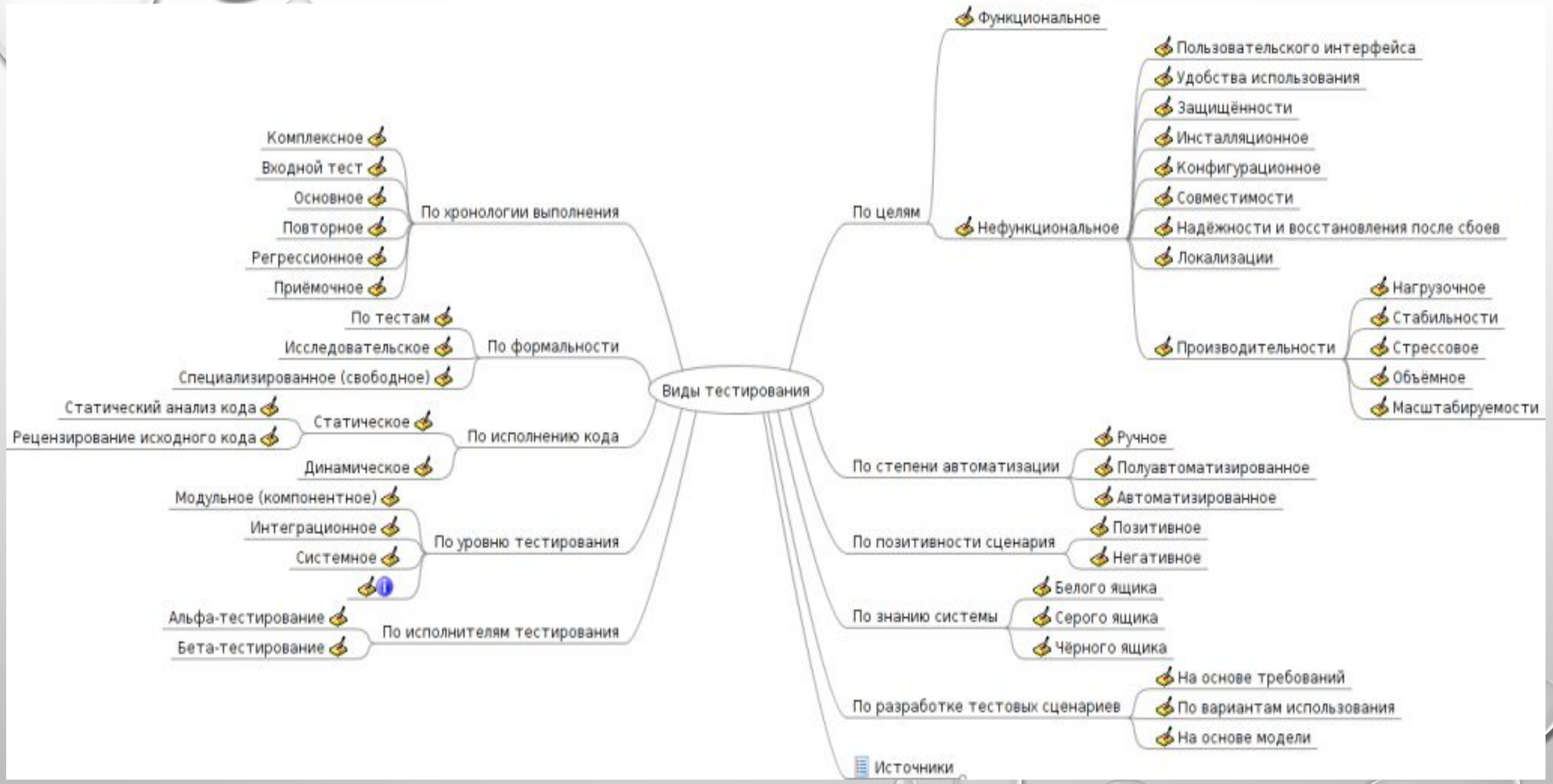
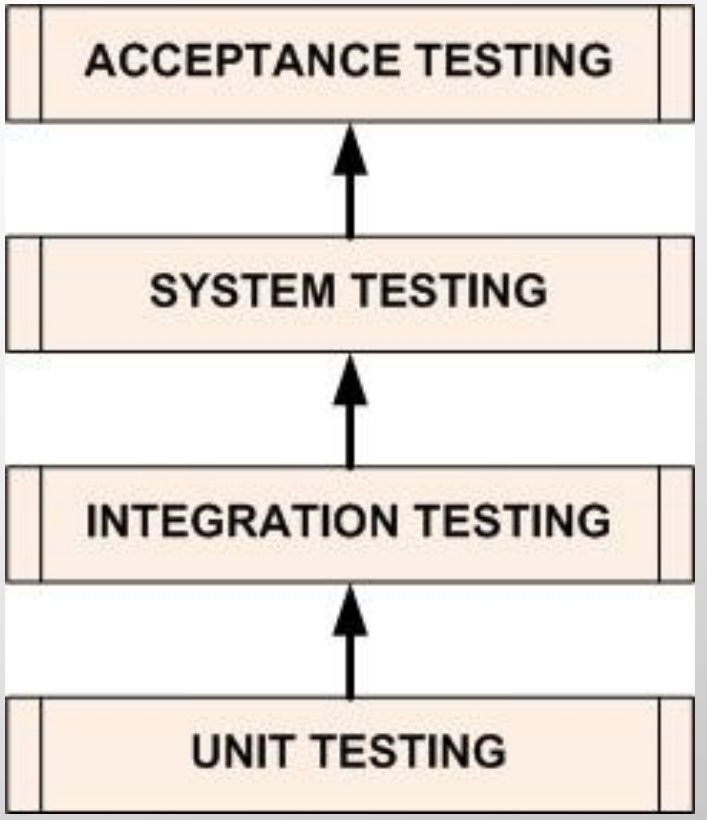
LEVELS AND TYPES OF TESTING

