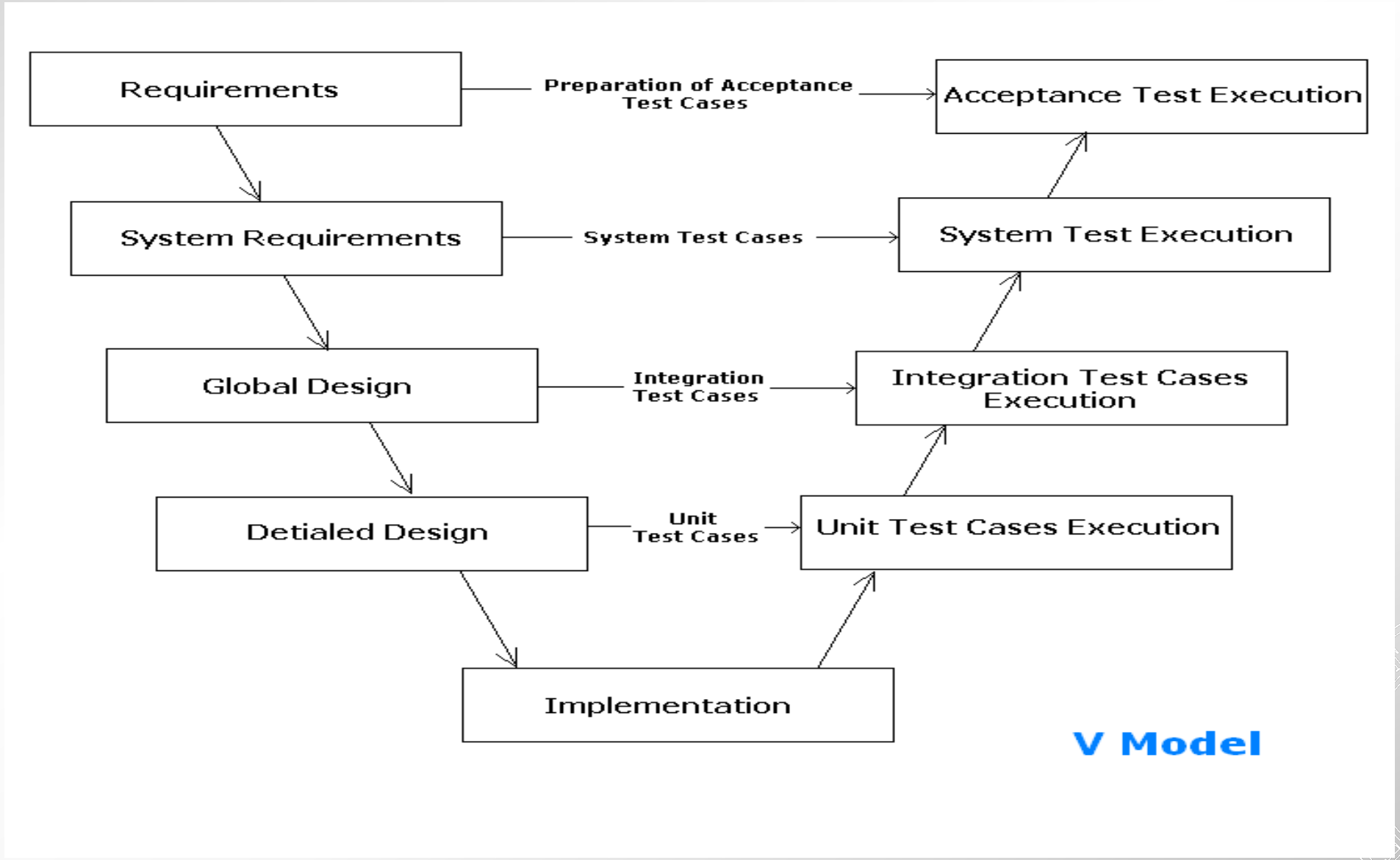
- A group of test activities that are organized and managed together.
- A test level is linked to the responsibilities in a project.
- Examples of test levels are component test, integration test, system test and acceptance test



V-Model uses four levels of testing

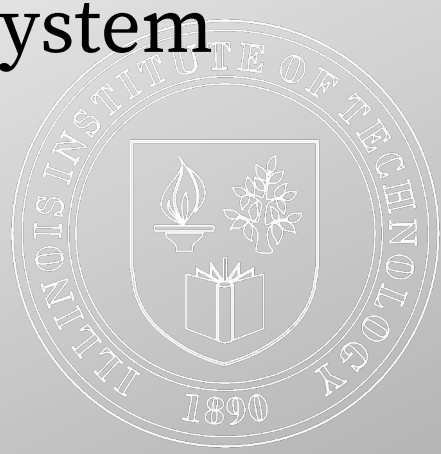
- **Component testing** – searching for defects in and verifies the functioning of software components (modules, programs, objects, classes, etc.)
- **Integration testing** – The process of combining components or systems into a large assemblies.
- **System testing** – The process of testing an integrated system to verify that it meets specified requirements.
- **Acceptance testing** – Validation testing with respect to user needs, requirements and business processes conducted to determine whether or not to accept the system.




Requirements
Preparation of Acceptance Test Cases
Acceptance Test Execution
System Requirements
System Test Cases
System Test Execution
Global Design
Integration Test Cases
Integration Test Cases Execution
Detialed Design
Unit Test Cases
Unit Test Cases Execution
Implementation
V Model

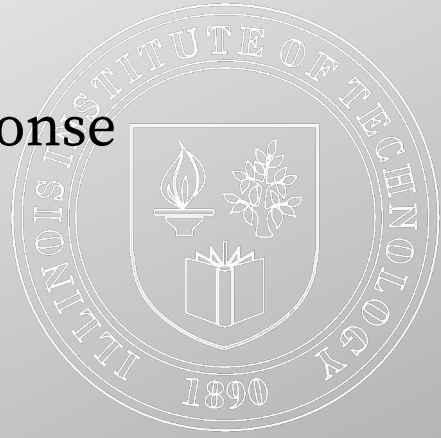
Commercial off-the-shelf software - COTS

- A software product that is developed for the general market, i.e. for a large number of customers, and that is delivered to many customers in identical format.
- Purchaser may only perform integration testing and may be acceptance testing.
- The acceptance testing can include both testing of system functions but also testing quality attributes such as performance and other non-functional tests.



Performance Testing

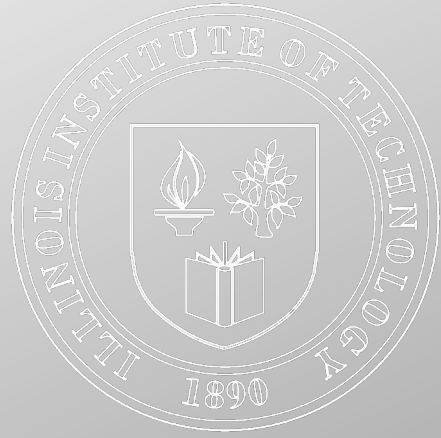
- **Performance testing** – The degree to which a system or component accomplishes its designated functions within given constraints regarding processing time and throughput rate.
- The process of testing to determine the performance of a software product.
- Load generation can simulate either multiple users or high volumes of input data.
- During execution, response time measurements are taken from selected transactions and these are logged.
- Performance testing provides test logs and graphs of load against response times.

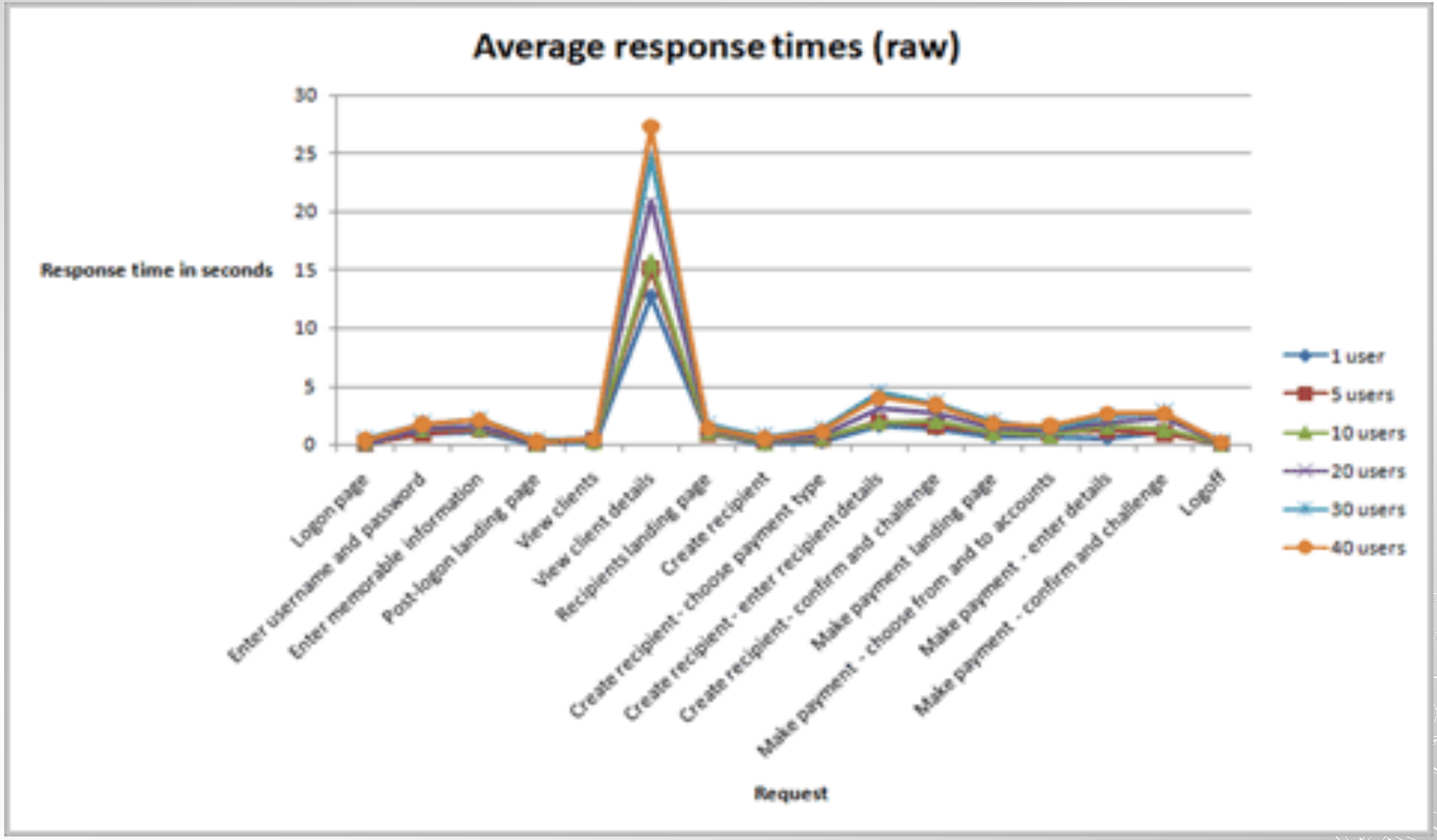


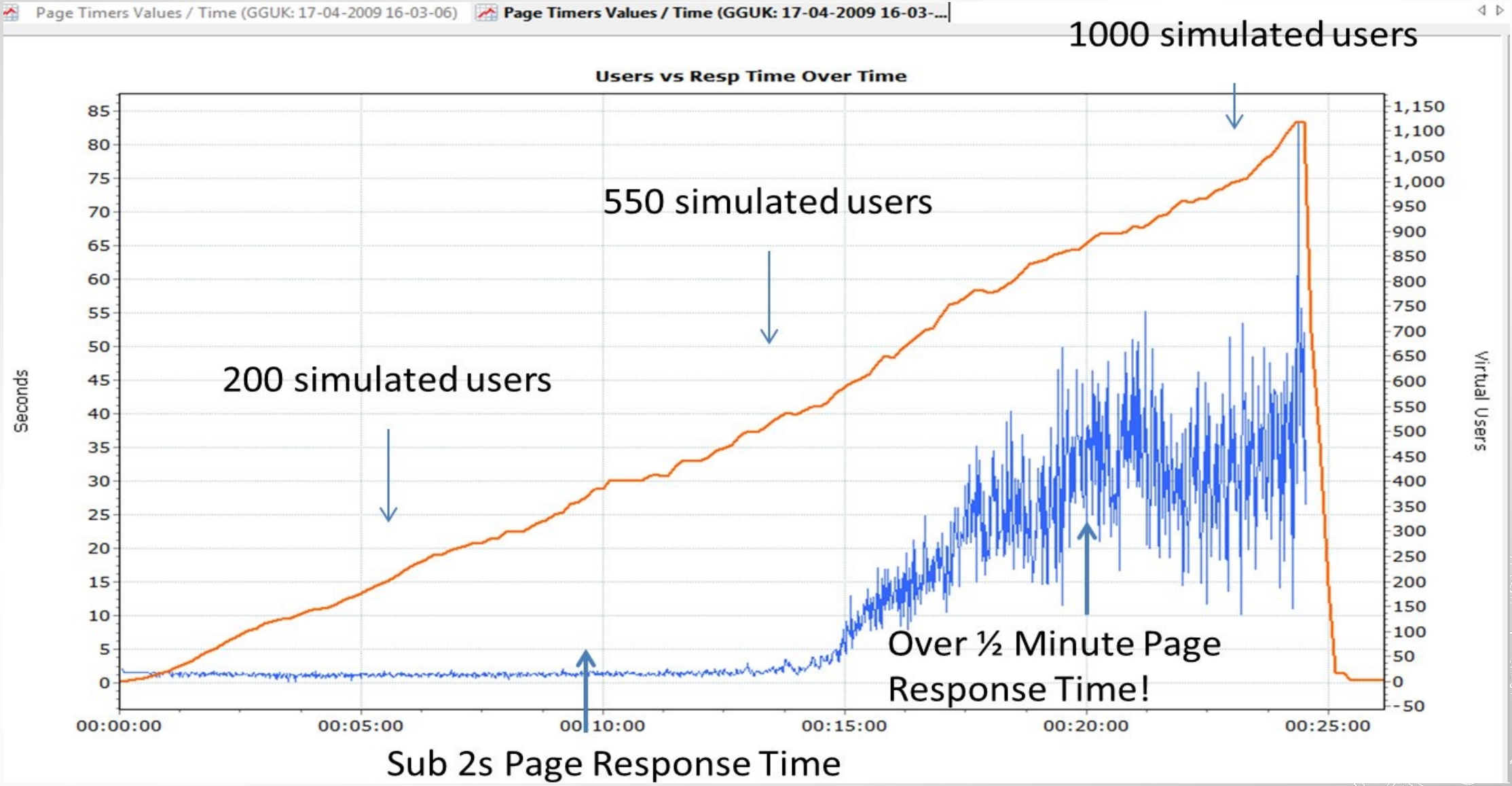
Performance – Stress Testing

Load Testing

- A type of performance testing conducted to evaluate the behavior of a component or system with increasing load, e.g., numbers of parallel users and/or numbers of transactions, to determine what load can be handled by the component or system.

