

# 3. ВИДОВЕ И НИВА НА СОФТУЕРНОТО ТЕСТВАНЕ (SOFTWARE TEST TYPES AND LEVELS)

# Съдържание

- ◎ Нива на тестване (Test Levels)
  - Компонентно тестване (Component Testing)
  - Интеграционно тестване (Integration Testing)
  - Системно тестване (System Testing)
  - Тестване при предаване (Acceptance Testing)



ILLUSTRATION BY SEGUE TECHNOLOGIES

# Съдържание(2)

- ◎ Видове тестване (Test Types)
  - Тестване базирано на риска (Risk-Based Testing)
  - Функционално тестване (Functional Testing)
  - Нефункционално тестване (Non-functional Testing)
  - Структурно тестване (Structural Testing)
  - Тестване свързано с промени (Testing Related to Changes):  
Ретестване и Регресионно тестване (Re-testing and Regression Testing)
- ◎ Тестване във фаза на поддръжка (Maintenance Testing)

# Нива на тестване (Test Levels)





# Основни термини

- ◎ Компонентно тестване (Component testing)
  - Тестване на отделни компоненти на софтуера
- ◎ Софтуерни единици (компоненти) (Software units (components))
  - Модули (modules), единици (units), програми (programs), функции (functions)
  - Класове (classes) – в обектно ориентираното програмиране
- ◎ Съответните тестове се наричат:
  - Module , unit, program или class tests
- ◎ В практиката обикновено се включва програмистът, който е написал кода
- ◎ Дефектите обикновено се поправят веднага, след като се намерят
  - без официално да се регистрира намерения инцидент(дефект)

# Единици и компоненти (Units vs. Components)

- ◎ Единица (unit)
  - Най-малкия компилируем (compilable) компонент
- ◎ Компонент (component)
  - Една единица (unit) сама по себе си също е компонент
  - Интегрирането на един или повече компоненти също е един компонент

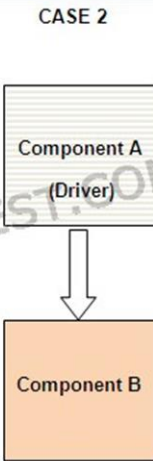
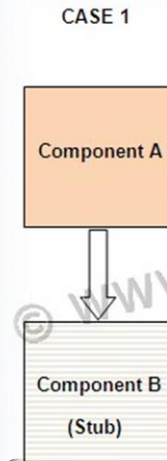
# Test Objects (Тестови обекти)

- ◎ Индивидуално тестване
  - Компонентите се тестват индивидуално
  - Изолирано от всички други софтуерни компоненти
- ◎ Изолация
  - Предотвратява външните влияния на други компоненти върху тествания компонент
- ◎ Компонентният тест проверява аспекти вътрешни за компонента
  - Не се извършва взаимодействие със съседни компоненти

# Помощници на компонентното тестване (Component Testing Helpers)

## Stubs

- При компонентното тестване извикваните тестове се заместват със stubs, simulators, или trusted components
- *Имплементация с определена цел, която е създадена с цел да се разработи или тества компонент който я извиква или разчита на нея. Замества извикван компонент*



## Drivers

- Извикващите компоненти са заменени с drivers или trusted super-components
- *Софтуерен компонент или тестови инструмент който замества компонент който отговаря за контрол и/или извикването на компонент или система*



# Интеграционно тестване (Integration Testing)



# Интеграция (Integration)

- ⦿ Съставяне на единици, за да се формират по-големи структурни единици и подсистеми
- ⦿ Извършена от програмисти (developers), тестери (QA) или специални екипи за интеграция (special integration teams)
- ⦿ Предполага, че компонентите вече са тествани индивидуално и поотделно

# Нива на интеграционно тестване (Levels of Integration Testing)

- ◎ Интеграционно тестване на компонент (Component integration testing)
  - Разкрива дефектите в интерфейсите (interfaces) и взаимодействието между интегрираните компоненти Също се нарича
    - “Integration test in the small”
- ◎ Интеграционно тестване на система (System integration testing)
  - Тестване на интеграцията на цели системи и/или пакети
  - Тестване на интерфейси за връзка с външни организации
  - Също се нарича
    - “Integration test in the large”



# Off-the-shelf Products (Готови продукти)

- ◎ Стандартни, съществуващи компоненти използвани с някаква модификация Standard, existing components used with some modification
- ◎ Обикновено не са предмет на компонентното тестване (component testing)
- ◎ Трябва да бъдат тествани за интеграция



# Защо да правим интеграционно тестване?

- ◎ След събиране (assembling) на компонентите може да възникне нова грешка (fault)
- ◎ Тестването трябва да потвърди, че всички компоненти работят и взаимодействат заедно правилно, а не само поотделно
- ◎ Главната цел- показване/откриване (exposing) на грешки
  - В интерфейсите
  - Във взаимодействието между интегрираните компоненти

