LESSON 2

1. Types of testing
2. Testing levels
3. Software Development Life Cycles modes

Manual and Automated

Manual testing is the process through which software developers run tests manually, comparing program expectations and actual outcomes in order to find software defects

VS

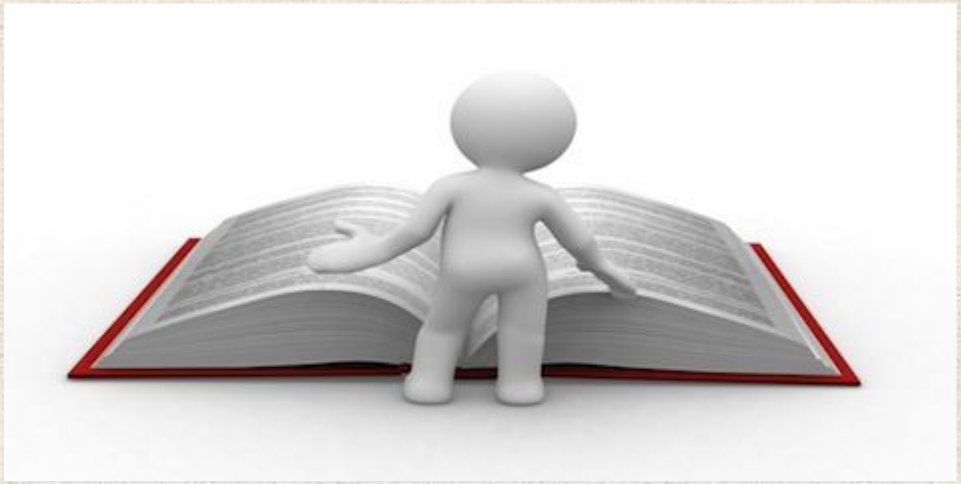
Automated testing is the process through which automated tools run tests that repeat predefined actions, comparing a developing program's expected and actual outcomes.
More info:

Manual Testing	Automated Testing
Time consuming and tedious: Since test cases are executed by human resources so it is very slow and tedious.	**Fast:** Automation runs test cases significantly faster than human resources.
Less reliable: Manual testing is less reliable as tests may not be performed with precision each time because of human errors.	**More reliable:** Automation tests perform precisely same operation each time they are run.
Self-contained: Manual testing can be performed and completed manually and provide self-contained results.	**Not self-contained:** Automation can't be done without manual testing. And you have to manually check the automated test results.
Implicit: Implicit knowledge are used to judge whether or not something is working as expected. This enables engineer to find extra bugs that automated tests would never find.	**Explicit:** Automated tests execute consistently as they don't get tired and/or lazy like us humans.

Verification and Validation

Are we building
the product **right**?

To ensure that work products meet their specified requirements.



Are we building
the **right** product?

To ensure that the product actually meets the user's needs, and that the specifications were correct in the first place.

Positive and Negative

In **positive** testing our intention is

to prove that an application will work on giving valid input data. i.e. testing a system by giving its corresponding valid inputs.

In **negative** testing our intention is

to prove that an application will not work on giving invalid inputs.

Black-box, White-box, Grey-box

Black-box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester.

White-box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

Grey-box Testing is a software testing method which is a combination of Black-box and White-box Testing methods.

Functional testing

Testing based on an analysis of the specification of the functionality of a component or system.

Non-functional testing

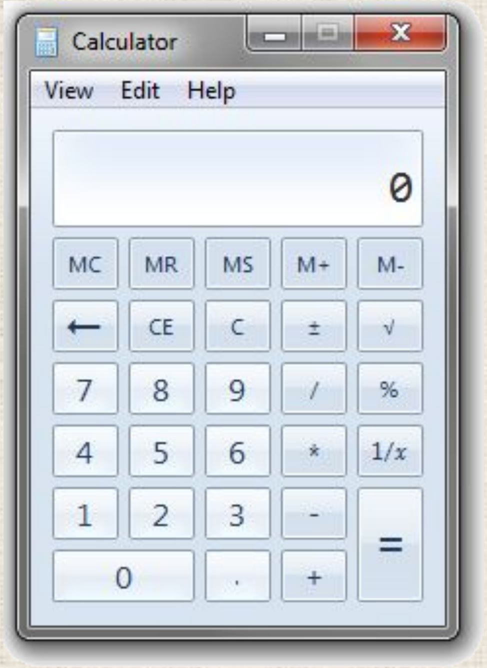
Testing the attributes of a component or system that do not relate to functionality.

Non-functional characteristics are:

- Performance efficiency
- Compatibility
- Usability
- Reliability
- Security
- Maintainability

Functional testing Example #1

- Verify adding of two numbers (5+3 should be 8);
- Verify subtraction of two numbers (5-2 should be 3);
- Verify multiplication of two number (5*3 should be 15);
- Verify division of two numbers (10/2 should be 5);
- Verify getting radical of some number (√25 should be 5);
- Verify multiplication of some number by zero (5*0 should be 0);
- Etc.



Smoke testing

A subset of all defined/planned test cases that cover the main functionality of a component or system, to ascertaining that the most crucial functions of a program work, but not bothering with finer details.

Sanity testing

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

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.

Smoke Testing	**Sanity Testing**
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality / bugs have been fixed
The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Regression testing	Sanity testing is a subset of Acceptance testing
Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

Performance testing

Testing with the intent of determining how efficiently a product handles a variety of events.

Purposes:

✓ demonstrate that the system meets performance criteria;

✓ compare two systems to find which performs better;

✓ measure what parts of the system or workload cause the system to perform badly.

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.

Purposes

- ✓ evaluation of performance and efficiency of software
- ✓ performance optimization (code optimization, server configuration)
- ✓ selection of appropriate hardware and software platforms for the application

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

Purposes:

- ✓ the general study of the behavior of the system under extreme loads
- ✓ examination of handling of errors and exceptions under extreme load
- ✓ examination of certain areas of the system or its components under the disproportionate load
- ✓ testing the system capacity

Localization & Internalization testing

Localization is the process of adapting a globalized application to a particular culture/locale.