BASIC CLASSIFICATION


Виды тестирования
По целям
Функциональное
Нефункциональное
Пользовательского интерфейса
Удобства использования
Защищённости
Инсталляционное
Конфигурационное
Совместимости
Надёжности и восстановления после сбоев
Локализации
Производительности
Нагрузочное
Стабильности
Стрессовое
Объёмное
Масштабируемости
По степени автоматизации
Ручное
Полуавтоматизированное
Автоматизированное
По позитивности сценария
Позитивное
Негативное
По знанию системы
Белого ящика
Серого ящика
Чёрного ящика
По разработке тестовых сценариев
На основе требований
По вариантам использования
На основе модели
Источники
По хронологии выполнения
Комплексное
Входной тест
Основное
Повторное
Регрессионное
Приёмочное
По формальности
По тестам
Исследовательское
Специализированное (свободное)
По исполнению кода
Статическое
Статический анализ кода
Рецензирование исходного кода
Динамическое
По уровню тестирования
Модульное (компонентное)
Интеграционное
Системное
По исполнителям тестирования
Альфа-тестирование
Бета-тестирование

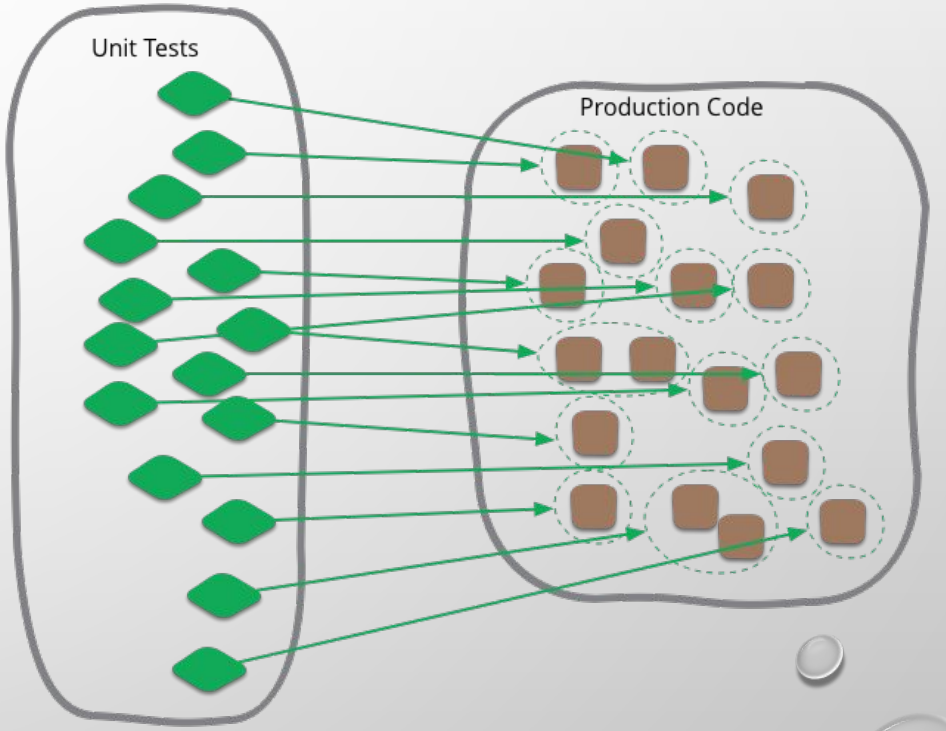
LEVELS OF TESTING

1. **UNIT TESTING**
2. **INTEGRATION TESTING**
3. **SYSTEM TESTING**
4. **OPERATION AND RELEASE TESTING**
5. **ACCEPTANCE TESTING**



1. UNIT TESTING

- THIS TYPE OF TESTING IS PERFORMED BY DEVELOPERS BEFORE THE SETUP IS HANDED OVER TO THE TESTING TEAM TO FORMALLY EXECUTE THE TEST CASES. UNIT TESTING IS PERFORMED BY THE RESPECTIVE DEVELOPERS ON THE INDIVIDUAL UNITS OF SOURCE CODE ASSIGNED AREAS.
- THE GOAL OF UNIT TESTING IS TO ISOLATE EACH PART OF THE PROGRAM AND SHOW THAT INDIVIDUAL PARTS ARE CORRECT IN TERMS OF REQUIREMENTS AND FUNCTIONALITY.



(DEVELOPERS)

LIMITATIONS OF UNIT TESTING

- TESTING CANNOT CATCH EACH AND EVERY BUG IN AN APPLICATION. IT IS IMPOSSIBLE TO EVALUATE EVERY EXECUTION PATH IN EVERY SOFTWARE APPLICATION. THE SAME IS THE CASE WITH UNIT TESTING.
- THERE IS A LIMIT TO THE NUMBER OF SCENARIOS AND TEST DATA THAT A DEVELOPER CAN USE TO VERIFY A SOURCE CODE.
- AFTER HAVING EXHAUSTED ALL THE OPTIONS, THERE IS NO CHOICE BUT TO STOP UNIT TESTING AND MERGE THE CODE SEGMENT WITH OTHER UNITS.