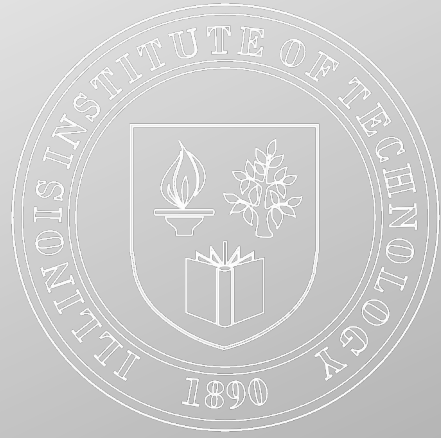


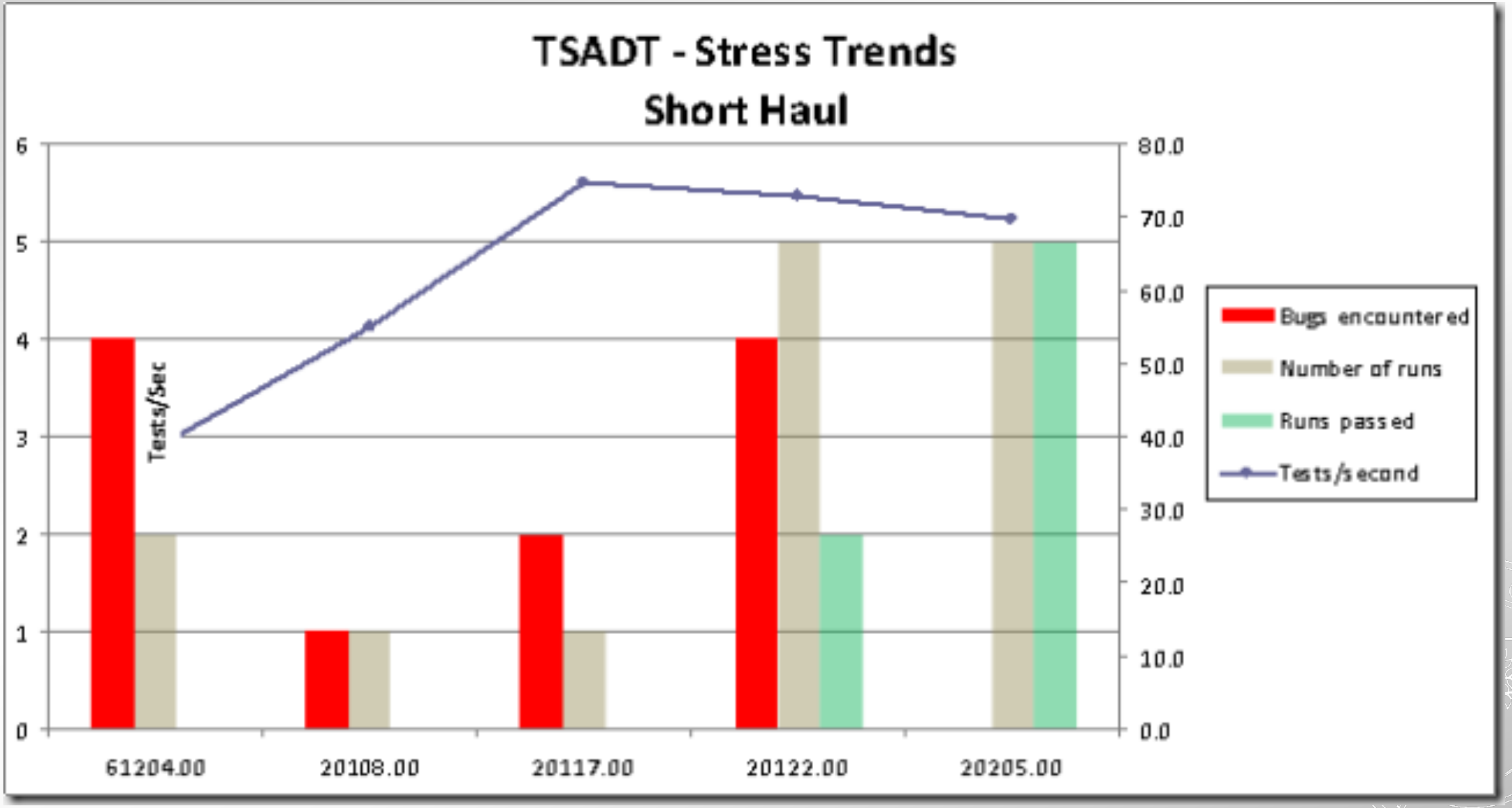

Average response times (raw)
Response time in seconds
30
25
20
15
10
5
0
Logon page
Enter username and password
Enter memorable information
Post-logon landing page
View clients
View client details
Recipients landing page
Create recipient
Create recipient - choose payment type
Create recipient - enter recipient details
Create recipient - confirm and challenge
Make payment landing page
Make payment - choose from and to accounts
Make payment - enter details
Make payment - confirm and challenge
Logoff
Request
1 user
5 users
10 users
20 users
30 users
40 users


Page Timers Values / Time (GGUK: 17-04-2009 16-03-06)
Page Timers Values / Time (GGUK: 17-04-2009 16-03-...
Users vs Resp Time Over Time
1000 simulated users
550 simulated users
200 simulated users
Seconds
Virtual Users
Over ½ Minute Page Response Time!
Sub 2s Page Response Time
00:00:00
00:05:00
00:10:00
00:15:00
00:20:00
00:25:00

Stress Testing

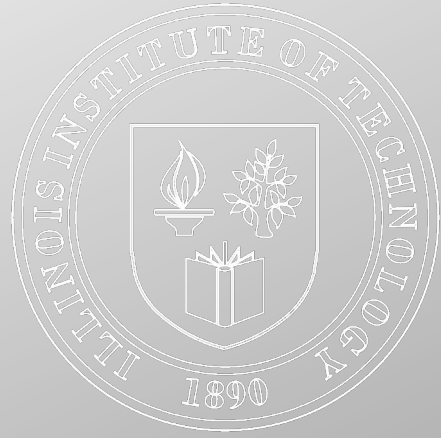
- A type of performance testing conducted to evaluate a system or component at or beyond the limits of its anticipated or specified work loads, or with reduced availability of resources such as access to memory or servers.




TSADT - Stress Trends
Short Haul
Tests/Sec
6
5
4
3
2
1
0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
61204.00
20108.00
20117.00
20122.00
20205.00
Bugs encountered
Number of runs
Runs passed
Tests/second

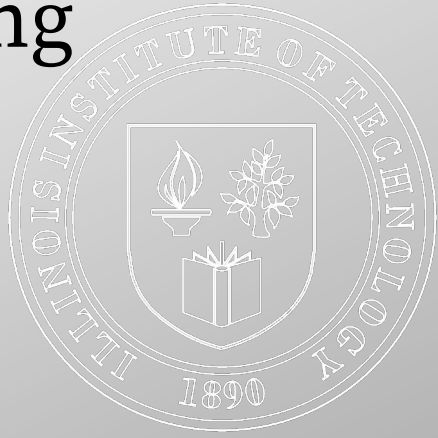
Capability Maturity Model (CMM)

- A first level staged framework that describes the key elements of an effective software process.
- The Capability Maturity Model covers best-practices for planning, engineering and managing software development and maintenance.



Capability Maturity Model Integration (CMMi)

- A framework that describes the key elements of an effective product development and maintenance process.
- The Capability Maturity Model Integration covers best-practices for planning, engineering and managing product development and maintenance.



Iterative life cycles

- **Iterative development model** – A development lifecycle where a project is broken into a usually large number of iterations. An iteration is a complete development loop resulting in a release (internal or external) of an executable product, a subset of final product and grows into final product.

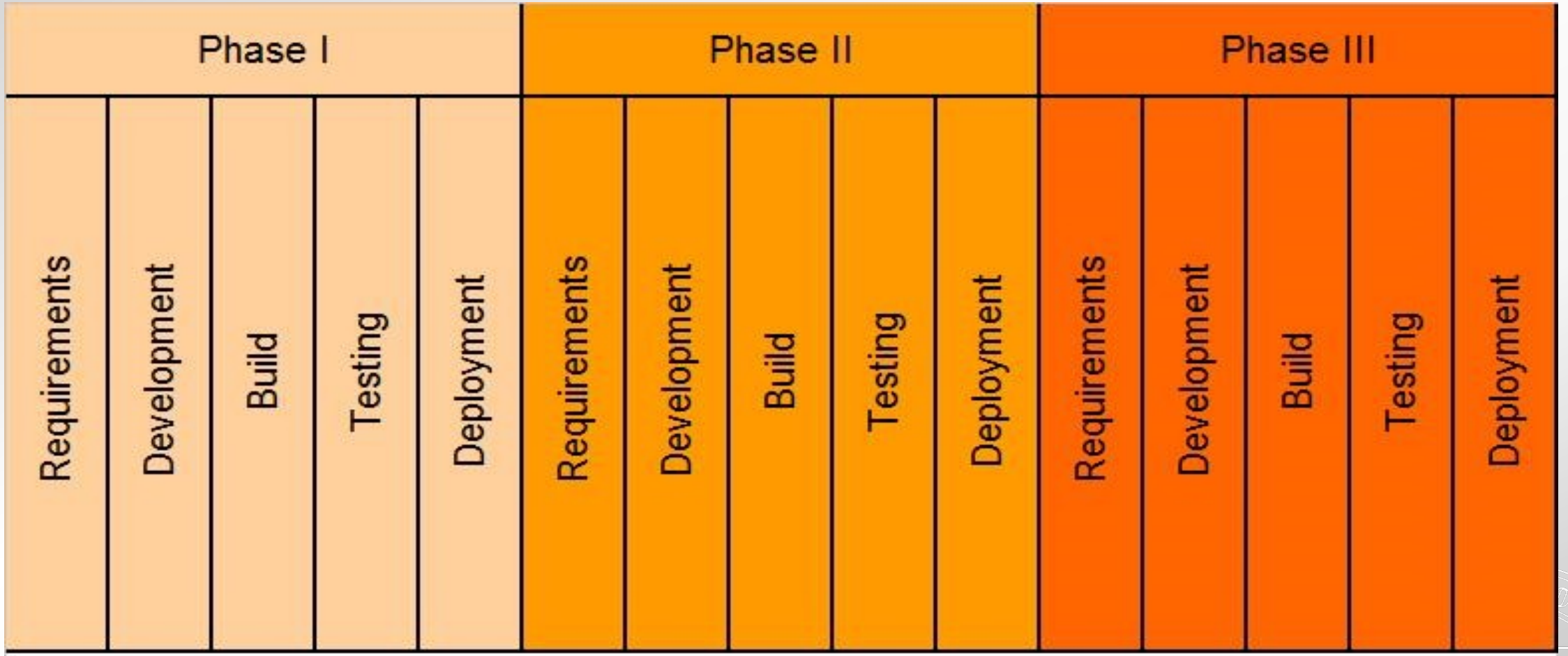


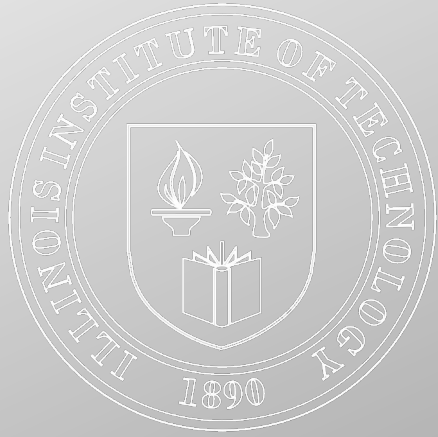
Figure 3: Iterative Development Model

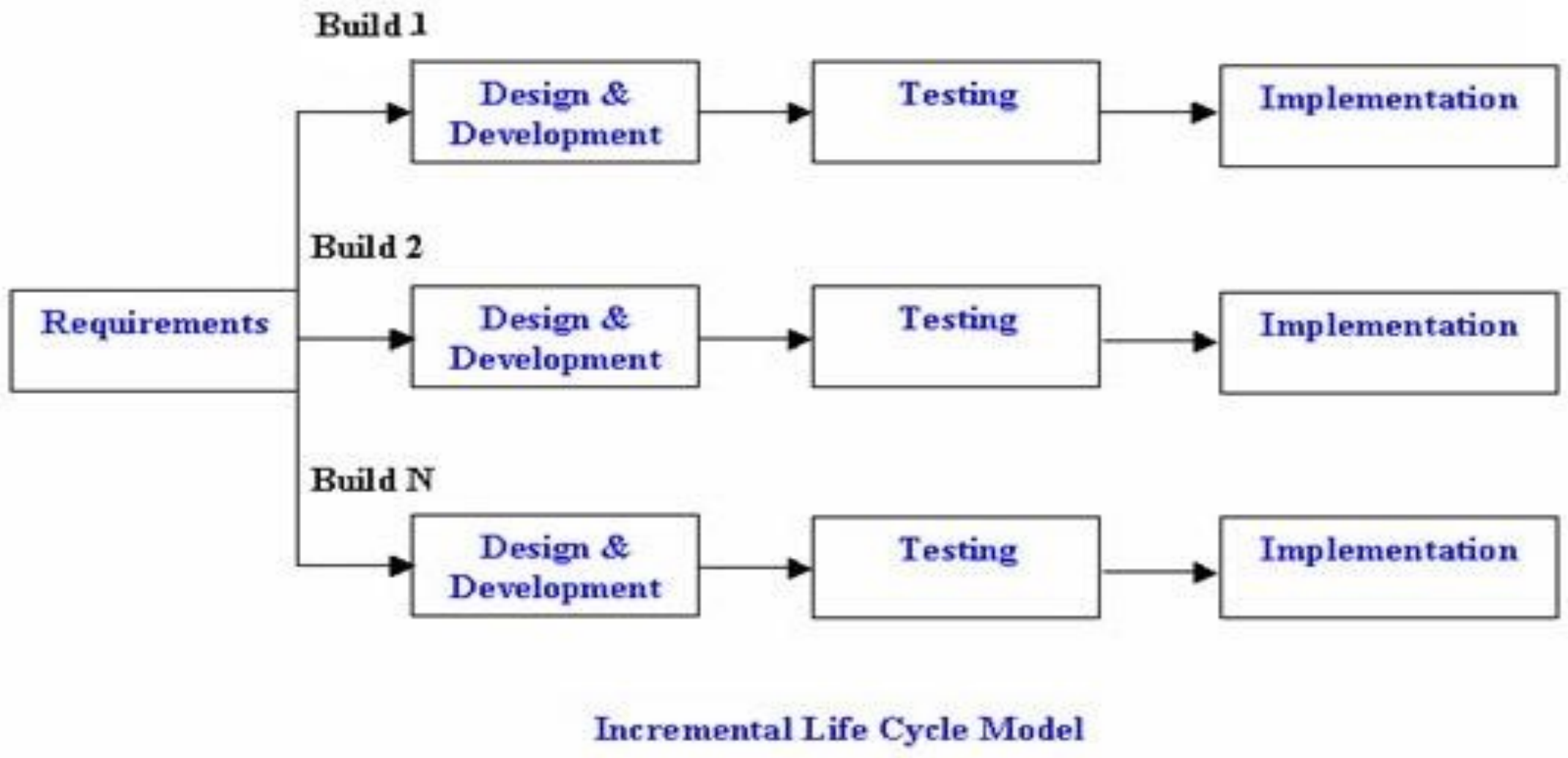
Incremental Development Model

- A development lifecycle where a project is broken into a series of incensements, each of which delivers a portion of the functionality in the overall project requirements.
- Requirements are prioritized and delivered in priority order in the appropriate increments.