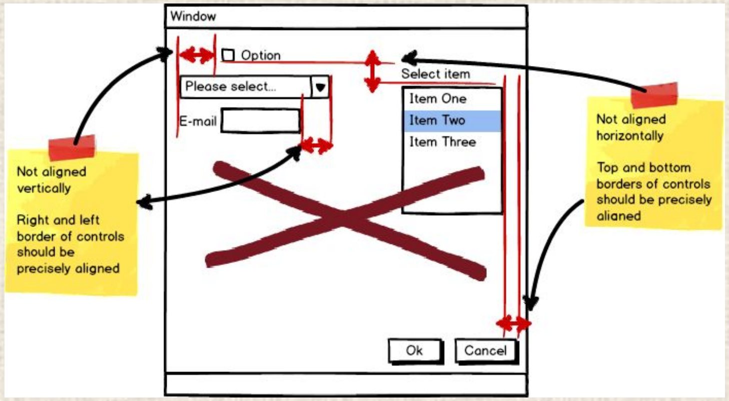
Localization (L10N) testing checks how well the application under test has been Localized into a particular target language.

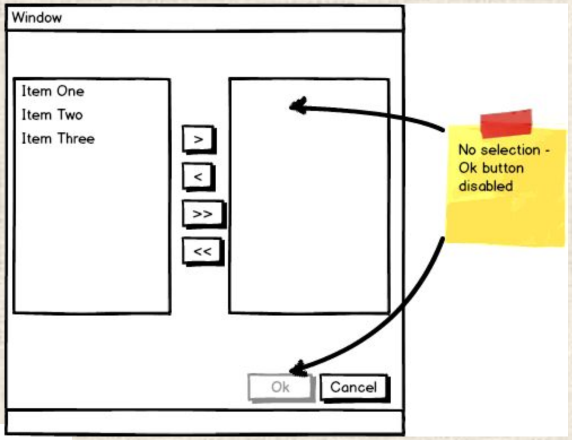
Internationalization is the process of designing and coding a product so it can perform properly when it is modified for use in different languages and locales.

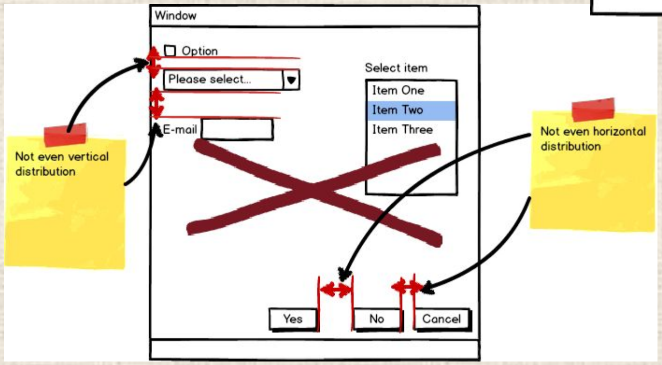
Internationalization (I18N) testing checks if all data/time/number/currency formats are displayed according to selected locale and if all language specific characters are displayed.

UI Testing

The testing a product's graphical user interface to ensure it meets its written specifications







Check if any UI recommendations exist for the application type your team develop. Make sure dialogs you test comply with these recommendations.

Compatibility Testing

Compatibility testing is used to determine if your software is compatible with other elements of a system with which it should operate, e.g. Browsers, Operating Systems, or hardware.

This type of testing helps find out how well a system performs in a particular environment that includes hardware, network, operating system and other software etc.

It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.

Usability Testing

__Usability testing__ a non-functional testing technique that is a measure of how easily the system can be used by end users. It is difficult to evaluate and measure but can be evaluated based on the below parameters:
Level of Skill required to learn/use the software. It should maintain the balance for both novice and expert user.

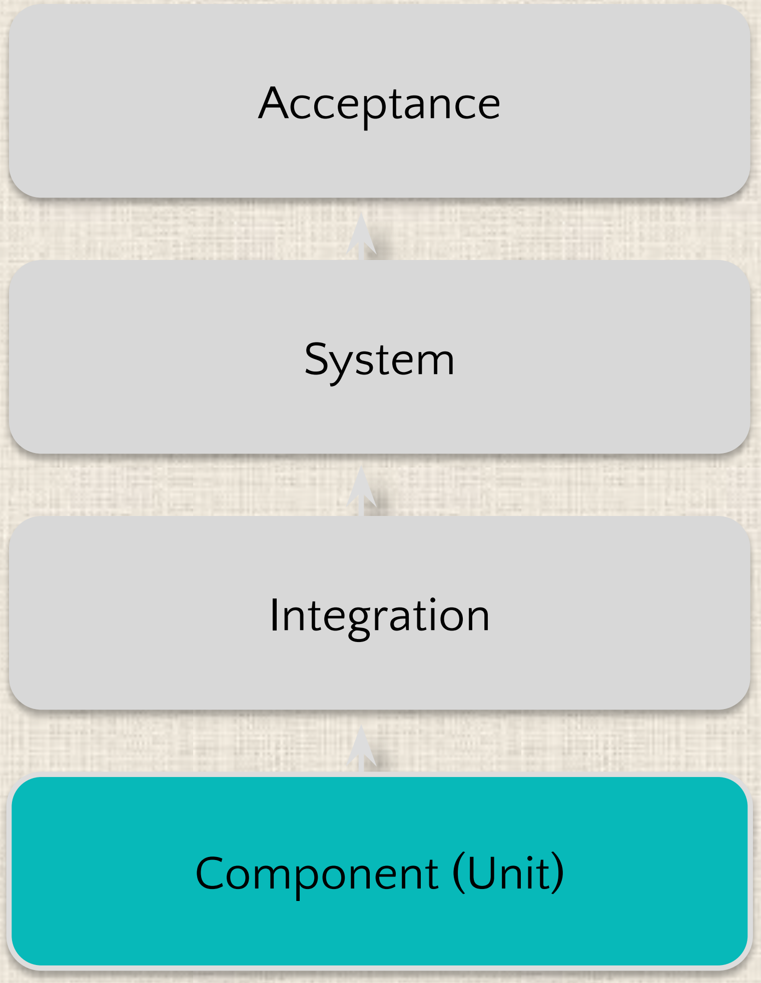
Time required to get used to in using the software.