2. INTEGRATION TESTING

- INTEGRATION TESTING IS DEFINED AS THE TESTING OF COMBINED PARTS OF AN APPLICATION TO DETERMINE IF THEY FUNCTION CORRECTLY.

  (DEVELOPERS AND TESTERS)

3. SYSTEM TESTING

- SYSTEM TESTING TESTS THE SYSTEM AS A WHOLE. THIS TYPE OF TESTING IS PERFORMED BY A SPECIALIZED TESTING TEAM.
- **SYSTEM TESTING IS IMPORTANT BECAUSE OF THE FOLLOWING REASONS:**

- SYSTEM TESTING IS THE FIRST STEP IN THE SOFTWARE DEVELOPMENT LIFE CYCLE, WHERE THE APPLICATION IS TESTED AS A WHOLE.
- THE APPLICATION IS TESTED THOROUGHLY TO VERIFY THAT IT MEETS THE FUNCTIONAL AND TECHNICAL SPECIFICATIONS.
- THE APPLICATION IS TESTED IN AN ENVIRONMENT THAT IS VERY CLOSE TO THE PRODUCTION ENVIRONMENT WHERE THE APPLICATION WILL BE DEPLOYED.
- SYSTEM TESTING ENABLES US TO TEST, VERIFY, AND VALIDATE BOTH THE BUSINESS REQUIREMENTS AS WELL AS THE APPLICATION ARCHITECTURE.

- (**TESTERS**)

System Testing	Integration Testing
1. Testing the completed product to check if it meets the specification requirements.	1. Testing the collection and interface modules to check whether they give the expected result
2. Both functional and non-functional testing are covered like sanity, usability, performance, stress an load .	2.Only Functional testing is performed to check whether the two modules when combined give correct outcome.
3. It is a high level testing performed after integration testing	3. It is a low level testing performed after unit testing
4. It is a black box testing technique so no knowledge of internal structure or code is required	4. It is both black box and white box testing approach so it requires the knowledge of the two modules and the interface
5. It is performed by test engineers only	5. Integration testing is performed by developers as well test engineers
6. Here the testing is performed on the system as a whole including all the external interfaces, so any defect found in it is regarded as defect of whole system	6. Here the testing is performed on interface between individual module thus any defect found is only for individual modules and not the entire system
7. In System Testing the test cases are developed to simulate real life scenarios	7. Here the test cases are developed to simulate the interaction between the two module
8. The System testing covers many different testing types like sanity, usability, maintenance, regression, retesting and performance	8. Integration testing techniques includes big bang approach, top bottom , bottom to top and sandwich approach.

4. RELEASE TESTING

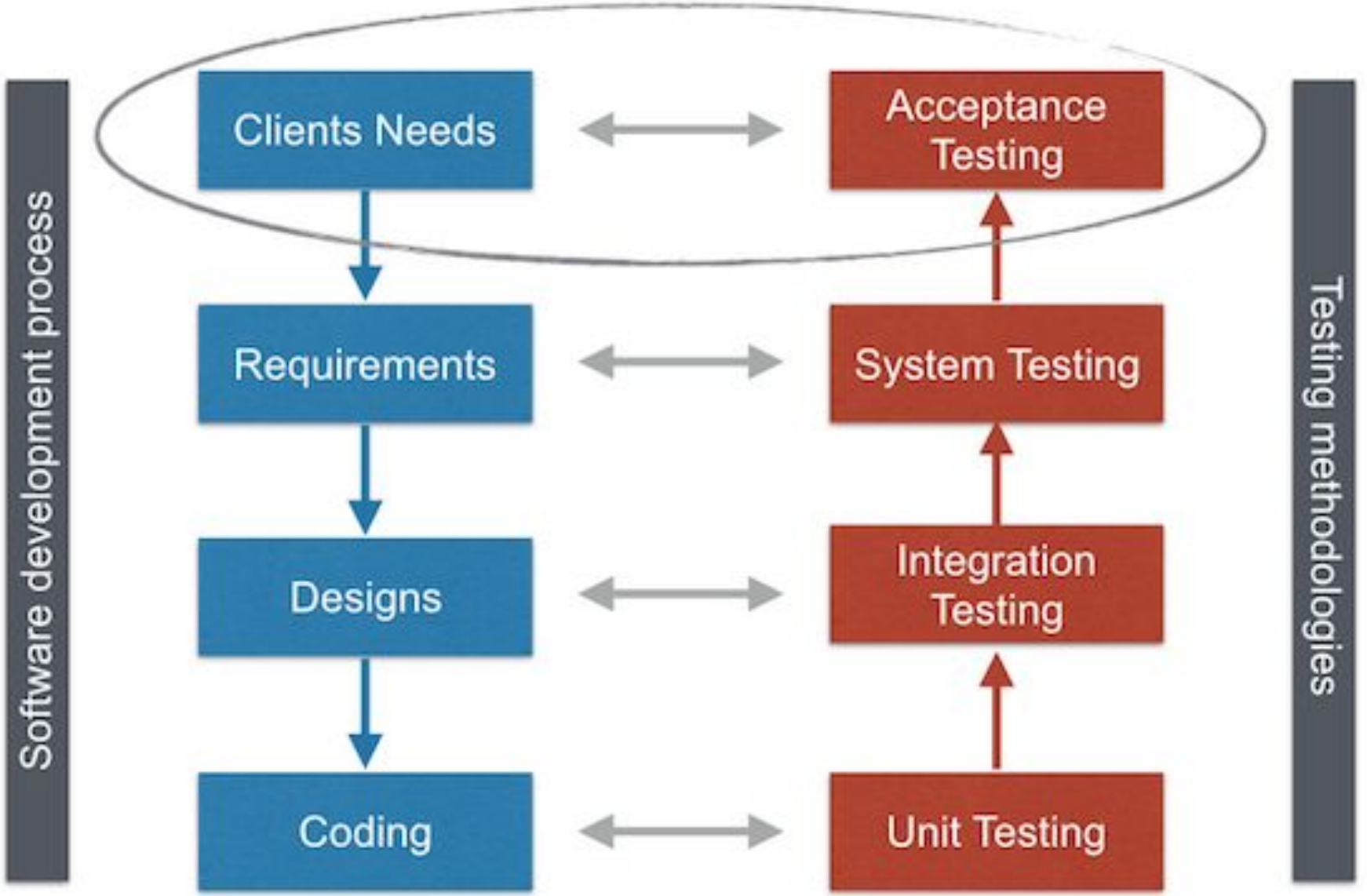
- A SEPARATE TEAM THAT HAS NOT BEEN INVOLVED IN THE SYSTEM DEVELOPMENT SHOULD BE RESPONSIBLE FOR RELEASE TESTING.
- SYSTEM TESTING BY THE DEVELOPMENT TEAM SHOULD FOCUS ON DISCOVERING BUGS IN THE SYSTEM (DEFECT TESTING). THE OBJECTIVE OF RELEASE TESTING IS TO CHECK THAT THE SYSTEM MEETS ITS REQUIREMENTS AND IS GOOD ENOUGH FOR EXTERNAL USE (VALIDATION TESTING).
- (TESTERS AND USERS)