Incremental Testing

- Testing where components or systems are integrated and tested one or some at a time, until all the components or systems are integrated and tested.

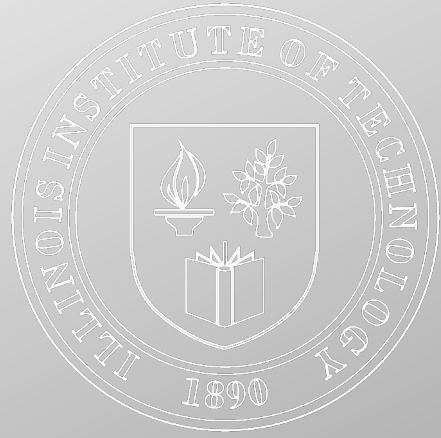




Incremental Life Cycle Model

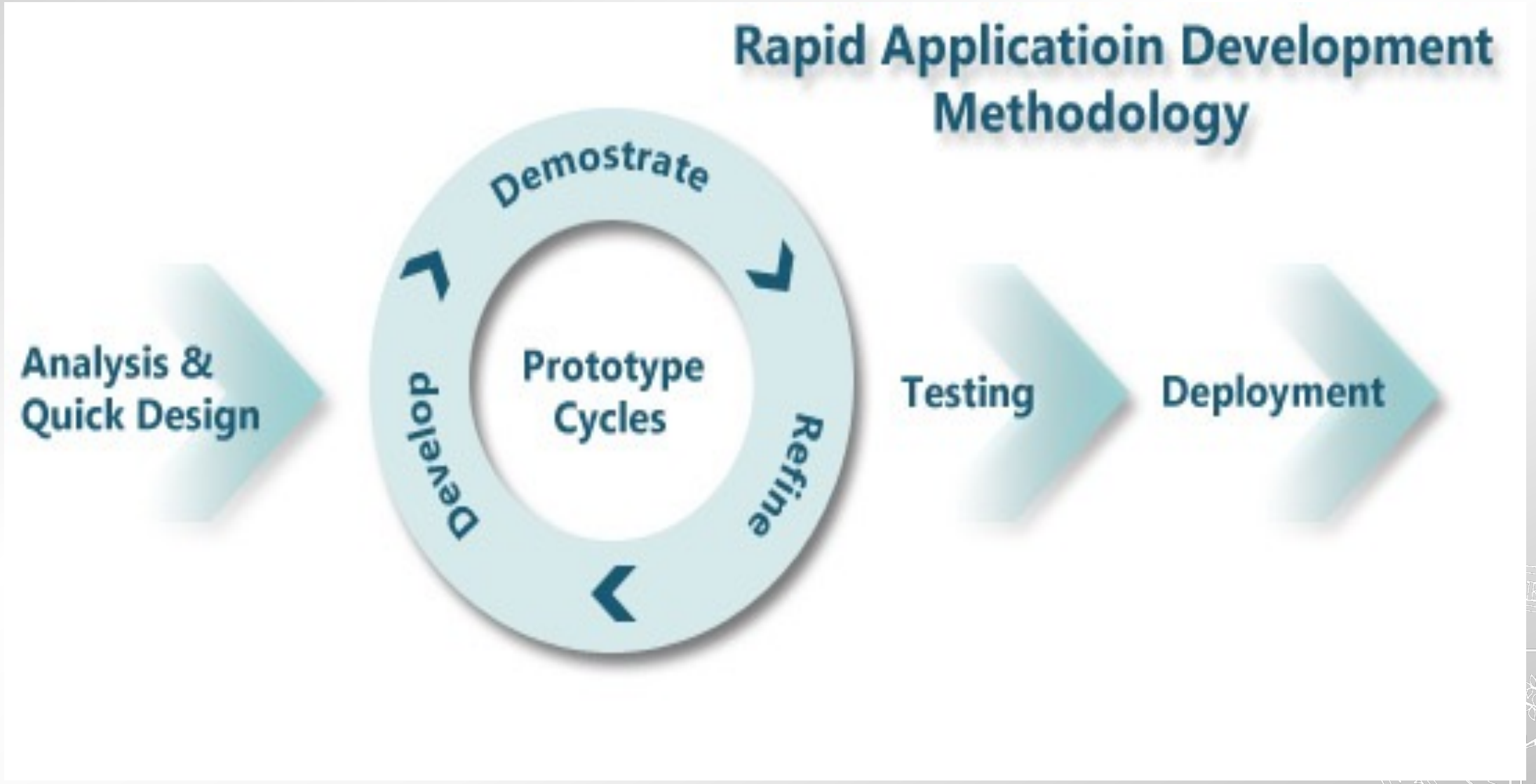
Rapid Application Development-RAD

- Rapid Application Development is formally a parallel development of functions and subsequent integration.
- This methodology allows early validation of technology risks and a rapid response to changing customer requirements.



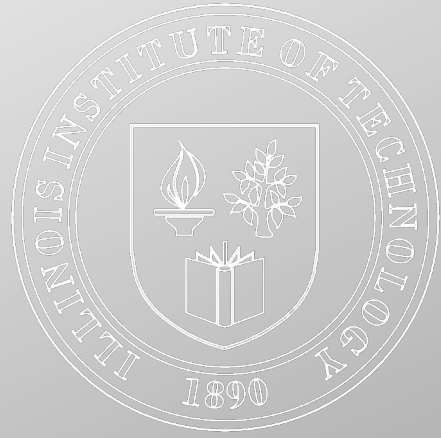
Rapid Application Development - RAD

- RAD provides early visibility of the product to the customer.
- Customer can provide feedback on the design and can decide, based on the existing functionality, whether to proceed with the development, what functionality to include in the next delivery cycle or stop the project.


Rapid Applicatioin Development Methodology
Analysis & Quick Design
Demostrate
Refine
Develop
Prototype Cycles
Testing
Deployment

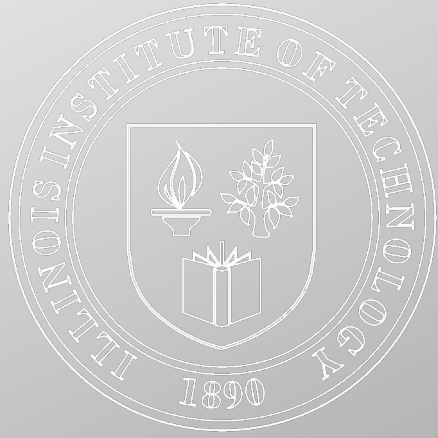
Agile Software Development

- A group of software development methodologies based on iterative incremental development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.



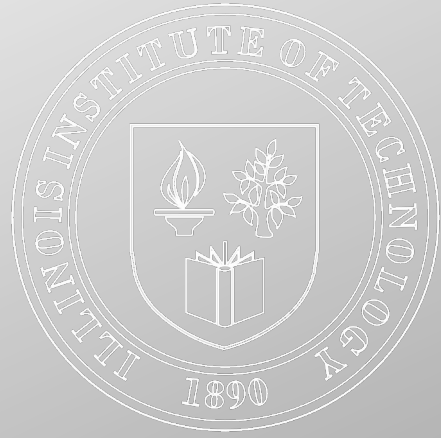
Agile Teams

- Most agile teams use Scrum, a management framework for iterative incremental development projects.
- Agile teams consist of 5-9 people



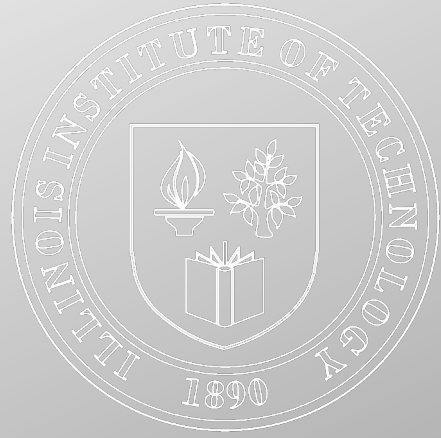
Agile Manifesto a Statement of Value

- Individuals & Interactions
- Working Software
- Customer collaboration
- Responding to Change
- Process & Tools
- Comprehensive documentation
- Contract negotiations
- Following a Plan



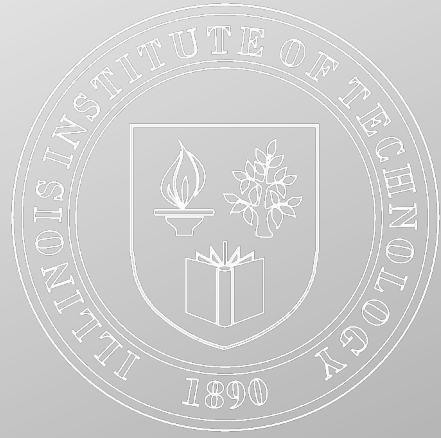
Test Driven Development

- A way of developing software where the test cases are developed, and often automated, before the software is developed to run those test cases.



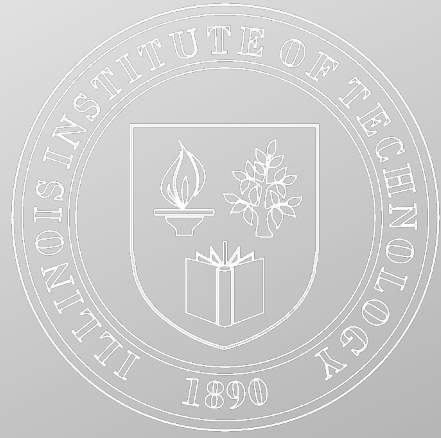
Agile Testing

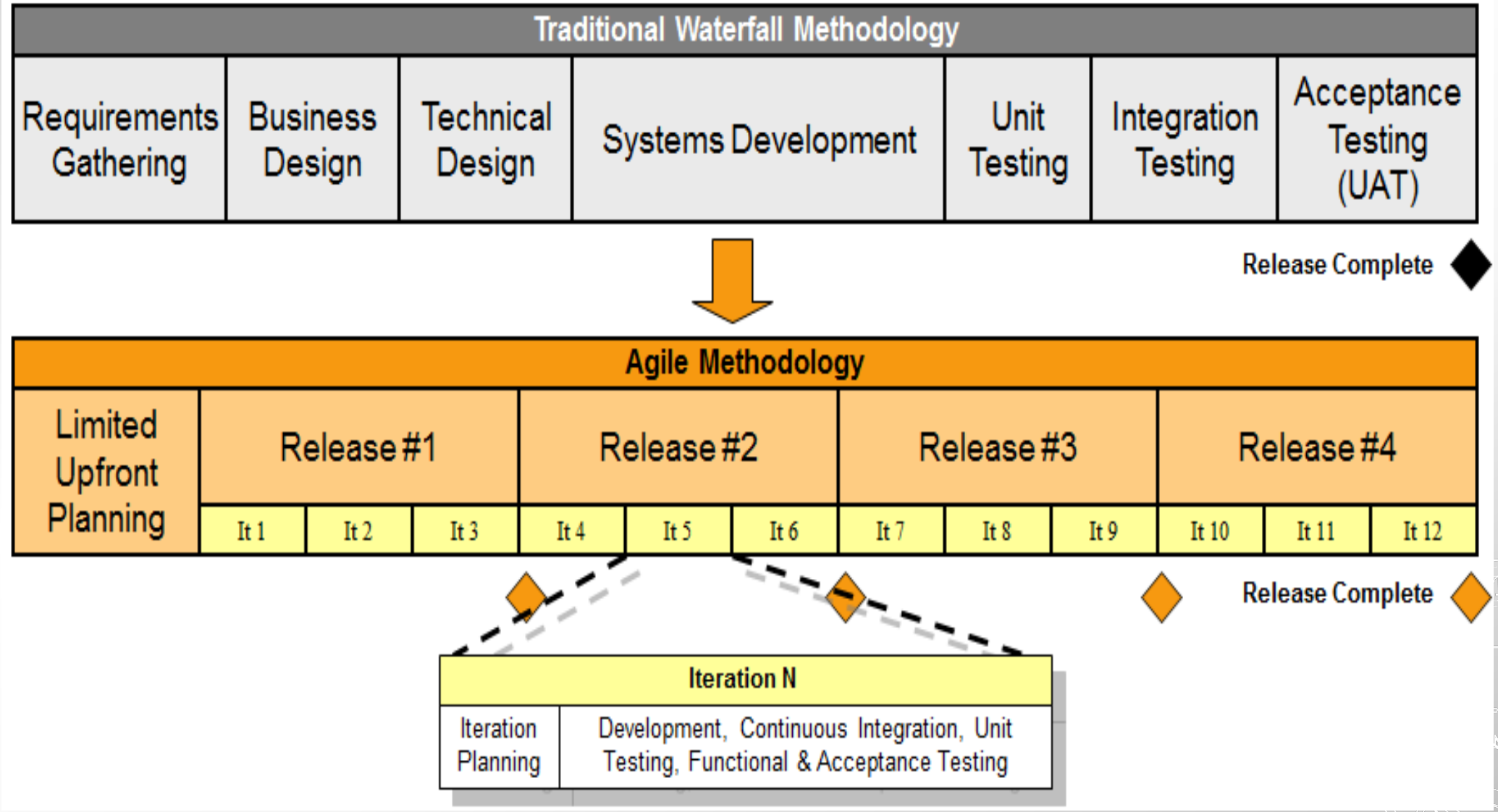
- Testing practice for a project using agile methodologies, such as extreme programming (XP), treating development as the customer of testing and emphasizing the test-first design paradigm.