The measure of increase in user productivity if any.

Assessment of a user's attitude towards using the software.

2. Test Levels

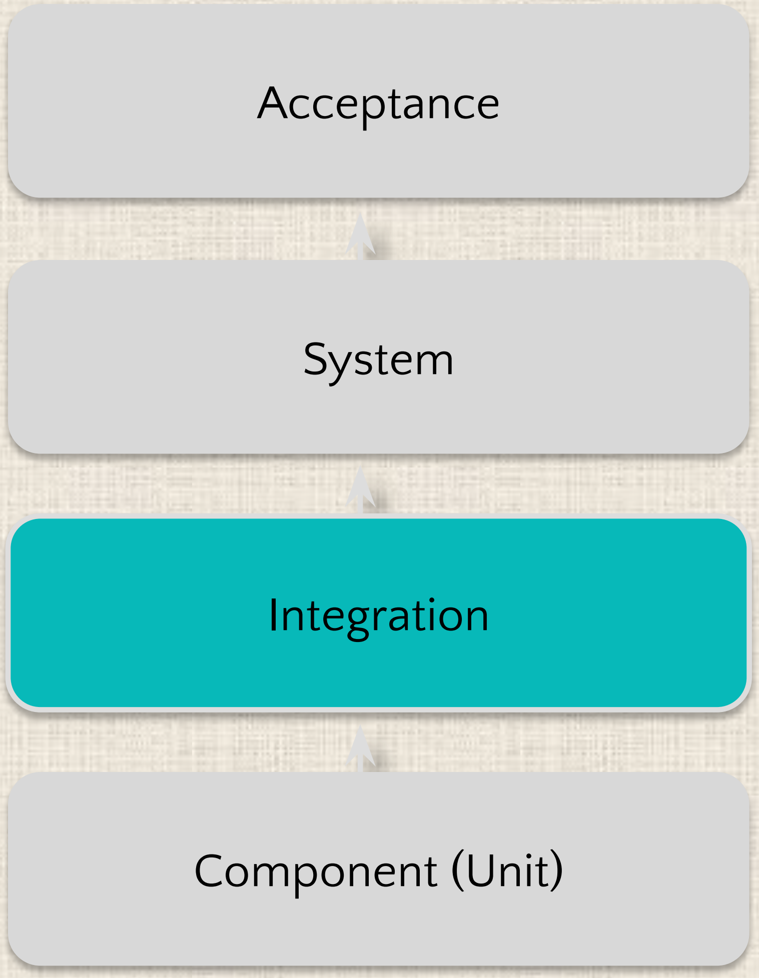
Component



Component (Unit) Test Level	
When	Component is developed
Why	To validate that each unit of the software performs as designed
How	White-box testing

Testing on the Component (Unit) Test Level is called Component (Unit) testing

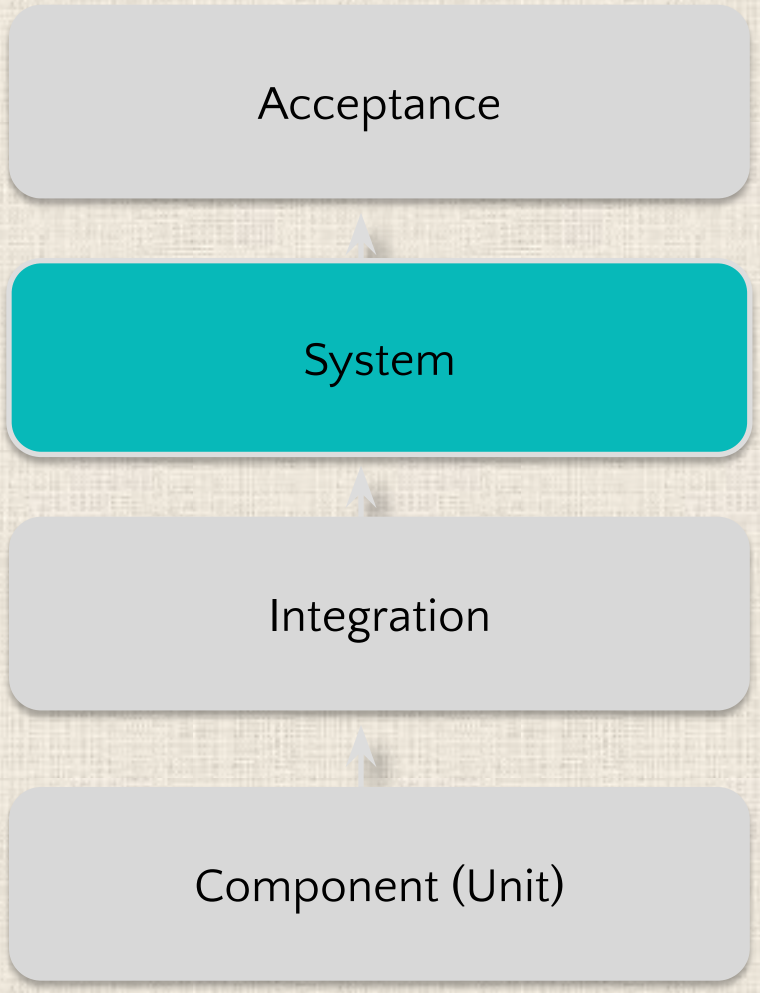
Integration



Integration Test Level	
When	Units to be integrated are developed
Why	To expose faults in the interaction between integrated units
How	White-box/ Black-box/ Grey-box Depends on definite units