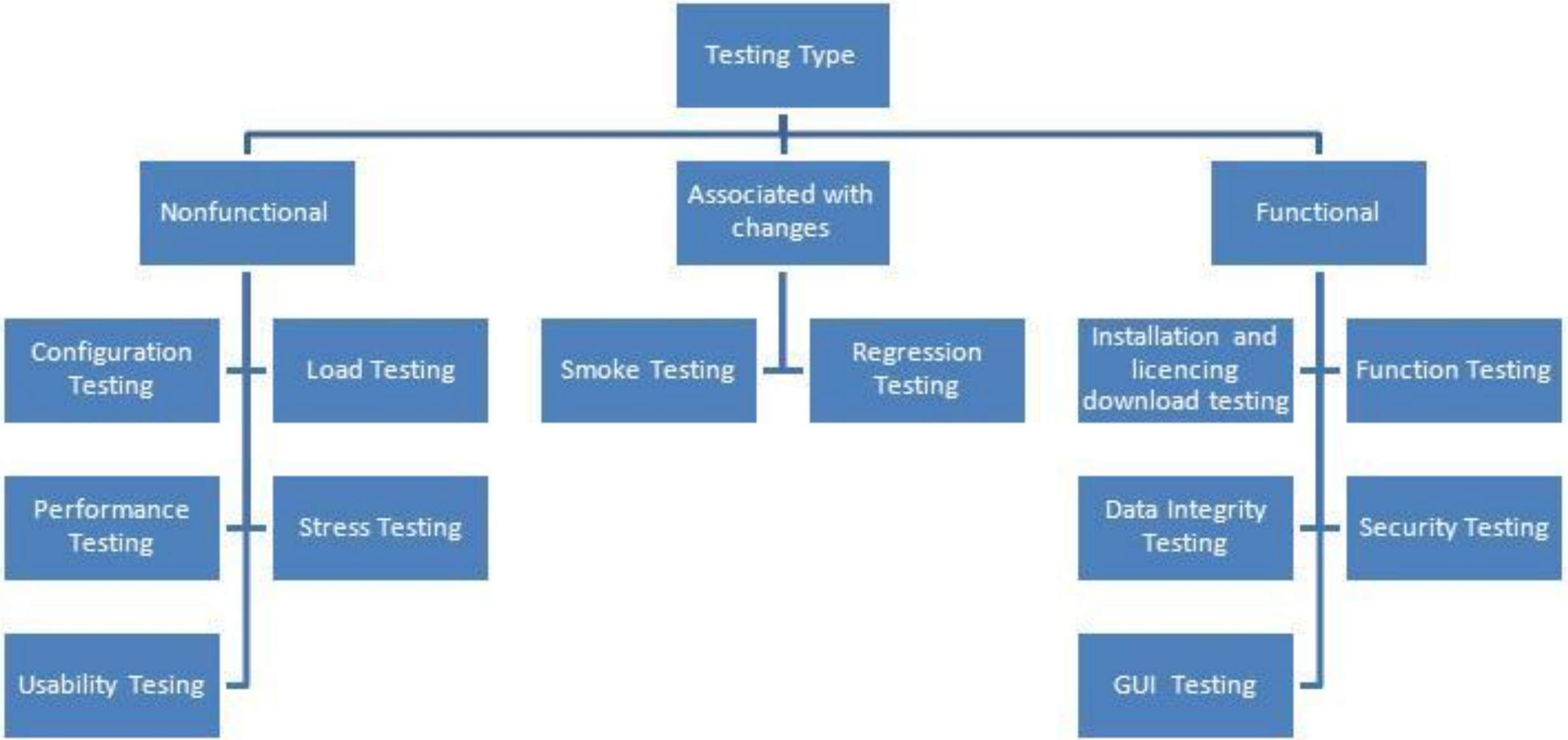
5. ACCEPTANCE TESTING

- THE MAIN QUESTION IS – DOES THE CLIENT OR A FUTURE USER NEED THIS FUNCTIONALITY
- EXAMPLE:

LET'S SAY, FACEBOOK LAUNCHES A NEW FEATURE, ALLOWING FACEBOOK USERS TO SEND POSTCARDS TO FAMILY & FRIENDS. TECHNICALLY THE IMPLEMENTED SOLUTION WORKS. TESTERS ALSO CAN USE IT – HOWEVER DUE TO LACK OF INTEREST AND NEED, NO ONE WILL WANT TO SEND PRINTED POSTCARDS. FUNCTIONAL TESTS WOULD GO WELL, USABILITY TESTS WOULD GO FINE TOO, BUT THE USER ACCEPTANCE TEST WOULD PROBABLY FAIL AS FACEBOOK USERS DO NOT DEMAND TO SEND POSTCARDS WITHIN FACEBOOK.