# Някои типични проблеми

- ◎ Грешен формат на интерфейса (interface formats)
  - Несъвместими формати на интерфейса (Incompatible interface formats)
  - Грешен формат на файлове
- ◎ Типични грешки в обмена на данни Typical faults in data exchange
  - Грешен синтаксис или няма данни (Syntactically wrong or no data)
  - Различно тълкуване на получените данни (Different interpretation of received data)
  - Времеви проблеми (Timing problems)

# Интеграционни подходи (Integration Approaches)

- ◎ Има различни подходи за интеграционно тестване (integration testing)
  - “Big Bang” подход
    - Всички компоненти или системи са интегрирани едновременно all components or systems are integrated simultaneously
    - Главен недостатък: трудно се проследява причината за повредите
  - Инкрементален (Incremental) подход
    - Главен недостатък: поглъща много време

# Incremental Approaches (Инкрементален подход)

## ◎ Top-Down подход

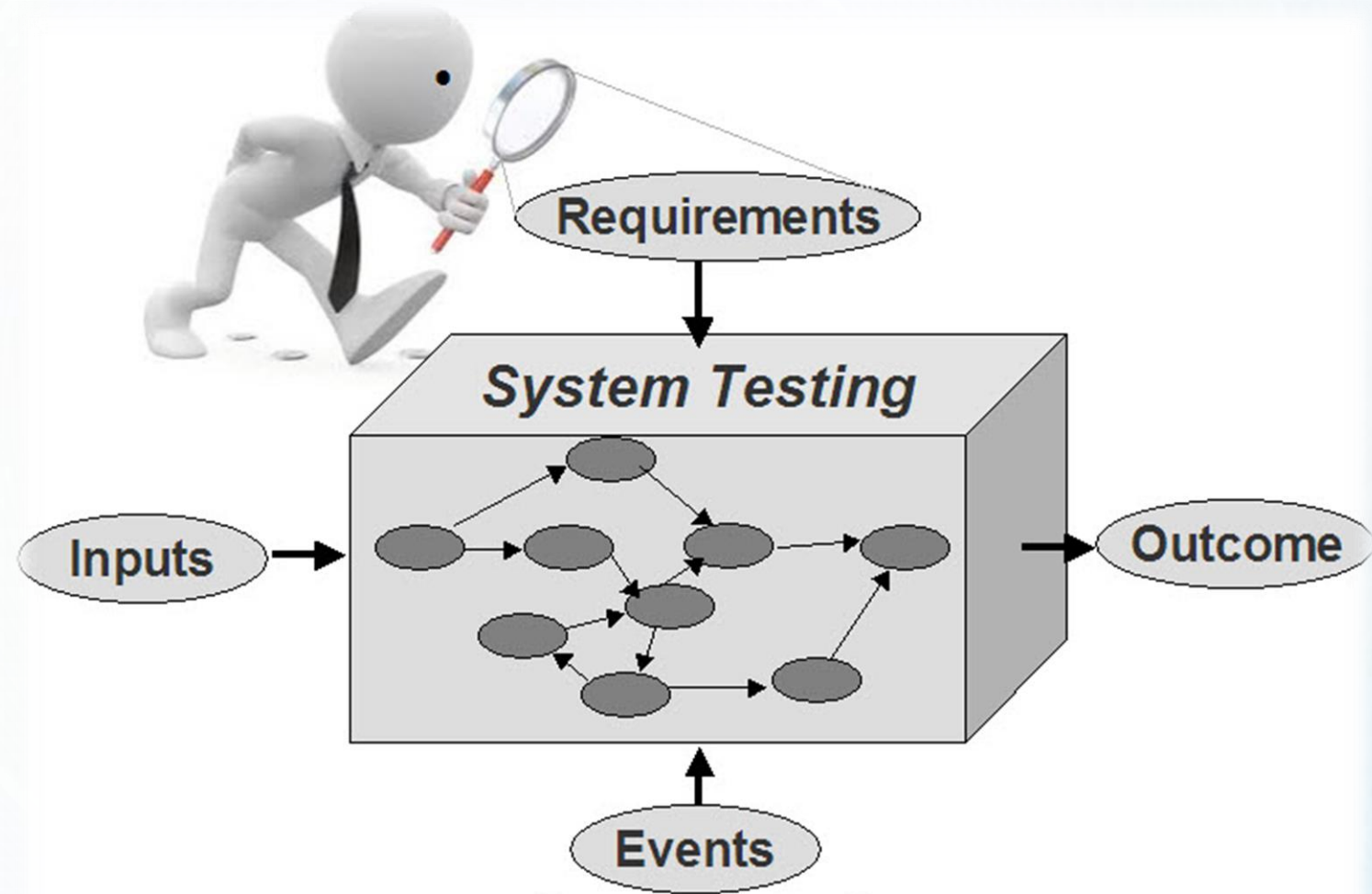
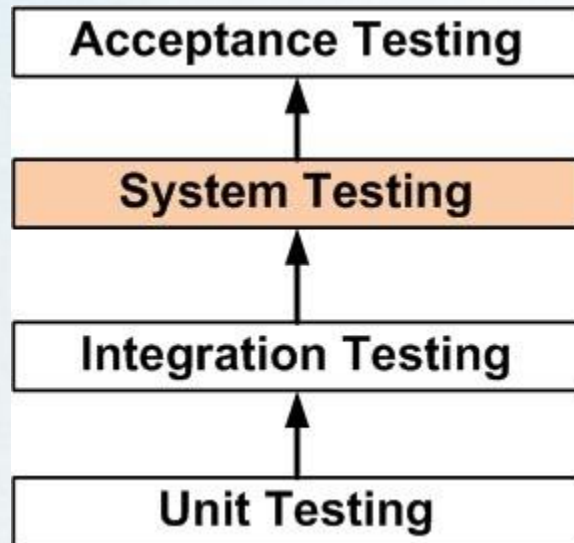
- Логиката на високо ниво и последователност се тестват първо, а компонентите на ниското ниво се тестват по-късно (The high level logic and flow are tested first - the low level components are tested later)