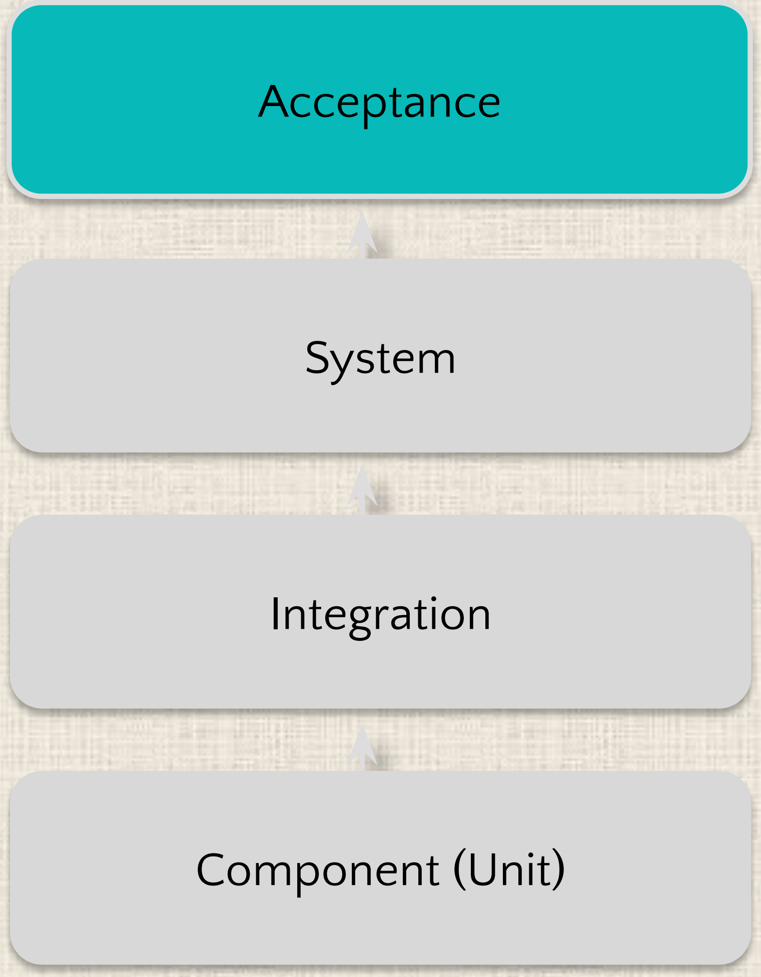
Testing on the Integration Test Level is called Integration testing

System



System Test Level	
When	Separate units are integrated into System
Why	To evaluate the system's compliance with the specified requirements
How	Black-box testing

Acceptance

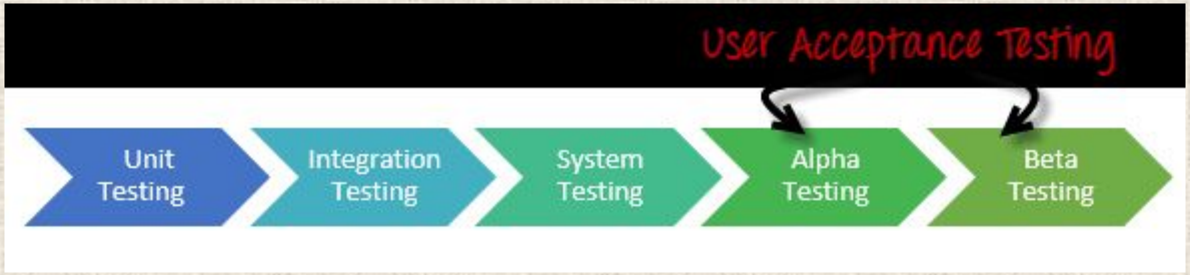


Acceptance Test Level	
When	Component is developed
Why	To evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery
How	Black-box testing

Testing on the Acceptance Test Level is called Acceptance testing

Alfa & Beta Testing

Alfa testing takes place at the developer's site by the internal teams, before release to external customers. This testing is performed without the involvement of the development teams.