(TESTER, BETA-TESTERS AND USERS)


Software development process
Clients Needs
Requirements
Designs
Coding
Acceptance Testing
System Testing
Integration Testing
Unit Testing
Testing methodologies


Testing Type
Nonfunctional
Associated with changes
Functional
Configuration Testing
Load Testing
Performance Testing
Stress Testing
Usability Tesing
Smoke Testing
Regression Testing
Installation and licencing download testing
Function Testing
Data Integrity Testing
Security Testing
GUI Testing

TYPES OF TESTING

1. **ACCORDING TO IT'S GOALS TESTING CAN BE:**

FUNCTIONAL AND NON-FUNCTIONAL

FUNCTIONAL TYPES:

1. **FUNCTIONS TESTING** (TESTING THE APP'S/PROGRAM'S FUNCTIONALITY ACCORDING TO SPECIFICATIONS AND USERS' NEEDS)
2. **SECURITY AND ACCESS CONTROL TESTING** (TESTING SYSTEMS LEVEL OF SECURITY, HOW IT IS READY TO FACE HACKER'S ATTACKS, MONEY/DATA STEELING ATTEMPTS, VIRUSES, UNAUTHORIZED ACCESS TO PRIVATE DATA ETC.)
3. **INTEROPERABILITY** (CHECKS HOW THE SYSTEM IS FUNCTIONING TOGETHER WITH OTHER SYSTEMS)

TYPES OF TESTING

NON- FUNCTIONAL TYPES:

ALL TYPES OF **PERFORMANCE** TESTING:

- PERFORMANCE AND LOAD TESTING
- STRESS TESTING
- STABILITY/RELIABILITY TESTING
- VOLUME TESTING

AND OTHERS:

- INSTALLATION TESTING
- USABILITY TESTING
- FAILOVER AND RECOVERY TESTING
- CONFIGURATION TESTING

PERFORMANCE TESTING

LOAD TESTING — AUTOMATED TESTING IMITATING THE WORK OF SOME (NORMAL, EXTREME OR PEAK) QUANTITY OF USERS

STRESS TESTING – HELPS CHECK HOW THE SYSTEM REACTS TO STRESS SITUATIONS. FOR EXAMPLE, IT MAY INCLUDE TAKING AWAY SOME RESOURCES OR APPLYING A LOAD BEYOND THE ACTUAL LOAD LIMIT.

STABILITY / RELIABILITY TESTING USING THE SYSTEM FOR A LONG TIME WITH NORMAL LOAD

VOLUME TESTING - CHECKING HOW THE SYSTEM WORKS WITH LARGE AMOUNTS OF DATA IN THE DATABASE

- **INSTALLATION TESTING ARE INSTALLATION/DEINSTALLATION/RE-INSTALLATION AND UPDATE GOING SUCCESSFULLY**
- **USABILITY TESTING** — ***UX*** AND UI TESTING. IS USING THE APP COMFORTABLE FOR USERS
- **FAILOVER AND RECOVERY TESTING** HOW THE SYSTEM IS RECOVERING AFTER CRASHES (LIKE POWER BREAKS OR SERVER/DATABASE FAILURES
- **CONFIGURATION TESTING** — HOW THE SYSTEM WORKS UNDER DIFFERENT CONFIGURATIONS OF HARDWARE AND SOFTWARE (OPERATION SYSTEM, BROWSER, SCREEN RESOLUTION…)