Benefits of Agile Testing

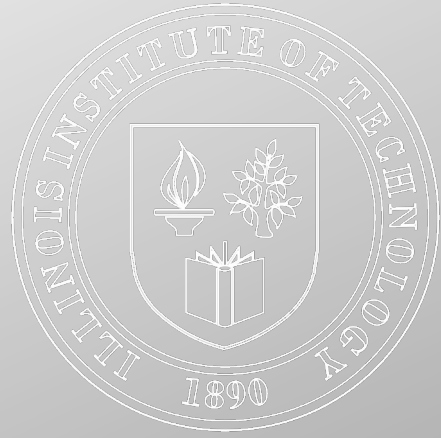
- Focus on working software and good quality
- Test driven development
- Accessibility of business stakeholders to help testers
- Self-organizing teams
- Simplicity of design that should be easier to test

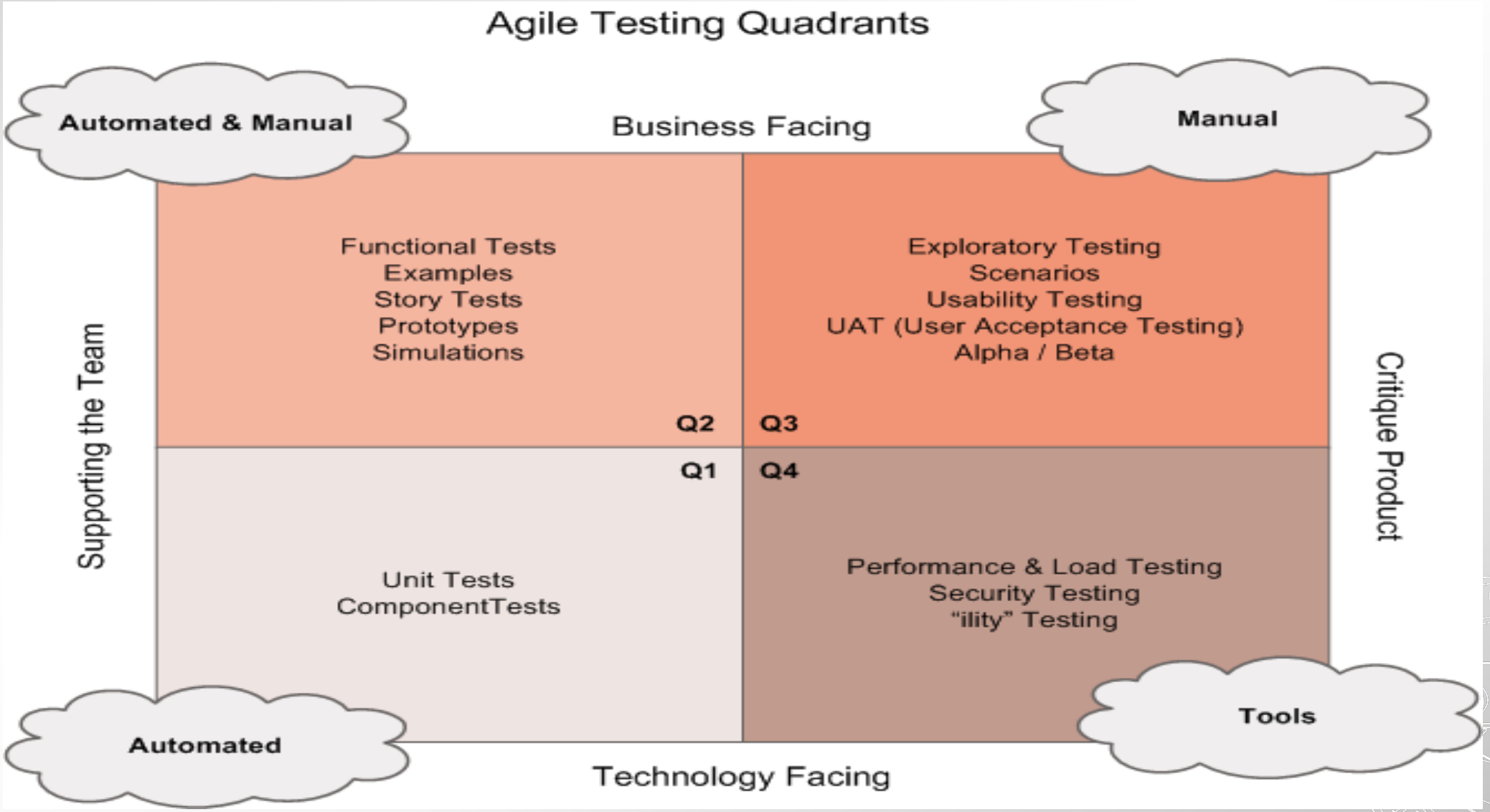



Traditional Waterfall Methodology
Requirements Gathering
Business Design
Technical Design
Systems Development
Unit Testing
Integration Testing
Acceptance Testing (UAT)
Release Complete
Agile Methodology
Limited Upfront Planning
Release #1
Release #2
Release #3
Release #4
It 1
It 2
It 3
It 4
It 5
It 6
It 7
It 8
It 9
It 10
It 11
It 12
Release Complete
Iteration N
Iteration Planning
Development, Continuous Integration, Unit Testing, Functional & Acceptance Testing

Test driven development

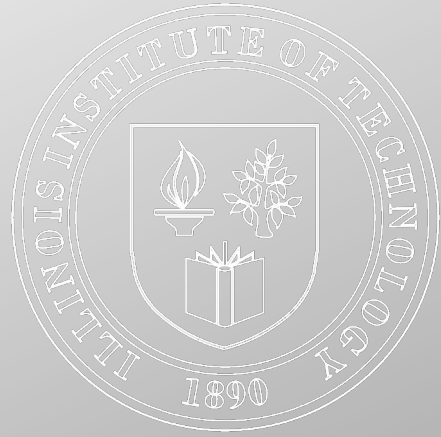
- Test driven development is a way of developing software where the test cases are developed, and often automated, before the software is developed to run those test cases.

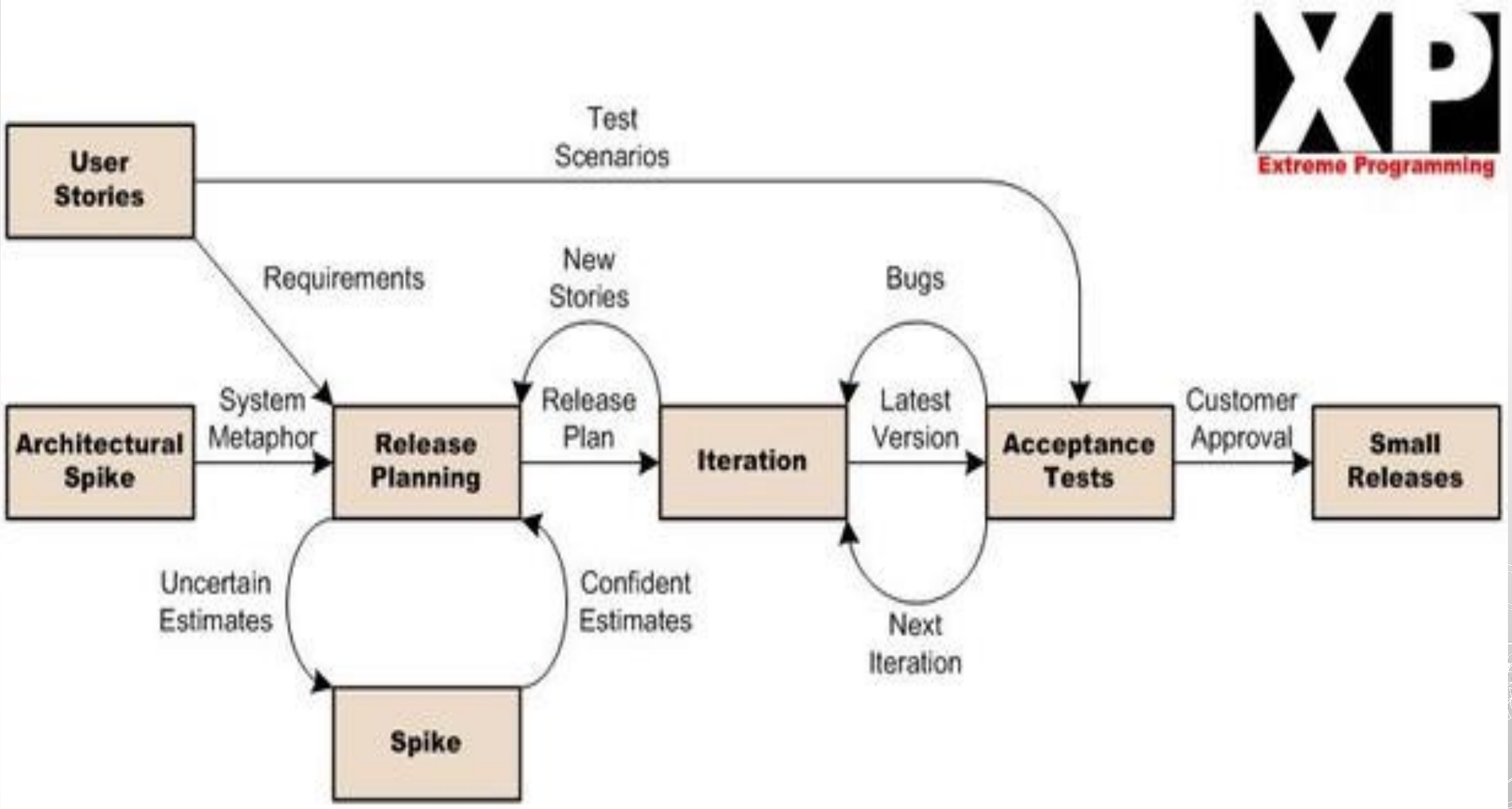



Agile Testing Quadrants
Automated & Manual
Business Facing
Manual
Functional Tests
Examples
Story Tests
Prototypes
Simulations
Exploratory Testing
Scenarios
Usability Testing
UAT (User Acceptance Testing)
Alpha / Beta
Supporting the Team
Critique Product
Q2
Q3
Q1
Q4
Unit Tests
ComponentTests
Performance & Load Testing
Security Testing
"ility" Testing
Automated
Tools
Technology Facing

Extreme Programming (XP)

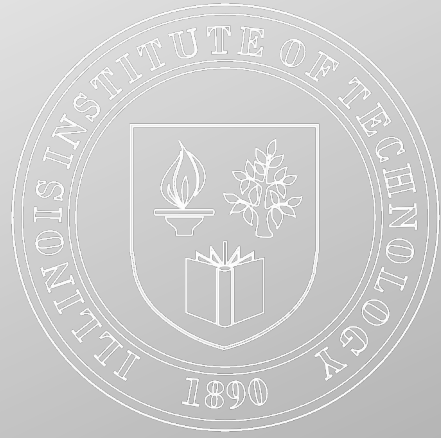
- A software engineering methodology used within agile software development whereby core practices are programming in pairs, doing extensive code review, unit testing of all code, and simplicity and clarity in code.




XP
Extreme Programming
User Stories
Test Scenarios
Requirements
New Stories
Bugs
Architectural Spike
System Metaphor
Release Planning
Release Plan
Iteration
Latest Version
Acceptance Tests
Customer Approval
Small Releases
Uncertain Estimates
Confident Estimates
Next Iteration
Spike

SCRUM

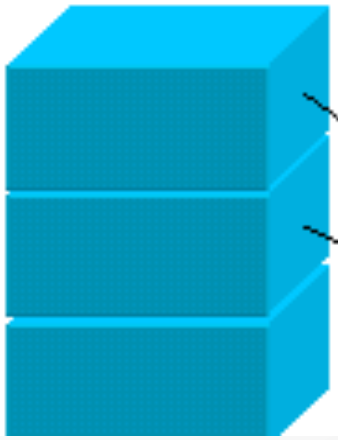
- Scrum is an iterative incremental framework for managing projects commonly used with agile software development.
- Changes are incorporated continuously into the software build.



SCRUM Sprint Cycle

Product Backlog:
Prioritized list of features required by the customer

Sprint Backlog:
Features to be done this sprint
Features are expanded into smaller tasks.



Workday:
One Day

Sprint:
One Month

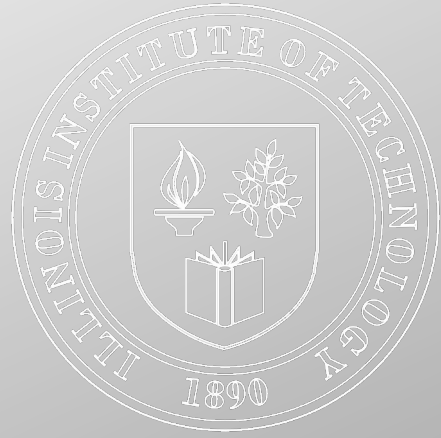
Every Day, a 15-minute meeting is held, and the SCRUM Master asks the 3 questions:

1) What have you accomplished since the last meeting?
2) Are there any obstacles in the way of meeting your goal?
3) What will you accomplish before the next meeting?

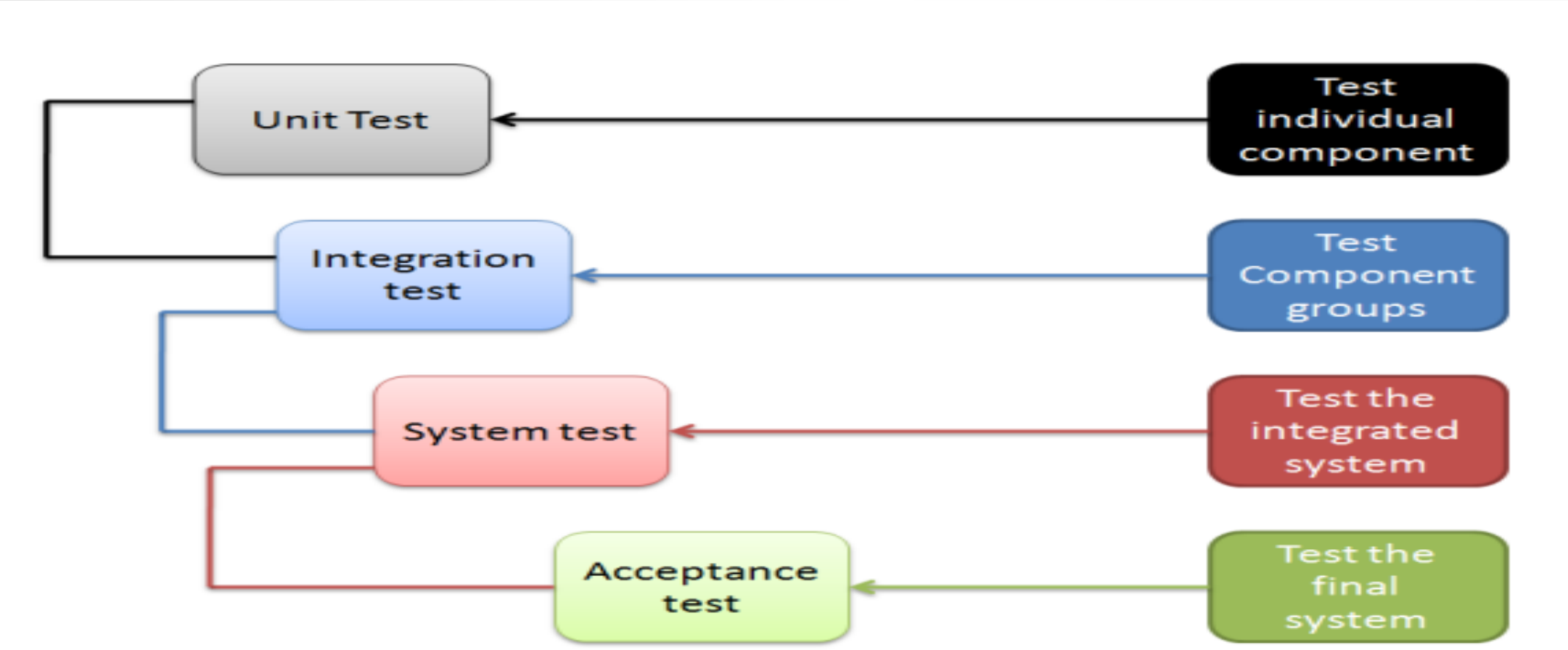
New Functionality
is demonstrated at the end of each sprint.