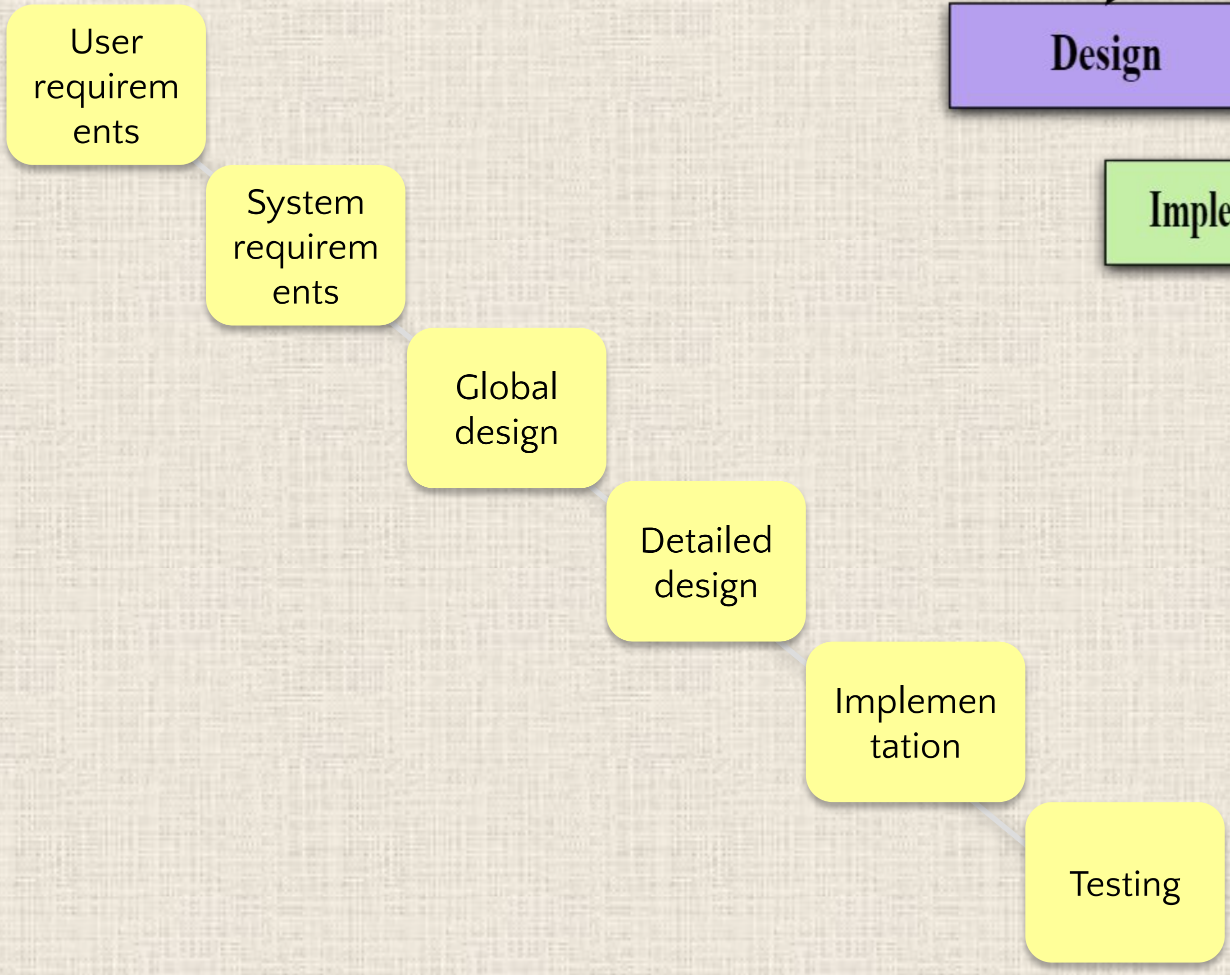


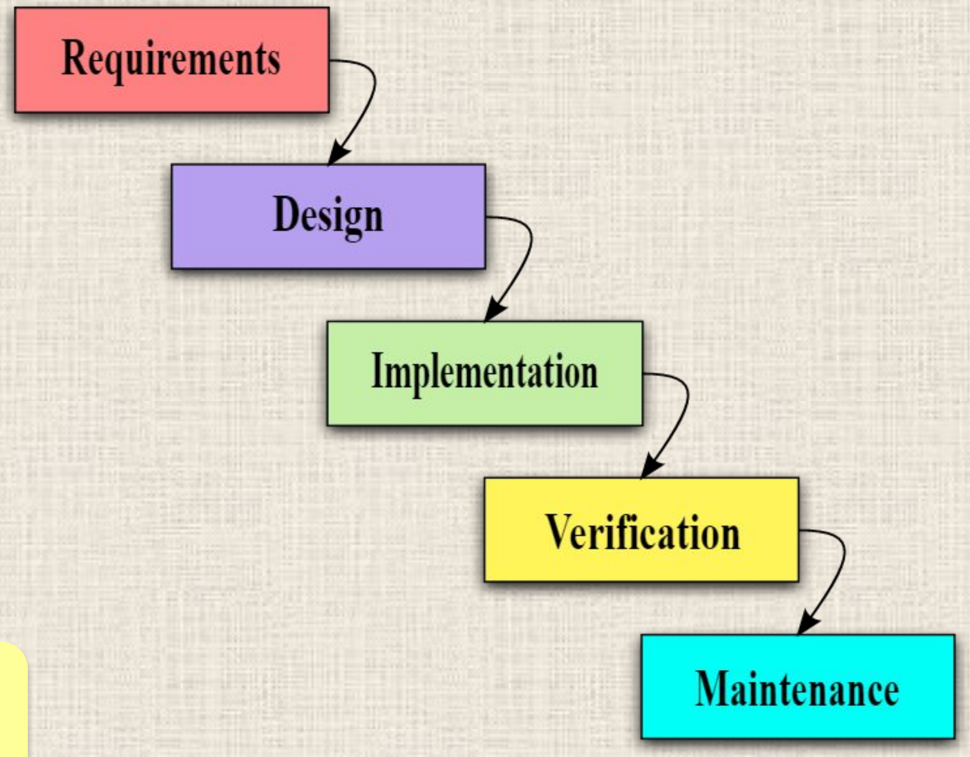
Beta testing also known as user testing takes place at the end users site by the end users to validate the usability, functionality, compatibility, and reliability testing.

Beta testing adds value to the software development life cycle as it allows the "real" customer an opportunity to provide inputs into the design, functionality, and usability of a product. These inputs are not only critical to the success of the product but also an investment into future products when the gathered data is managed effectively.

3. SDLC models

WATERFALL


User requirements
System requirements
Global design
Detailed design
Implementation
Testing