2. ACCORDING TO CHANGES TESTING CAN BE:

- SMOKE TESTING
- REGRESSION TESTING
- RE-TESTING
- SANITY TESTING

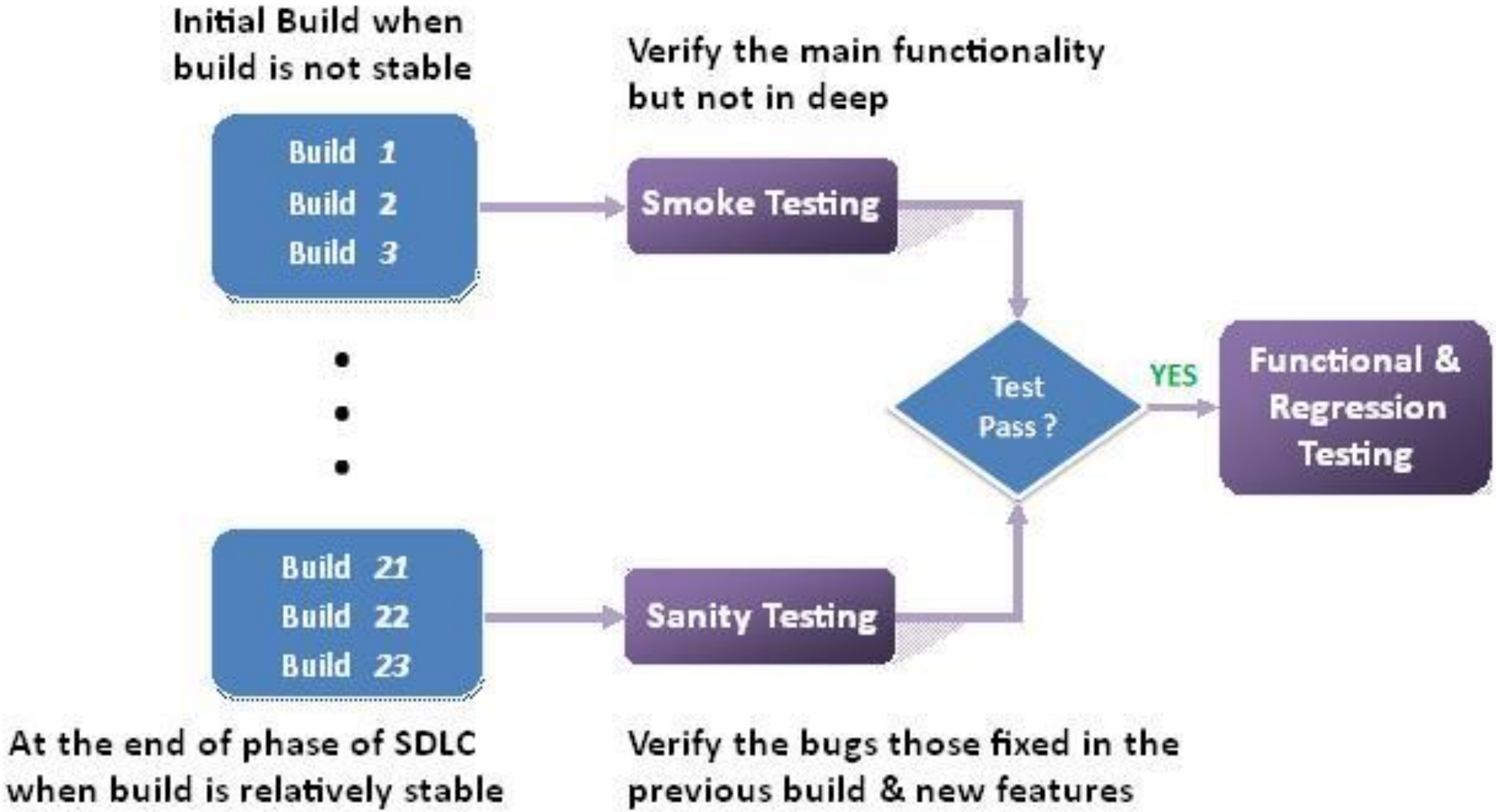
SMOKE TESTING SHORT SET OF TESTS THAT IS DESIGNED TO VERIFY THAT AFTER COMPILING A BUILT (FIRST VERSION OR UPDATE) SYSTEM STARTS AND FULFILLS BASIC TASKS. ONLY AFTER SMOKE TESTS ALL OTHERS CAN BE STARTED.

REGRESSION TESTING — WHENEVER A CHANGE IN A SOFTWARE APPLICATION IS MADE, IT IS QUITE POSSIBLE THAT OTHER AREAS WITHIN THE APPLICATION HAVE BEEN AFFECTED BY THIS CHANGE. REGRESSION TESTING IS PERFORMED TO VERIFY THAT A FIXED BUG HASN'T RESULTED IN ANOTHER FUNCTIONALITY OR BUSINESS RULE VIOLATION. THE INTENT OF REGRESSION TESTING IS TO ENSURE THAT A CHANGE, SUCH AS A BUG FIX SHOULD NOT RESULT IN ANOTHER FAULT BEING UNCOVERED IN THE APPLICATION.

RE-TESTING — CHECKING THAT THE FIXED MISTAKES ARE REALLY FIXED AS REPORTED

SANITY TESTING — SANITY TESTING IS USUALLY PERFORMED WHEN ANY MINOR **BUG** IS FIXED OR WHEN THERE IS A SMALL CHANGE IN THE FUNCTIONALITY. IT IS A KIND OF SOFTWARE TESTING WHICH IS DONE BY THE TESTERS TO ENSURE THAT THE FUNCTIONALITY IS WORKING AS EXPECTED. SANITY TESTING IS NARROW AND DEEP. UNLIKE **SMOKE TESTING**, SANITY TESTING IS FOCUSED ON ONE OR TWO FUNCTIONALITIES WHEREAS SMOKE TESTING IS DONE TO ENSURE THAT ALL THE MAJOR FEATURES OF THE PROJECT IS WORKING FINE.

*WHAT'S THE DIFFERENCE BETWEEN REGRESSION TESTING, SANITY AND RE-TESTING?
RE-TESTING — BUG FIXING IS CHECKED
SANITY TESTING — CHECKING THAT NEW BUGS WERE NOT PRODUCED DURING BUG FIXESREGRESSION TESTING – CHECKING THAT OLD **FUNCTIONALITY** WORKS CORRECTLY AFTER ADDING NEW FUNCTIONS


Initial Build when build is not stable
Verify the main functionality but not in deep
Build 1
Build 2
Build 3
Smoke Testing
Test Pass ?
YES
Functional & Regression Testing
Build 21
Build 22
Build 23
Sanity Testing
At the end of phase of SDLC when build is relatively stable
Verify the bugs those fixed in the previous build & new features