2.2 Test Levels

- A group of test activities are organized and managed together.
- A test level is linked to the responsibilities in a project.
- Component Test,
- Integration Test
- System Test and
- Acceptance Test

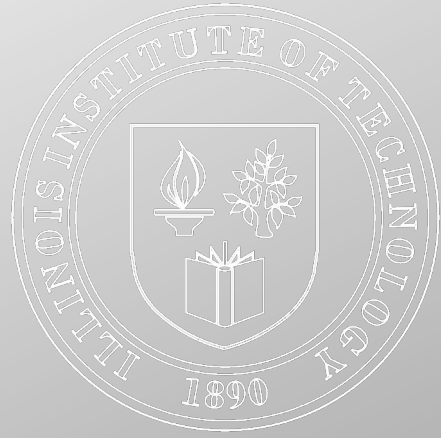


Test Levels


Unit Test
Test individual component
Integration test
Test Component groups
System test
Test the integrated system
Acceptance test
Test the final system

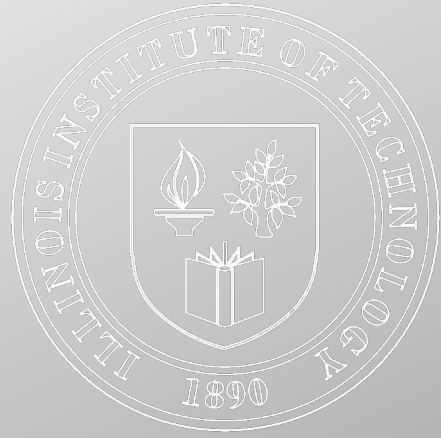
Testing Best Practices

- Measurable effectiveness & efficiency
- The test basis
- The item build – test object
- The typical defects and failures
- Applicable requirements for test harnesses
- Approaches we intend to use



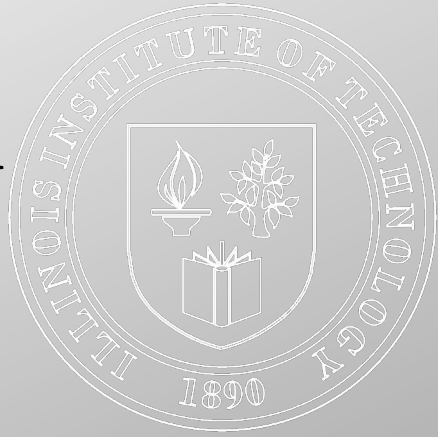
Efficiency and Efficiency Testing

- The capability of the software product to provide appropriate performance, relative to the amount of resources used under state conditions. (ISO)
- The process of testing to determine the efficiency of a software product.



Component Testing (Unit/Module)

- A single procedure
- A function
- A body of code that implements a single function
- Source code that fits on one page
- Code done in 4-40 hours – 300 lines
- The smallest body of code that can be compiled and executed by itself

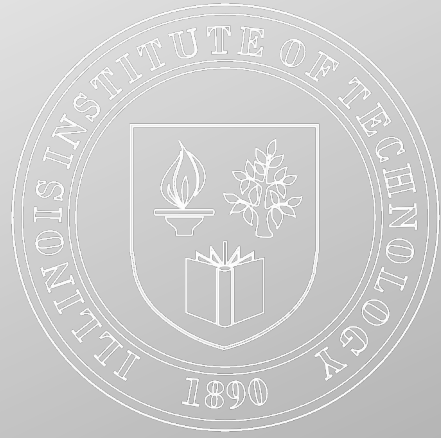


Stubs and Drivers

- Component testing may be done in isolation from the rest of the system depending on the context of the development life cycle and the system.
- Stubs and drivers are used to replace the missing software and simulates interface between software components

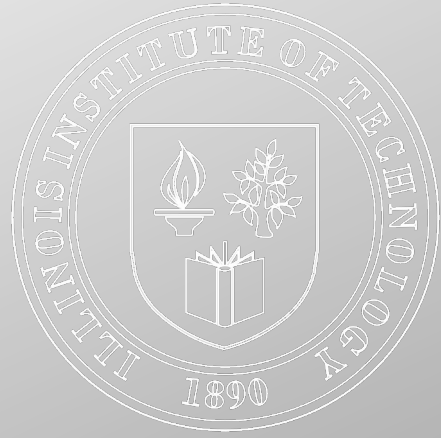
Stub

- A skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a called component.

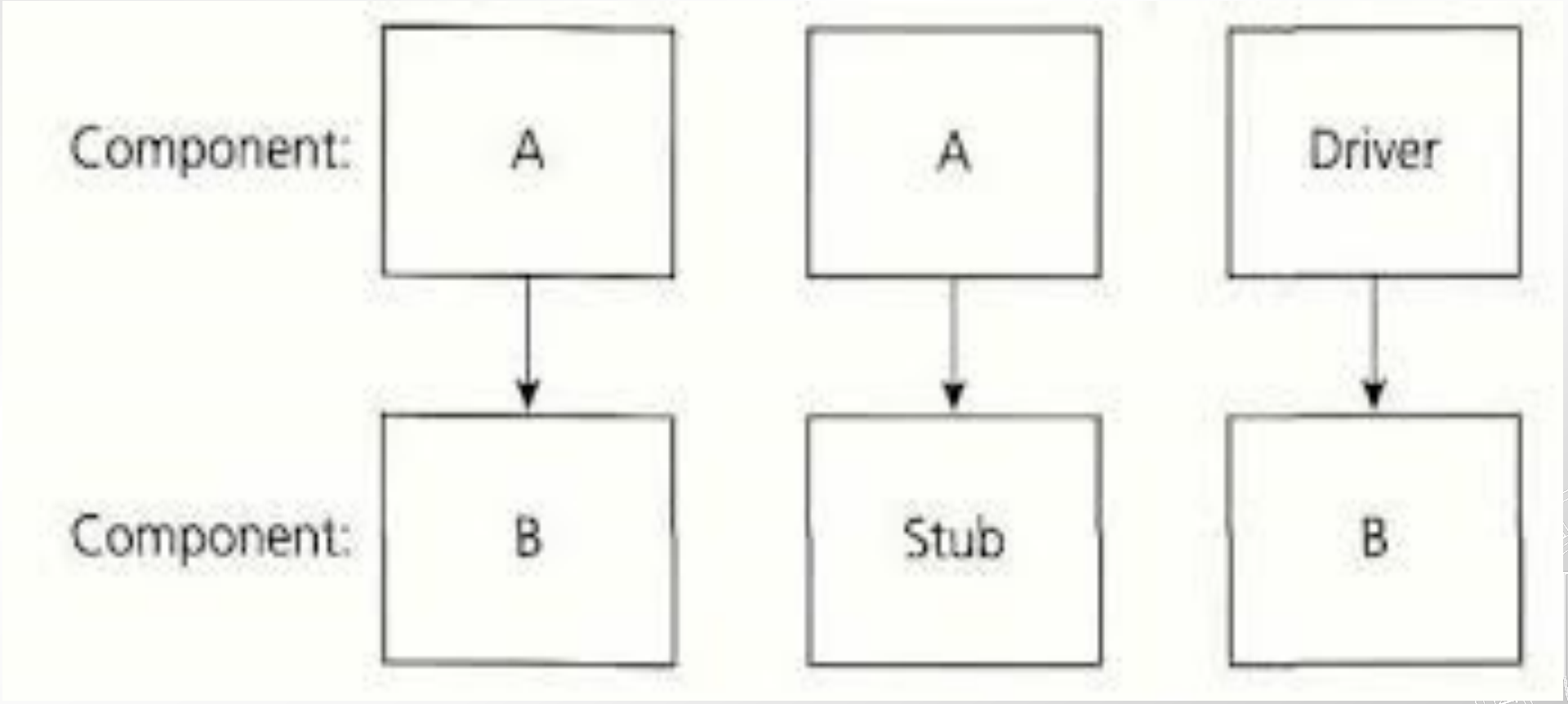


Driver (Test Driver)

- A software component or test tool that replaces a component that takes care of the control and/or the calling of a component or system.

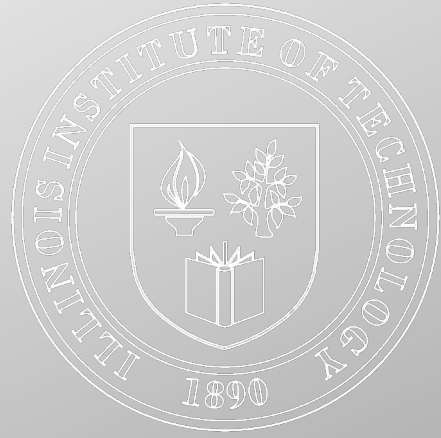


Stubs and Drivers



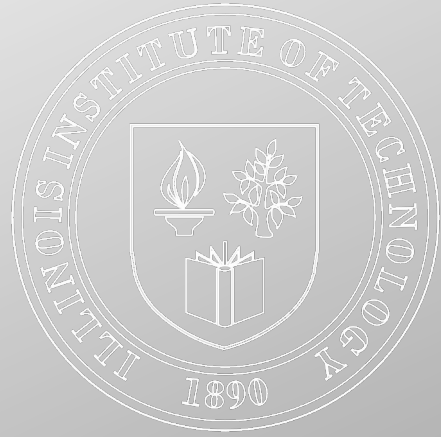
Robustness Testing

- Robustness: The degree to which a component or system can function correctly in the presence of invalid inputs or stressful environmental conditions
- Testing to determine the robustness of the software product
- Error or fault tolerance (ISO)



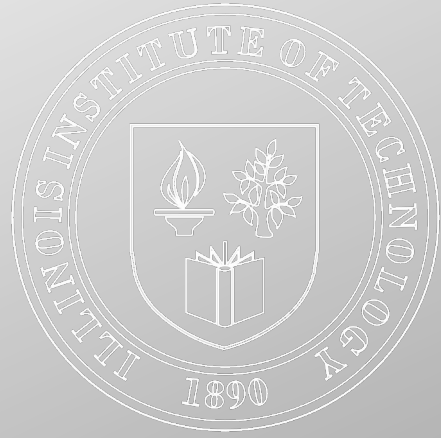
Component Testing

- Include testing of functionality and specific non-functional characteristics such as resource-behavior (e.g. memory leaks), performance or robustness testing, as well as structural testing (e.g. decision coverage)



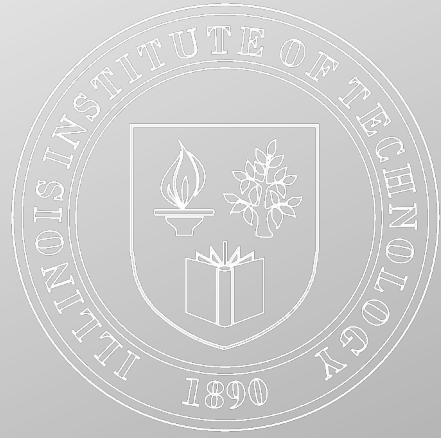
Test Driven Development

- A way of developing software where the test cases are developed and often automated, before the software is developed to run those test case
- Extreme programming (XP), prepares and automates test cases before coding

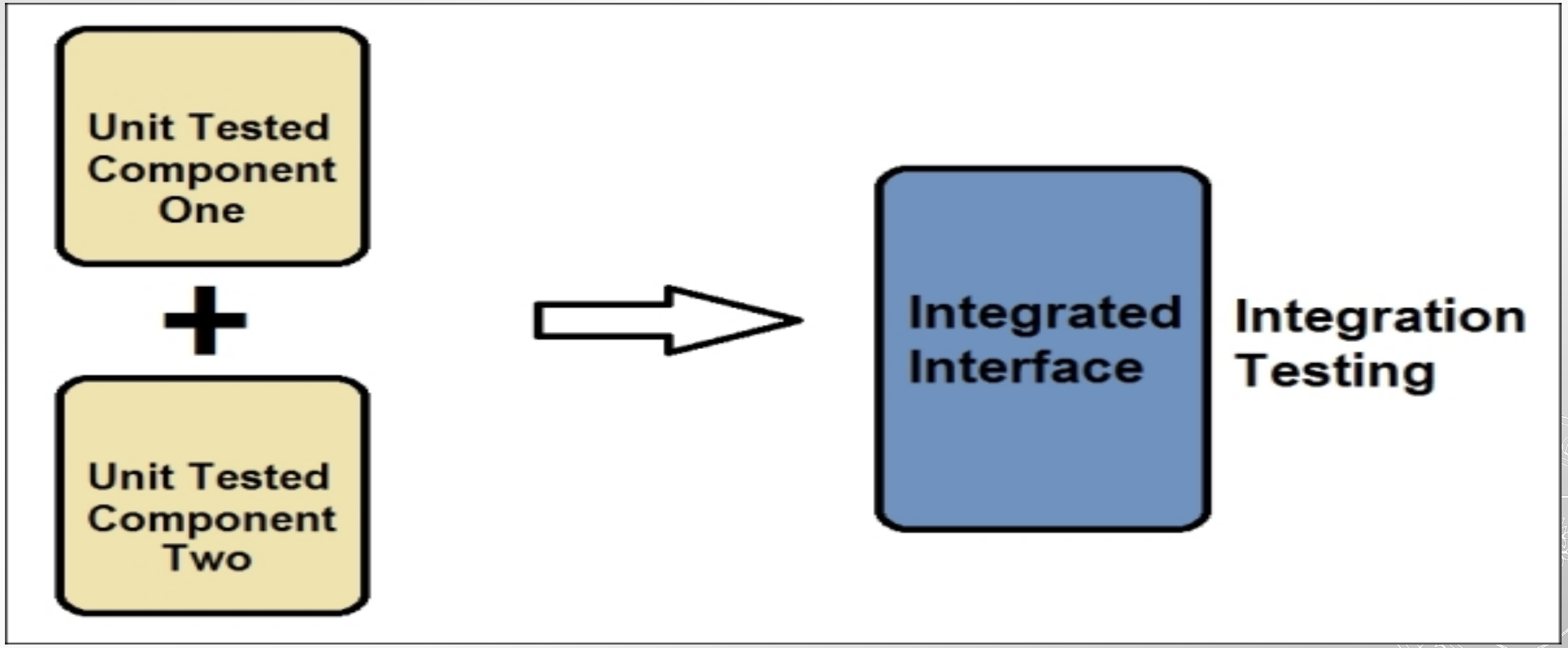


Integration Testing

- The process of combining components or systems into larger assemblies
- Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems

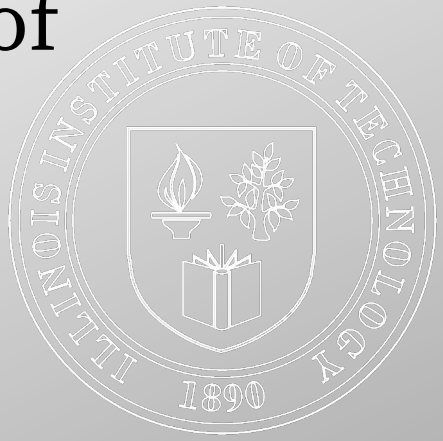


Component to Integration Testing



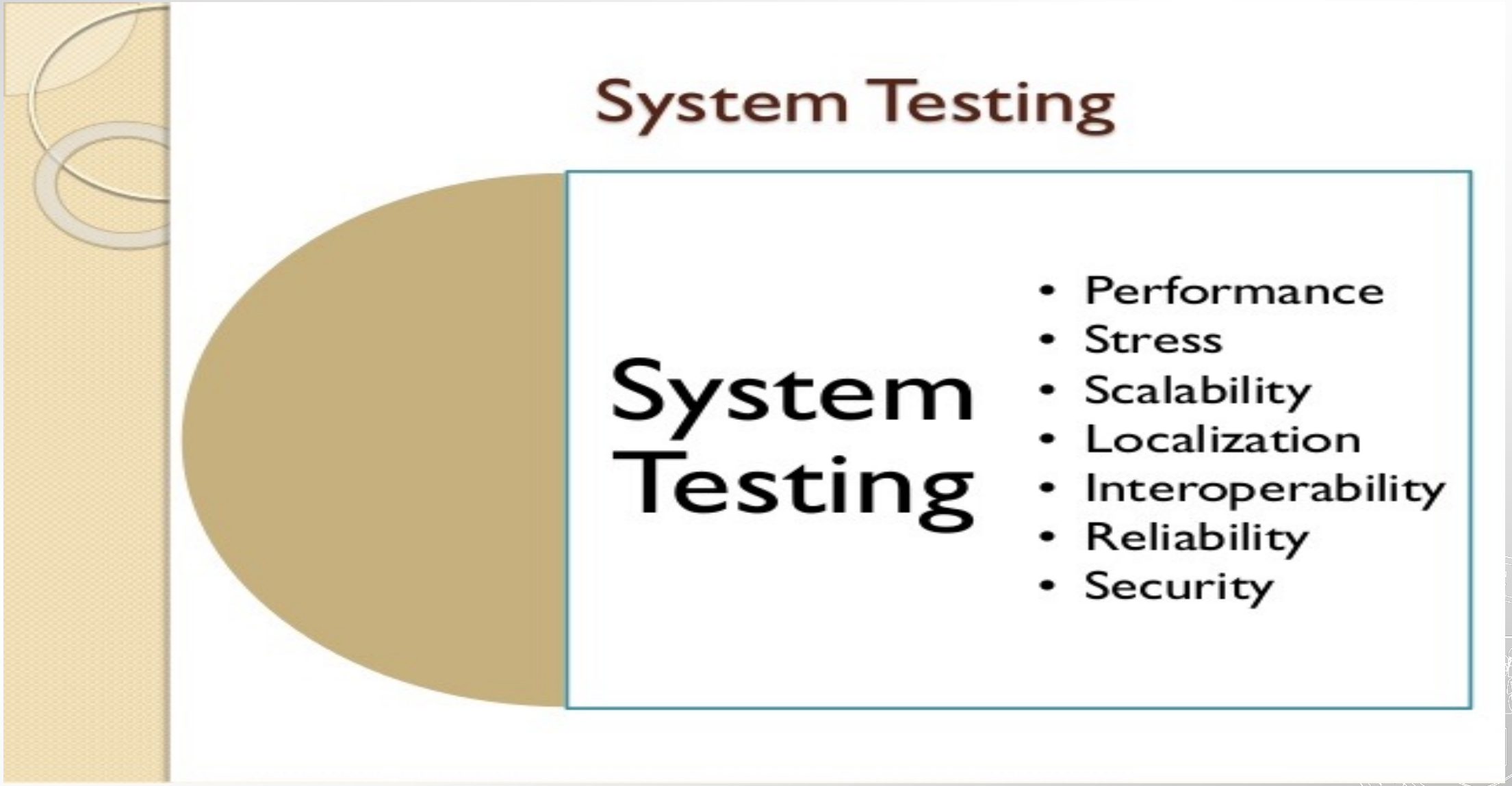
System Integration Testing

- Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet)
- Integration testing may be carried out by the developers, but can be done by a separate team of specialist integration testers



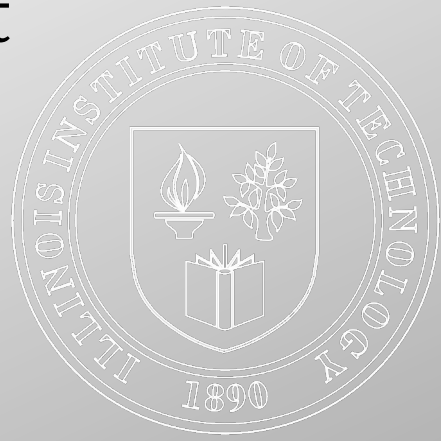
System Testing

- The process of testing an integrated system to verify that it meets specified requirements
- Includes tests based on risk analysis reports, system, functional, or software requirements specifications, business processes, use cases, and interactions with the operating system and system resources
- System testing is most often the final test on behalf of development to verify that the system to be delivered meets the specification and its purpose may be to find as many defects as possible.


System Testing
System Testing
Performance
Stress
Scalability
Localization
Interoperability
Reliability
Security

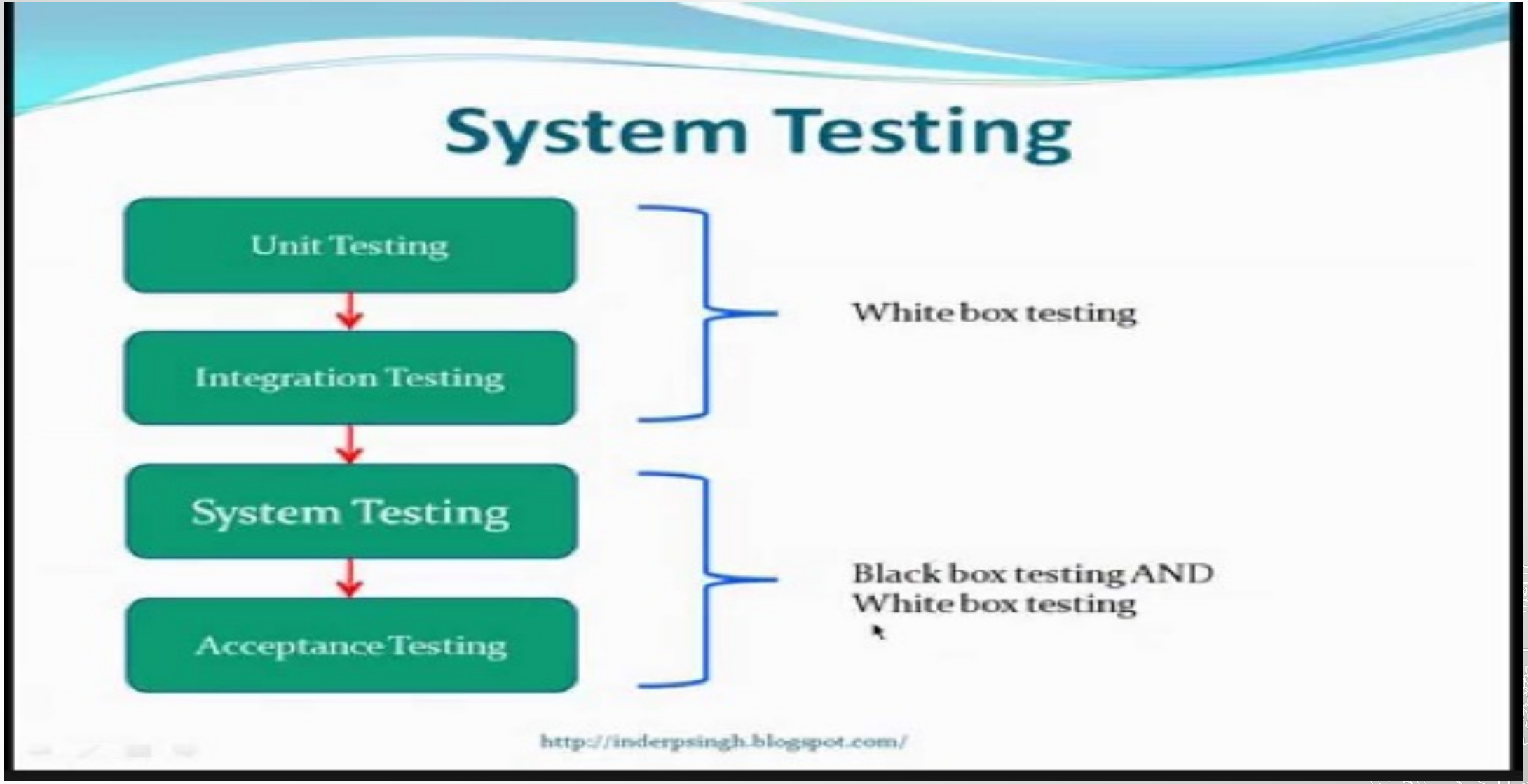
System Testing

- System testing should investigate both functional and non-functional requirements of the system.
- **Functional Requirement** – A requirement that specifies a function that a component or system must perform
- **Non-Functional Requirement** – A requirement that attributes to reliability, efficiency, usability, maintainability and probability



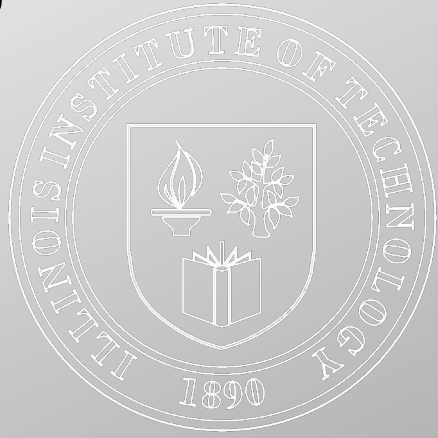
System Testing

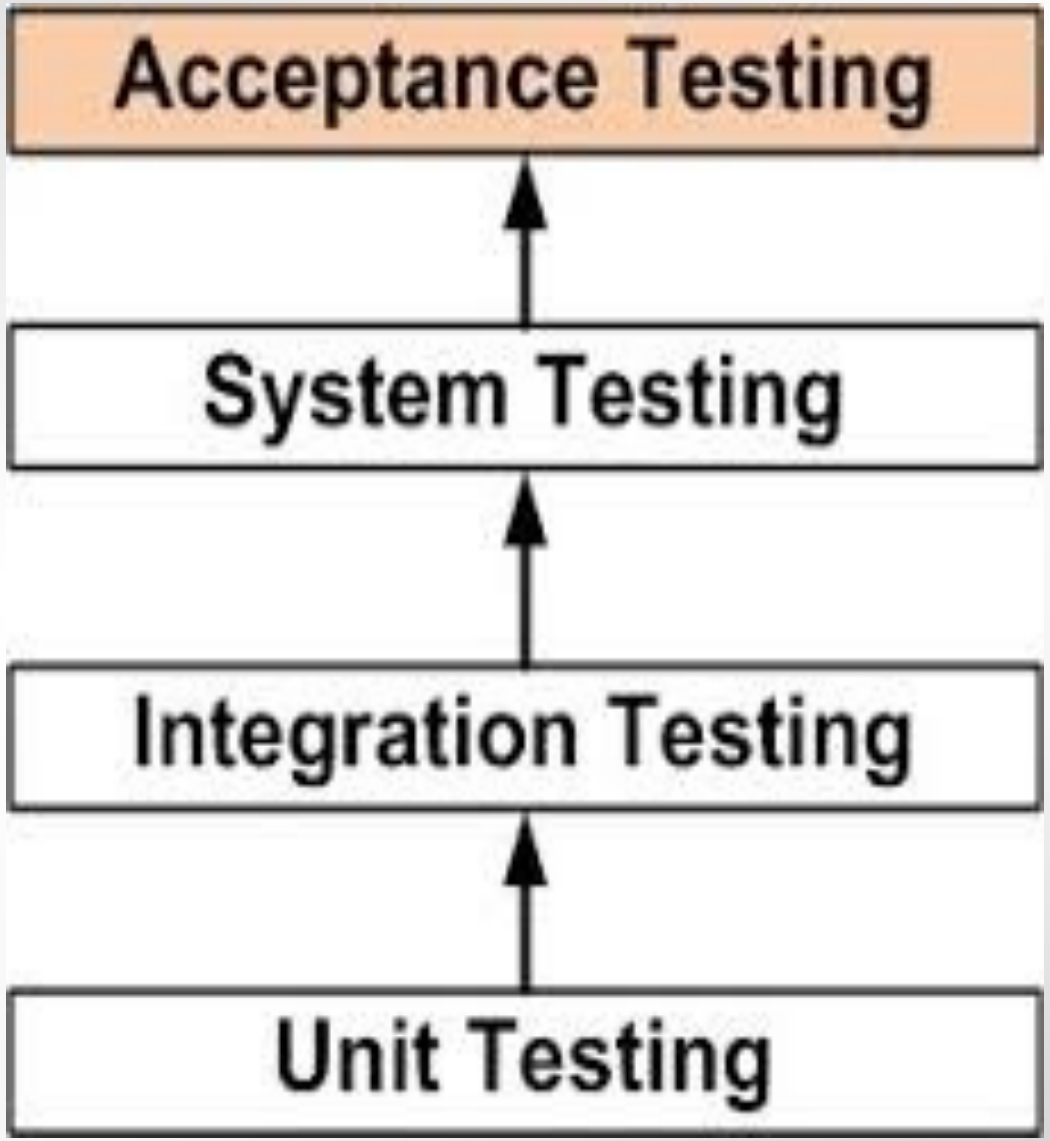
- System testing requires a controlled **test environment** including controlled software versions, testware and the test data
- The test environment should be similar to the production environment in order to minimize the risk of environment-specific failures


System Testing
Unit Testing
Integration Testing
System Testing
Acceptance Testing
White box testing
Black box testing AND
White box testing
http://inderpsingh.blogspot.com/

Acceptance Testing

- Formal testing with respect to user need, requirements and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system






Acceptance Tests
NO
YES

Alpha Testing

- Simulated or actual operational testing by potential users/customers or an independent test team at the developers' site, but outside the development organization. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing.