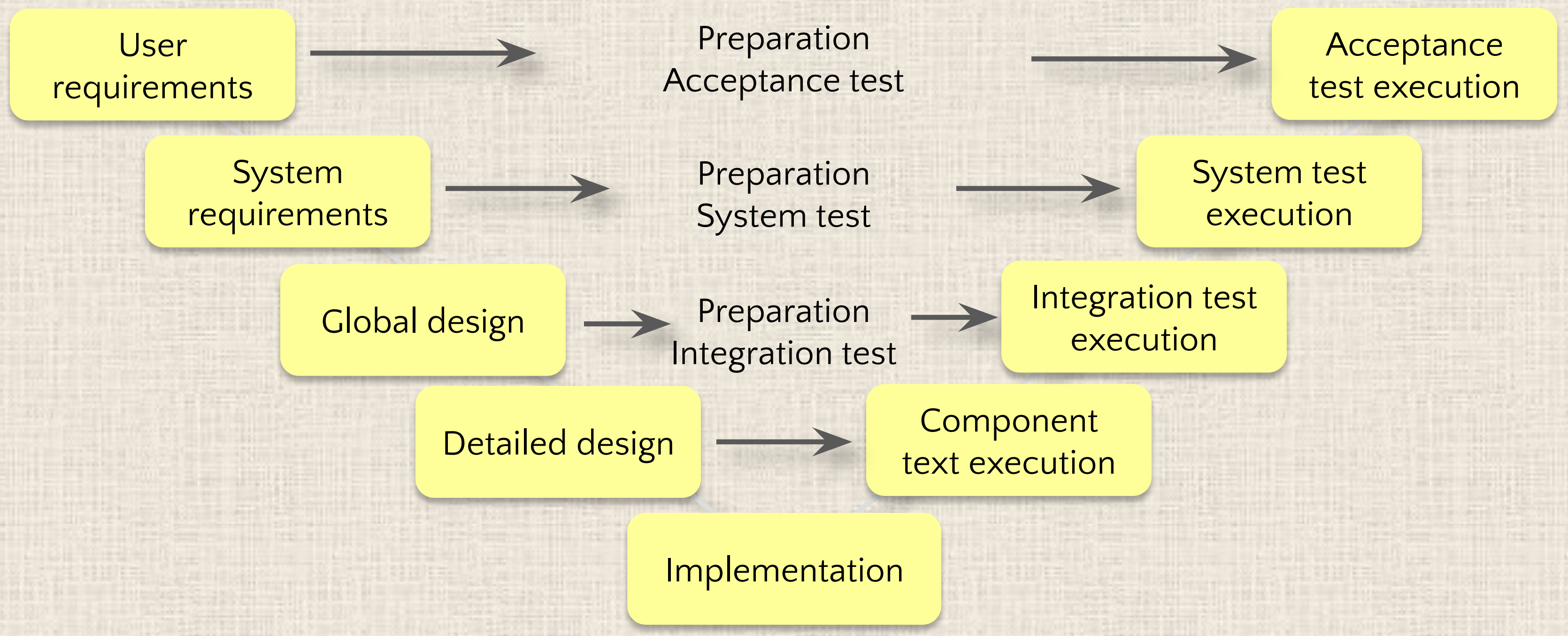

Requirements
Design
Implementation
Verification
Maintenance

Waterfall

- ✓ Time spent early in the software production cycle can lead to greater economy at later stages
- ✓ Waterfall model places emphasis on documentation
- ✓ Waterfall model has simple approach and is more disciplined
- ✓ Easily identifiable milestones and deliverables
- ✓ Track progress easily due to clear stages

- ✓ Inflexible: difficult to respond to changing requirements
- ✓ No working software is produced until late during the life cycle.
- ✓ Some problems in requirements, design and coding might be not discovered until system testing
- ✓ Defects cost is high