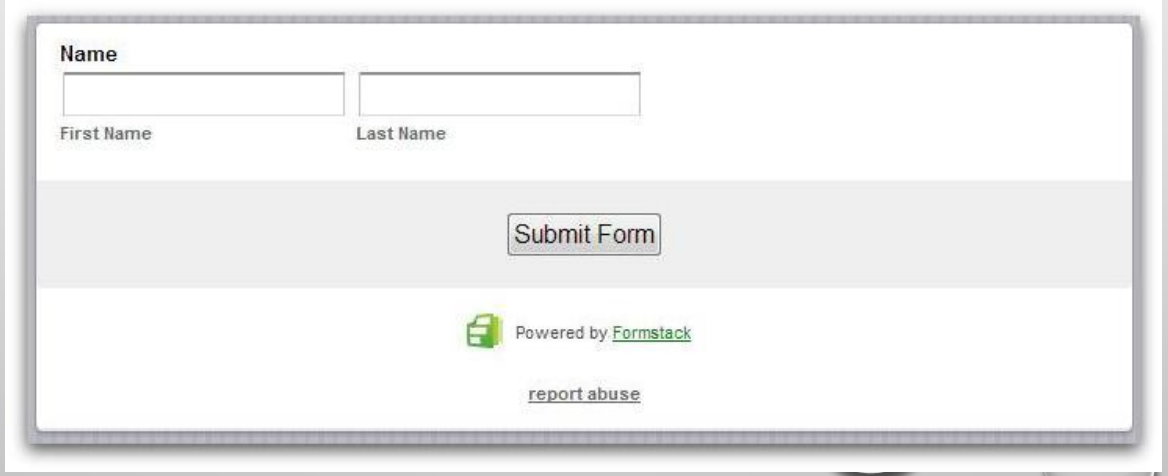
3. ACCORDING TO CODE RUNNING: STATIC AND DYNAMIC

- **STATIC TESTING**: NO CODE IS RUNNING (CODE REVIEW, DOCUMENTATION TESTING)
- **DYNAMIC TESTING**: CODE IS RUNNING (THE SYSTEM WORKS WHILE TEST ARE EXECUTED)

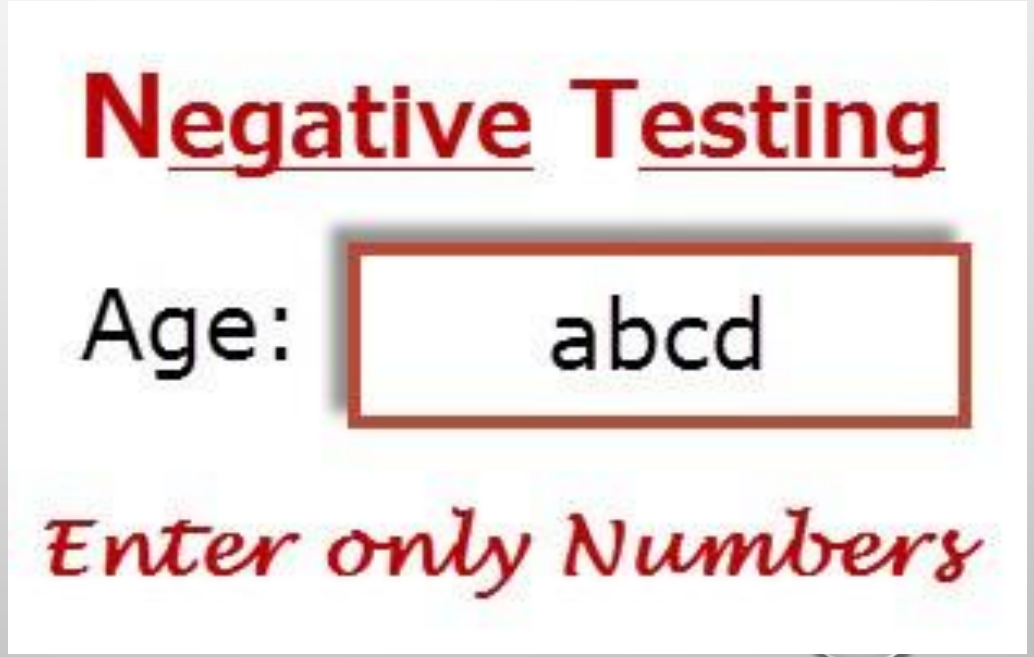
4. ACCORDING TO SCENARIO: POSITIVE AND NEGATIVE TESTING

- **POSITIVE TESTS CHECK CORRECT SCENARIOS** WITH VALID DATA
- WE WANT TO MAKE SURE THAT THE SYSTEM DOES THE THINGS IT WAS DESIGNED FOR
- POSITIVE SCENARIO – DOESN'T MEAN THE ONLY. QUANTITY OF TESTS DEPENDS ON TESTER'S QUALIFICATION

**REMEMBER THE EXAMPLE WITH THE NAME FIELD*



- **NEGATIVE TESTING –** THE OPPOSITE TO POSITIVE (SURPRISE☺️).
- IS USED TO MAKE SURE THAT THE SYSTEM DOES NOT DO THE THINGS, IT WAS NOT DESIGNED FOR.
- INVALID DATA, VALUES BEYOND THE LIMITATIONS, UNAUTHORIZED ACCESS ATTEMPTS ETC.
- NOT ONLY MAKE SURE THAT THE SYSTEM DOESN'T ACCEPT INCORRECT INPUTS BUT ALSO THAT IT REACTS ADEQUATELY: SENDS MESSAGES, ALERTS ETC.