ALPHA TESTING
WHERE IT ALL BEGINS

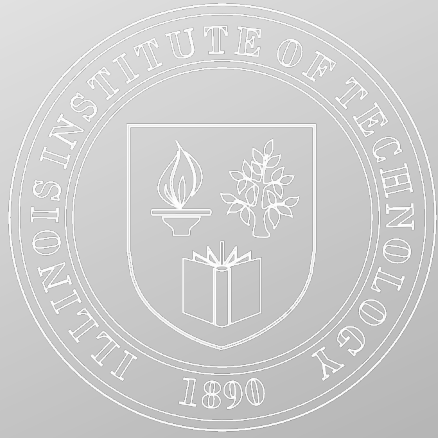
Beta Testing

- Operational testing by potential and/or existing users/customers at an external site not otherwise involved with the developers, to determine whether or not a component or system satisfies the user/customer needs and fits within the business processes. It is an external form of testing to acquire the feedback from the market

Beta Testing (Field Testing)

Test Types

- A group of test activities aimed at testing a component or system focused on a specified test objective, i.e. functional test, usability test, regression test etc. A test type may take place on one or more test levels or test plans.



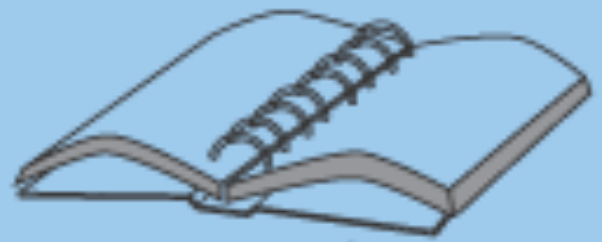
Functional Testing

- Testing based on analysis of the specifications of the functionality of a component or system
- Functional testing is also referred to as black-box testing (specification based testing)
- The process of testing to determine the functionality of a software product (ISO)

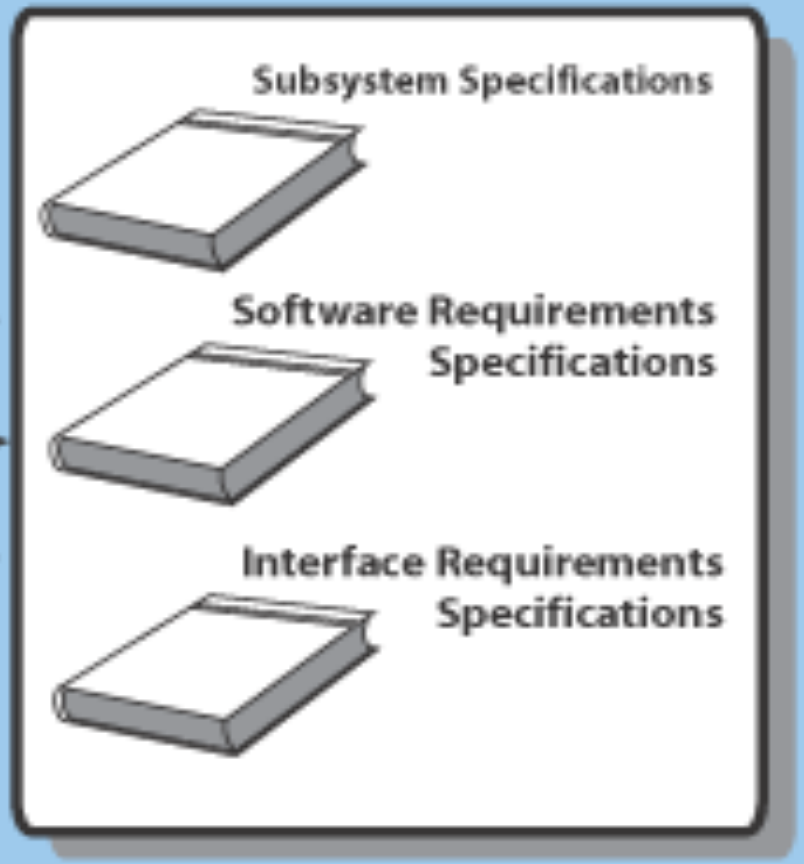
Function Test

Objectives

- Demonstrate that all software requirements are met
- Demonstrate that security safeguards are met



System Requirements Specifications



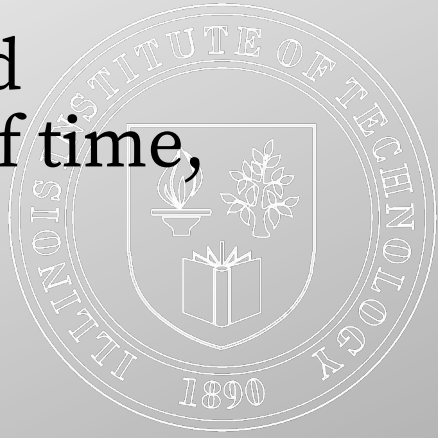
Functional Testing

- Focused on suitability, interoperability testing, security, accuracy and compliance.
- Interoperability testing determines the interoperability of a software product
- **Security**: Attributes of software products that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data
- **Security Testing**: Testing to determine the security of the software product

Security Testing

Non-Functional Testing

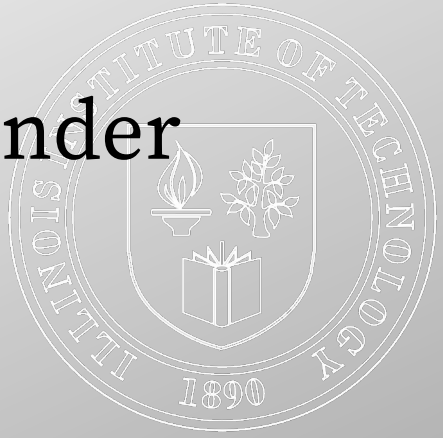
- Non-functional testing is performed at all levels
- Testing the attributes of a component or system that do not relate to functionality, e.g. reliability, efficiency, usability, maintainability and portability
- **Reliability Testing**: The process of testing determine the reliability of a software product
- The ability of the software product to perform its required functions under stated conditions for a specified period of time, or for a specified number of operations (robustness)




Reliability
/re-ly-a-bi-li-ti/
The ability of the software
to function under the given
conditions

Non-Functionality Testing

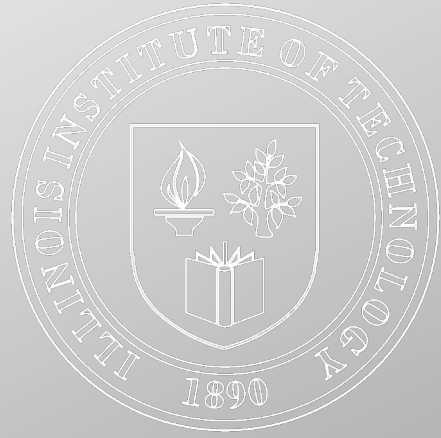
- **Usability Testing:** Testing to determine the extent to which the software product is understood, easy to learn, easy to operate and attractive to the users under specified conditions
- The capability of the software to be understood, learned, used and attractive to user when used under specified conditions – easy to use




Inconsistent
Consistent
Organised
Unrefined
Understandable
Simplistic
Misleading
Counter intuitive
Useful
Hard to Use
Unattractive
Fresh
Controllable
Effective
Engaging
Ordinary
Contradictory
Powerful
Clear
Straightforward
Clean
Appealing
Desirable
Distracting
Satisfying
Business like
Comprehensive
Entertaining
Ambiguous
Accessible
Credible
Motivating
Time-saving
Stimulating
Non standard
Overwhelming
Vague
Usable
Efficient
Confusing
Stable
Easy to use
Innovative
Effortless
Impressive
Creative
High quality
Professional
Flexible
Unconventional
Familiar
Approachable
Relevant
Annoying
Sophisticated
Advanced
Expected
Irrelevant
Responsive
Convenient
Meaningful
Friendly
Intuitive
Attractive
Trustworthy

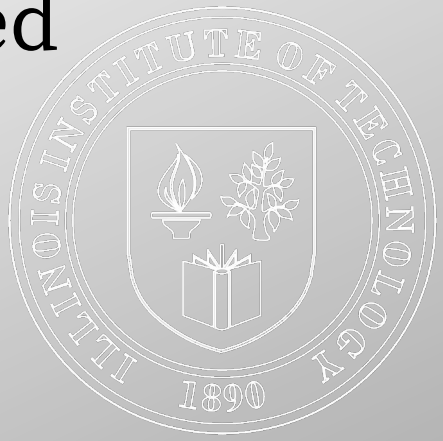
Non-Functional Testing

- **Efficiency Testing**: The process of testing to determine the efficiency of a software product
- The capability of the software product to provide appropriate performance, relative to the amount of resources used under stated conditions



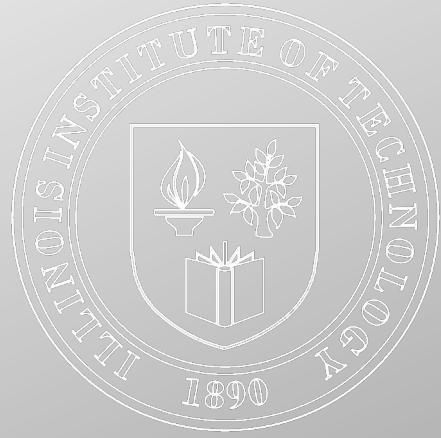
Non-Functional Testing

- **Maintainability Testing**: The process of testing to determine the maintainability of the software product
- The ease with which a software product can be modified to meet new requirements, modified to make future maintenance easier, or adapted to changed environment



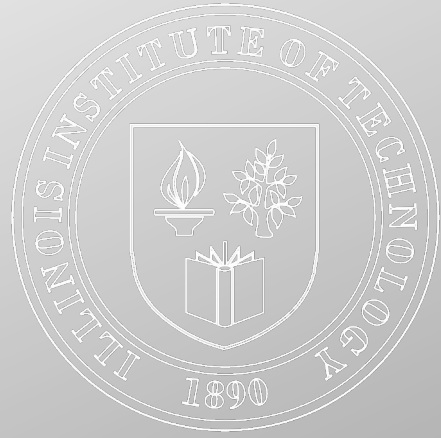
Non-Functional Testing

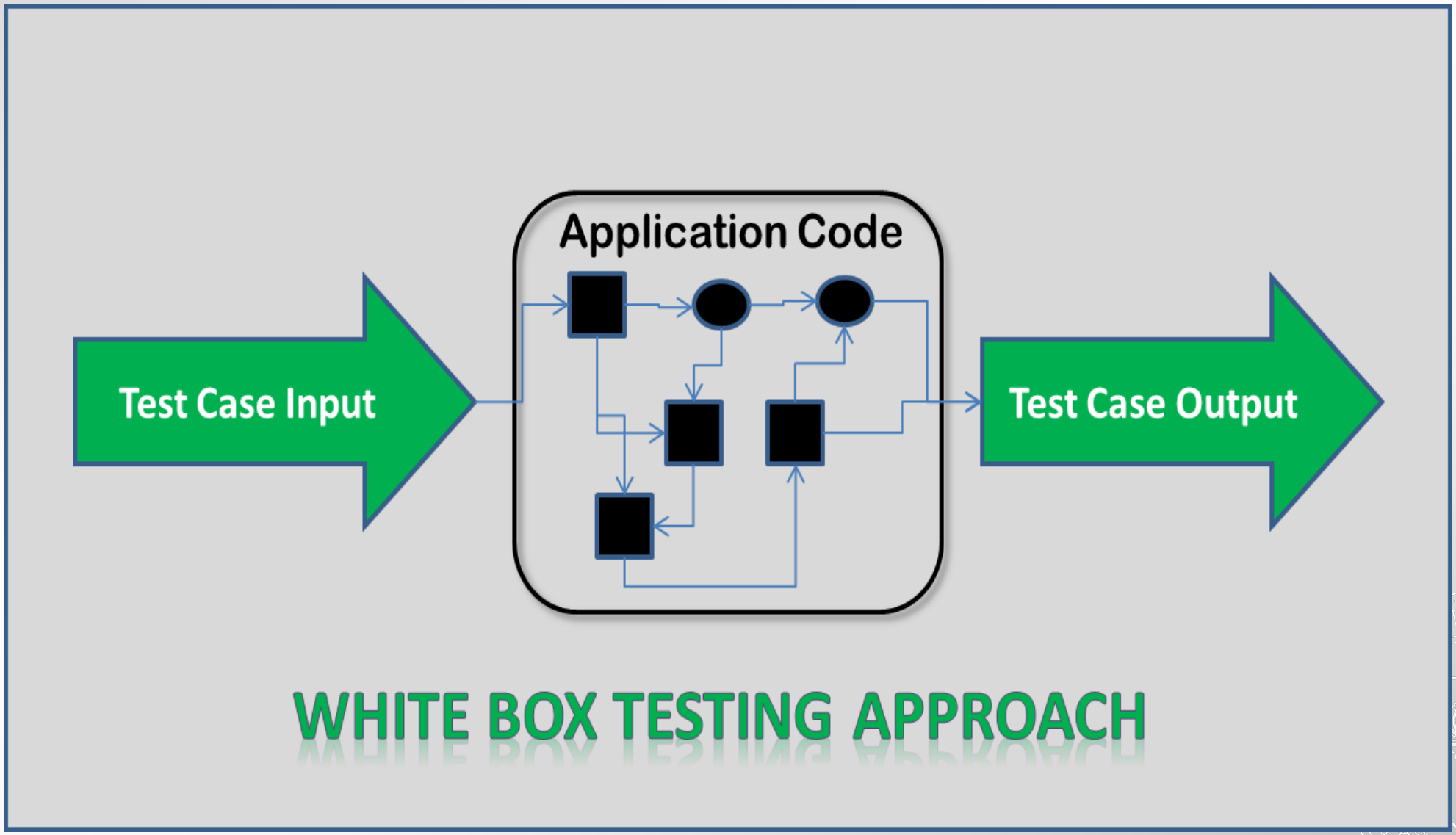
- **Portability Testing**: The process of testing to determine the portability of a software product
- The ease with which the software product can be transferred form one hardware or software environment to another



Structural Testing (White-Box)

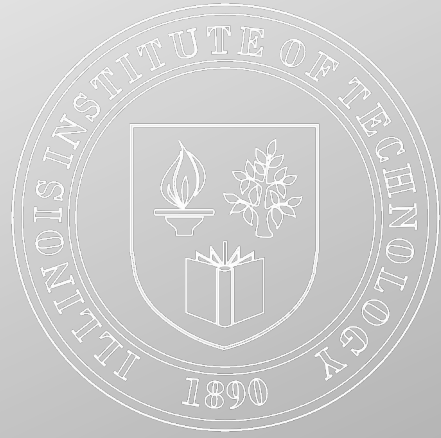
- Testing is based on an analysis of the internal structure of the component or system (code coverage)
- Procedures to derive and/or select test cases based on an analysis of the internal structure of a component or system