V-MODEL

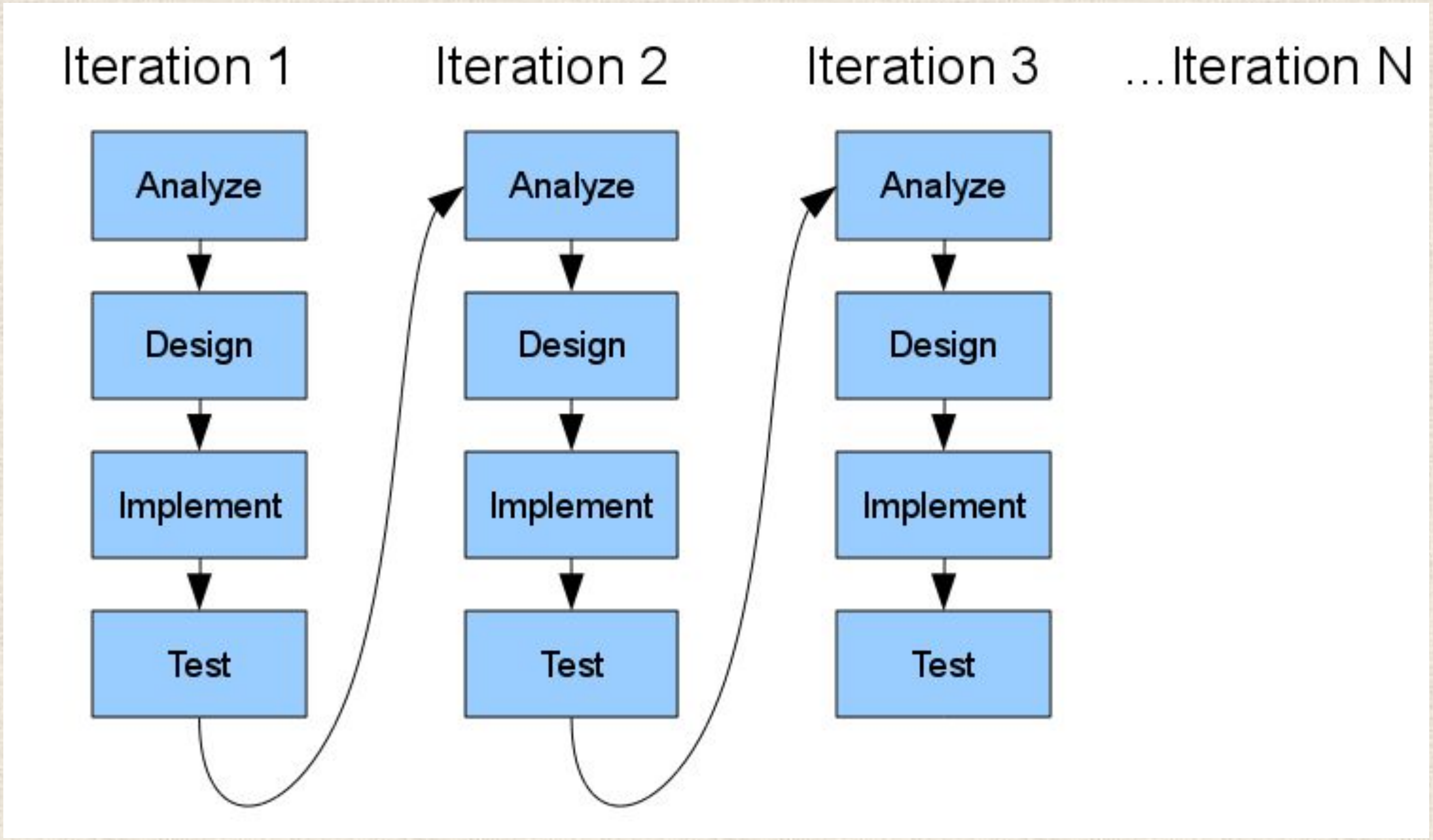

User requirements
Preparation Acceptance test
Acceptance test execution
System requirements
Preparation System test
System test execution
Global design
Preparation Integration test
Integration test execution
Detailed design
Component text execution
Implementation

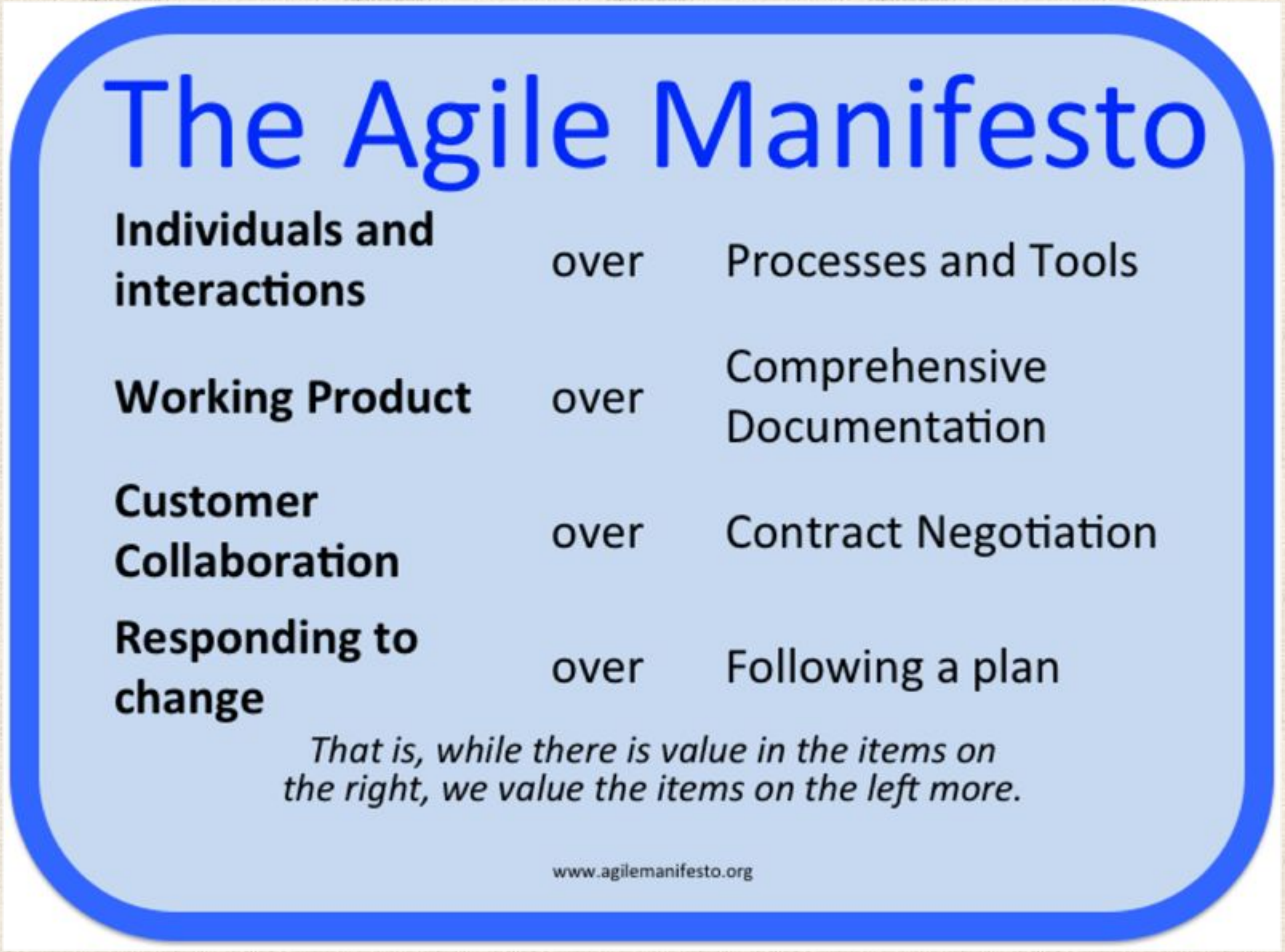
V-model

- ✓ Time spent early in the software production cycle can lead to greater economy at later stages
- ✓ Easily identifiable milestones and deliverables
- ✓ Testing activities like planning, test designing happens well before coding. This saves a lot of time
- ✓ Proactive defect tracking – that is defects are found at early stages when they are introduced

- ✓ Rigid and Inflexible: difficult to respond to changing requirements
- ✓ If any changes happen mid way, not only the requirements documents but also the test Documentation needs to be updated
- ✓ No working software is produced until late during the life cycle.