EXAMPLES OF NEGATIVE TESTS:

- ***LEAVE REQUIRED FIELD EMPTY***
- ***USE LETTERS TO FILL NUMERIC FIELDS***
- ***WRONG INPUT INTO "CONFIRM PASSWORD" FIELD***

*GIVE YOUR EXAMPLES

GOLDEN RULE

POSITIVE TESTS BEFORE NEGATIVE

5.ACCORDING TO TEST-DOCUMENTATION: EXPLORATORY VS. SCENARIO

- **EXPLORATORY** AND **AD HOC**

  TESTING:

  TEST DESIGN AND TEST- EXECUTION AT THE SAME TIME.

  NO TEST-CASES!

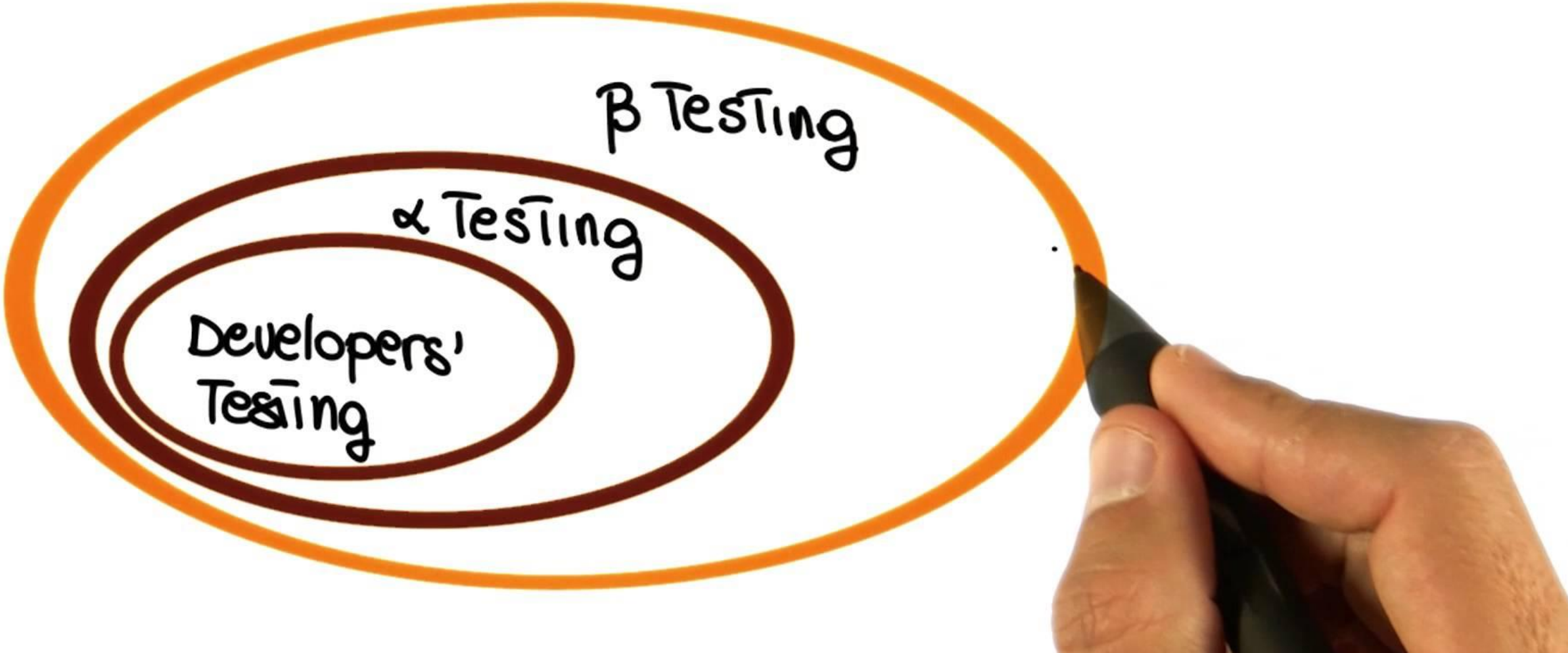
IMPROVISATION

MONKEY TESTING

- **MONKEY TESTING** - MONKEY TESTING IS A TECHNIQUE USED IN SOFTWARE TESTING TO TEST THE APPLICATION OR PRODUCT BY PERFORMING **RANDOM** ACTIONS AND PROVIDING **RANDOM** DATA AND OBSERVING IF THE SYSTEM OR APPLICATION CRASHES
- DIFFERS FROM EXPLORATORY AND AD HOC

6.ACCORDING TO TESTERS

- **ALFA** — THIS TEST IS THE FIRST STAGE OF TESTING AND WILL BE PERFORMED AMONGST THE TEAMS (DEVELOPER AND QA TEAMS). UNIT TESTING, INTEGRATION TESTING AND SYSTEM TESTING WHEN COMBINED TOGETHER IS KNOWN AS ALPHA TESTING.
- **BETA -** THIS TESTS ARE PERFORMED AFTER ALPHA TESTING HAS BEEN SUCCESSFULLY PERFORMED. IN BETA TESTING, A SAMPLE OF THE INTENDED AUDIENCE TESTS THE APPLICATION. BETA TESTING IS ALSO KNOWN AS **PRE-RELEASE TESTING**. BETA TEST VERSIONS OF SOFTWARE ARE IDEALLY DISTRIBUTED TO A WIDE AUDIENCE ON THE WEB, PARTLY TO GIVE THE PROGRAM A "REAL-WORLD" TEST AND PARTLY TO PROVIDE A PREVIEW OF THE NEXT RELEASE.


β Testing
α Testing
Developers' Testing

TIME FOR PRACTICE

- *LET'S TEST A PEN (FIVE COLORS)