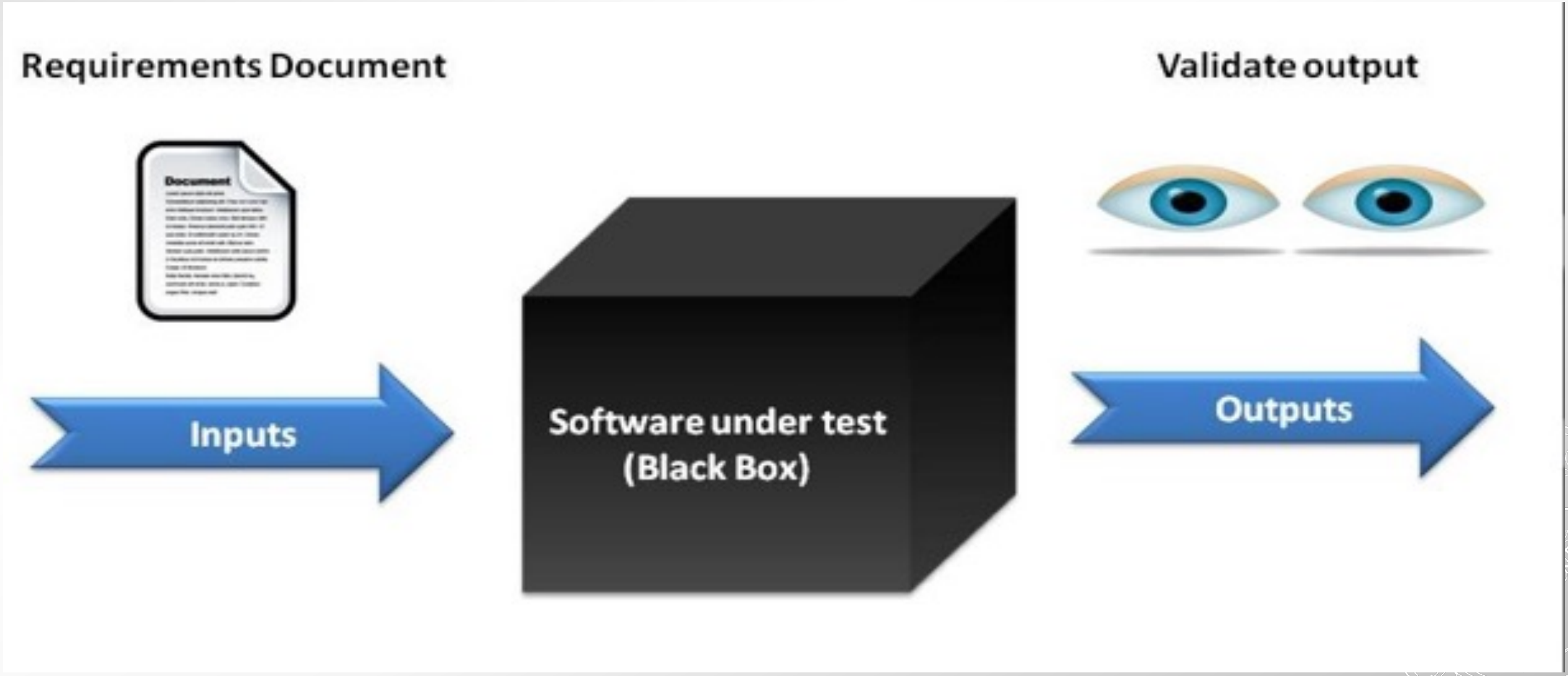

Application Code
Test Case Input
Test Case Output
WHITE BOX TESTING APPROACH

Black-Box Testing(specification based)

- Procedure to derive and/or select test cases based on an analysis of the specification, either functional or on-functional, of a component or system without reference to its internal structure
- Requirement based testing



Black Box Testing

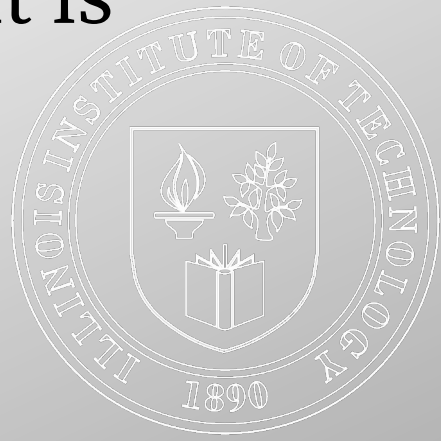


Confirmation Testing (Re-Testing)

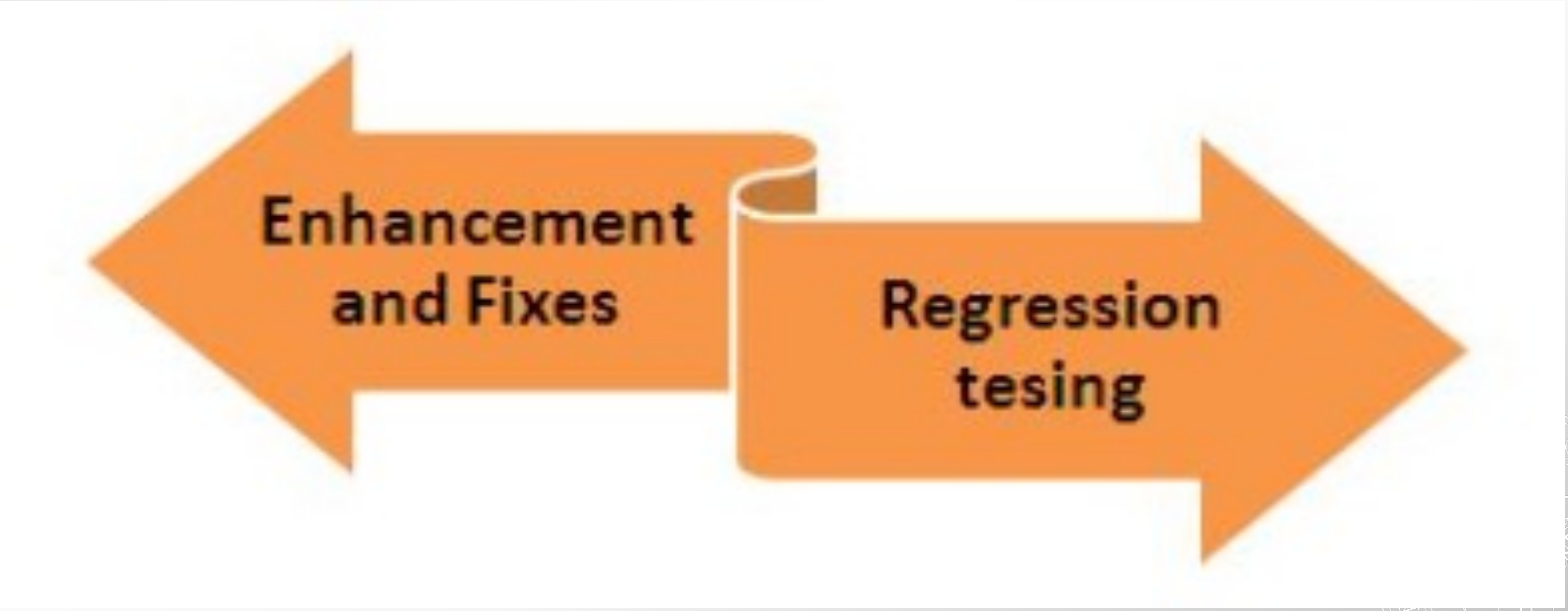
- Testing that runs test cases that failed the last tie they were run, in order to verify the success of corrective actions
- It's a verification of fixed issues
- It needs to be executed in the same manner as it was initially executed

Regression Testing

- Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed



Regression Testing

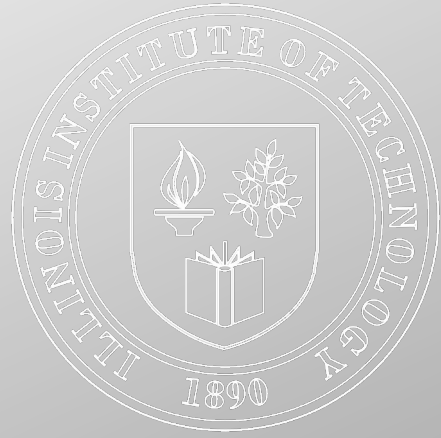


2.4 Maintenance Testing

- Testing the changes to an operational system or the impact of a changed environment to an operational system
- Modifications from planned enhancement changes a minor release includes bug fixes or urgent emergency changes

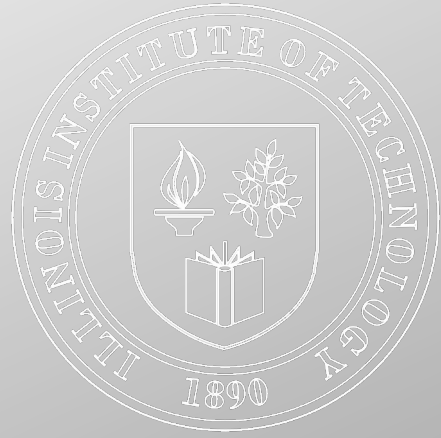
Impact Analysis & Regression Testing

- The assessment of change to the layers of development documentation, test documentation and components, in order to implement a given change to specified requirements



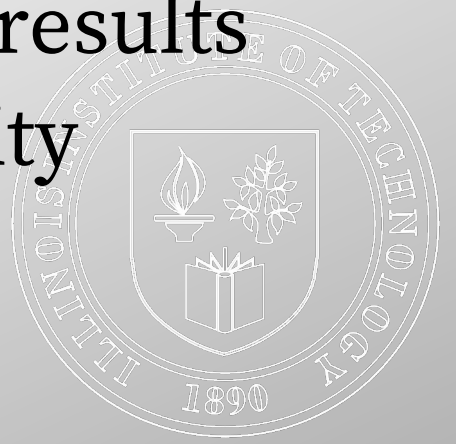
Triggers for Maintenance Testing

- **Planned modifications:**
- New function or enhance performance
- Adaptive modifications – environmental changes (new hardware, software, system or legislation)
- Corrections of defects



Ad-hoc Corrective Modifications

- Ad-hoc corrective modifications are required for immediate solution (production issue)
- Testing carried out informally; no formal test preparation takes place, no recognized test design technique is used, there are no expectations for results and arbitrariness guides the test execution activity



Sample Questions and Answers – Question 1

1. What are good practices for testing within the development life cycle?

a. Early test analysis and design.

b. Different test levels are defined with specified objectives.

c. Testers will start to get involved as soon as coding is done.

d. A and B above.

Question 2

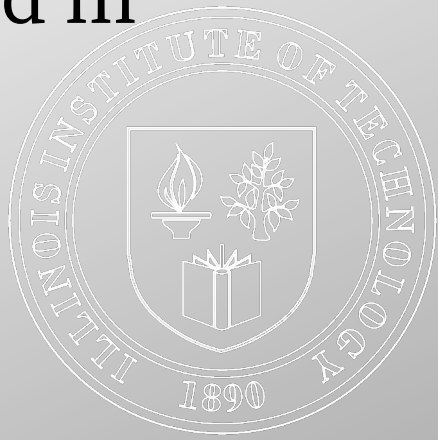
2.Which option best describes objectives for test levels with a life cycle model?

a. Objectives should be generic for any test level.

b. Objectives are the same for each test level.

c. The objectives of a test level don’t need to be defined in advance.

d. Each level has objectives specific to that level.



Question 3

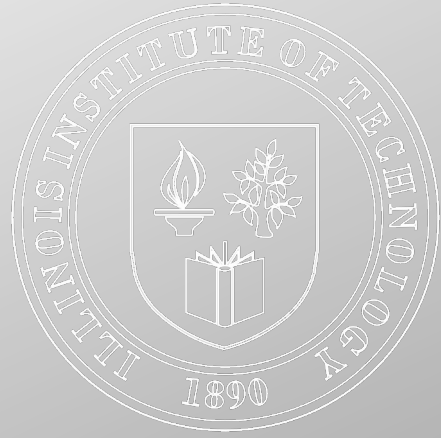
3. Which of the following is a test type?

a. Component testing

b. Functional testing

c. System testing

d. Acceptance testing



Question 4

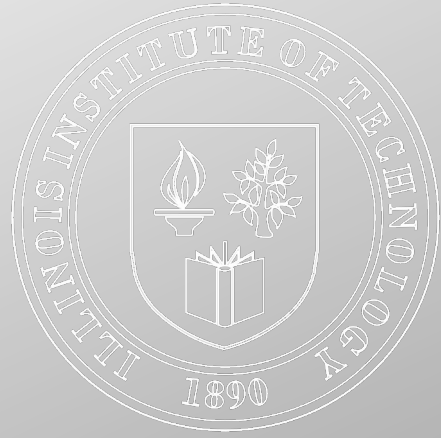
4.Which of the following is a non-functional quality characteristic?

a. Feasibility

b. Usability

c. Maintenance

d. Regression



Question 5

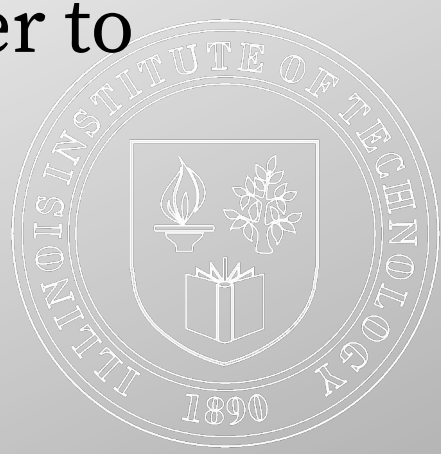
5. Which of these is a functional test?

a. Measuring response time on an on-line booking system.

b. Checking the effect of high volumes of traffic in a call-center system.

c. Checking the on-line bookings screen information and the database contents against the information on the letter to the customers.

d. Checking how easy the system is to use.



Question 6

6. Which of the following is a true statement regarding the process of fixing emergency changes?

a. There is no time to test the change before it goes live, so only the best developers should do this work and should not involve testers as they slow down the process.

b. Just run the retest of the defect actually fixed.

c. Always run a full regression test of the whole system in case other parts of the whole system have been adversely affected.

d. Retest the changed area and then use risk assessment to decide on a reasonable subset of the whole regression test to run in case other parts of the system have been adversely affected.

Question 7

7. A regression test:

a. Is only run once.

b. Will always be automated.

c. Will check unchanged areas of the software to see if they have been affected.

d. Will check changed area of the software to see if they have been affected.

Question 8

8. Non-functional testing includes:

a. Testing to see where the system does not function correctly.

b. Testing the quality attributes of the system including reliability and usability.

c. Gaining user approval for the system

d. Testing a system feature using only the software required for that function.

Question 9

9. Beta testing in:

a. Performed by customers at their own site.

b. Performed by customers at the software developer's site.

c. Performed by an independent test team.

d. Useful to test software developed for a specific customer or user.