Iterative model





The Agile Manifesto

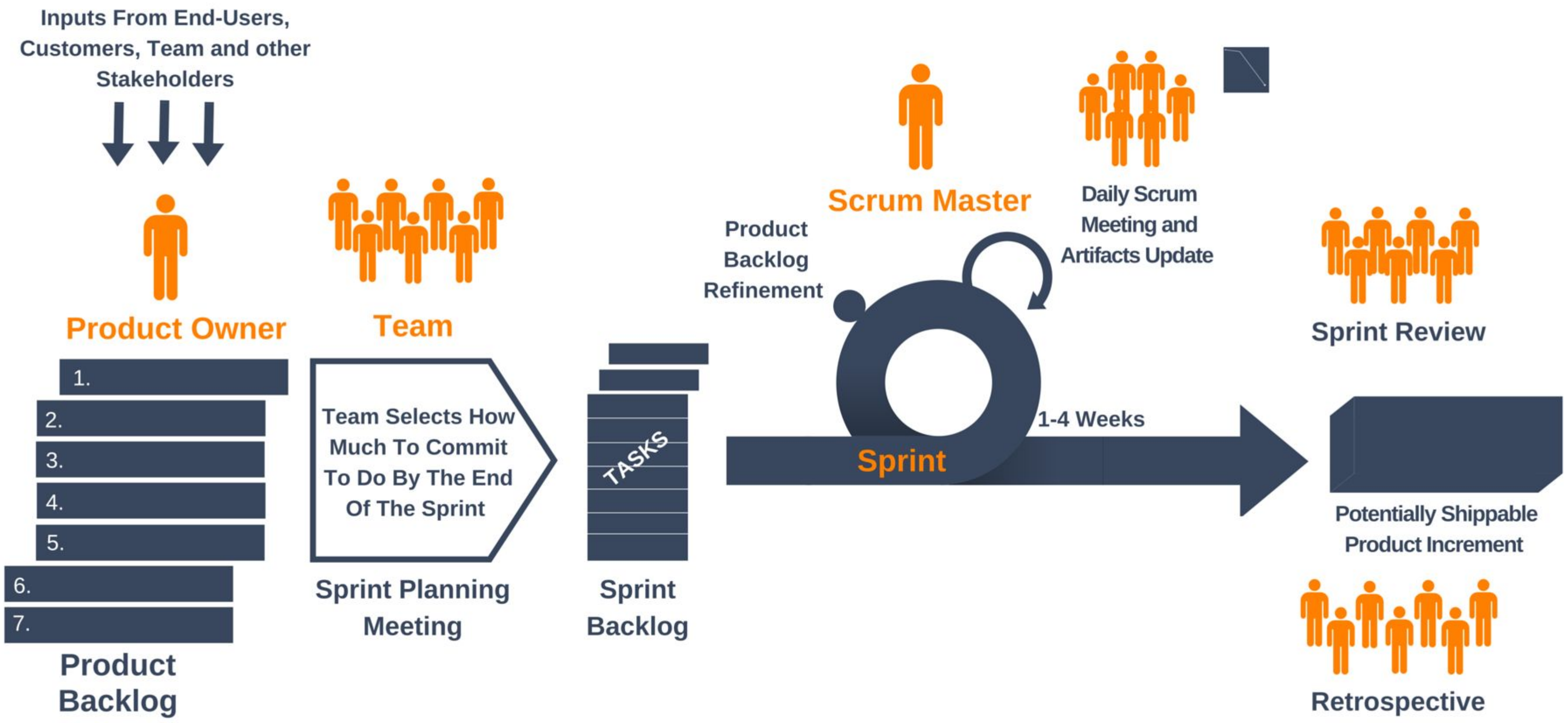
Individuals and interactions	over	Processes and Tools
Working Product	over	Comprehensive Documentation
Customer Collaboration	over	Contract Negotiation
Responding to change	over	Following a plan

That is, while there is value in the items on the right, we value the items on the left more.

www.agilemanifesto.org

SCRUM

SCRUM is an iterative and incremental agile software development framework for managing product development.


Inputs From End-Users, Customers, Team and other Stakeholders
Product Owner
Team
1.
2.
3.
4.
5.
6.
7.
Product Backlog
Team Selects How Much To Commit To Do By The End Of The Sprint
Sprint Planning Meeting
TASKS
Sprint Backlog
Product Backlog Refinement
Scrum Master
Daily Scrum Meeting and Artifacts Update
Sprint
1-4 Weeks
Sprint Review
Potentially Shippable Product Increment
Retrospective

SCRUM

- ✓ Great emphasis on team work
- ✓ Team learns and contributes throughout the process, team becomes autonomous and strives for excellence
- ✓ Iterative model leading to a delivery every sprint
- ✓ Frequent and late changes welcoming
- ✓ Creates an open environment and encourages immediate feedback

- ✓ The basic premise that the team is committed to the project. If the team is not committed then process collapses
- ✓ The size of the team is restricted due to the involvement of all team members
- ✓ Reliance on experience
- ✓ The management's comfort level in delegation of tasks

Practice to lesson 2:

1. Try to find bugs on next sites and describe them in Jira – "Website bugs (lection 2)" project (see next slide for the details):
   a. http://www.tort.ua/
   b. http://zoocomplex.com.ua/

Please, send me email when you will report all found bugs with their numbers.

1. Additional hometask:

Create checklist (based on different test types) in *excel/txt* file and send it via email – choose one point from the list (8 points in list):

b. window
c. pen
d. stapler
e. elevator
f. control desk

Jira using:

1. Open site
2. Log in (choose the project - web sites)
3. Click on Create button
4. Choose next fields:

- ☐ Issue type: bug
- ☐ Summary (add summary to your bug)
- ☐ Description (add next info: Created by; Environment description; Preconditions (if needed); Step to reproduce; Actual Result; Expected Result; Additional info(if needed))
- ☐ set priority (as severity)

7. Files (add some screenshots if needed)

8. Click on Create button

/!\Note: don't forget to add your name to the created issue

LINK to Jira tool: https://group7-skillup.atlassian.net/secure/Dashboard.jspa
login: testSkillup@i.ua
password: QAskillup17

Thank you!