## ◎ Bottom-Up подход

- Противоположен на Top-Down подхода
- Главен недостатък – високото ниво на логика или най-сложните функционалности се тестват късно



# Системно тестване (System Testing)



[www.softwaretestinggenius.com](http://www.softwaretestinggenius.com)

# Защо правим системно тестване?

- ◎ Предишните тестове са били направени срещу технически спецификации (technical specifications)
- ◎ Системният тест(system test)
  - Гледа на системата от друга перспектива
    - На клиента
    - На бъдещия потребител
- ◎ Много функции и системни характеристики са резултат от взаимодействието на всички системни компоненти

# Среда на тестване (Test Environment)

- ◎ Системното тестване изисква специфична среда на тестване (test environment)
  - Хардуер (Hardware)
  - Системен софтуер (System software)
  - Драйвъри за определени устройства (Device driver software)
  - Мрежи (Networks)
  - Външни системи (External systems)

# Среда на тестване (Test Environment) (2)

- ◎ Често срещана грешка е тестване в работната среда на клиента (customer's operational environment)
  - Повредите могат да нанесат щети (damage) на системата
  - Няма контрол върху този тип среда
    - Паралелни процеси могат да повлияят (Parallel processes may influence)
    - Тестът едва ли може да бъде възпроизведен (reproduced)



# Обичайни проблеми Common Problems

- ◎ Неясни или липсващи системни изисквания
  - Липсваща спецификация на правилното поведение на системата
- ◎ Пропуснати решения
  - Не прегледани и не одобрени изисквания (Not reviewed and not approved requirements)
- ◎ Възможност за провал на проекта (Project failure possible)
  - Реализацията може да се окаже в грешната посока

# Тестване при предаване (Acceptance Testing)



# Главна идея

- ◎ Фокусът е върху изгледа и преценката на клиента
  - Особено за специфичен клиентски софтуер (customer specific software)
- ◎ Клиентът всъщност е включен в самият процес The customer is actually involved
  - Това е единственият тест, който той наистина рабира
  - Може да има главната отговорност (responsibility)
- ◎ Изпълнява се в среда подобна на клиентската (customer's like environment)
  - Възможно най-близка до оперативната целева среда
  - Може да възникнат нови проблеми

# Видове тестване при предаване (Forms of Acceptance Testing)

- ◎ Типични аспекти на тестване при предаване:
  - Верификация на изпълняването на договора (Contract fulfillment verification)
  - Тестване при предаване на потребителя (User acceptance testing)
  - Operational (acceptance) тестване
  - Тестване върху продуктова среда( Field test (alpha and beta testing) )



# Верификация на изпълняването на договора (Contract Fulfillment Verification)

- ◎ Тестване според договора (the contract)
  - Изпълнена ли е разработката/ услугата по договор
  - Софтуера без (големи) недостатъци (deficiencies) ли е?
- ◎ Критерии за предаване по време на Acceptance тест
  - Определени в договора за разработка (development contract)
  - Някакви регулации, които трябва да бъдат спазени
    - Правителствени, законови или регулации за безопасност

# User Acceptance тестване

- ⦿ Клиентът може да не е потребителя
- ⦿ Всяка потребителска група трябва да бъде включена
  - Различни потребителски групи може да имат различни очаквания
  - Отхвърляне от дори една потребителска група може да бъде проблематично



# Предварителен Acceptance тест

- ◎ Тестовете при предаване (acceptance tests) могат да бъдат изпълнени в по-ниските нива на тестване
  - По време на интеграция
    - Например- готов продукт (off-the-shelf software)
  - По време на компонентно тестване
    - За използваемост на компонент (component's usability)
  - Преди системно тестване (system testing)
    - Използвайки прототип (prototype)
    - За нова функционалност (new functionality)

# Операционен (Operational Acceptance) тест

- ◎ Приеман тест проведен от системните администратори
  - Тестване на backup/restore цикли
  - Възстановяване при провал на системата (Disaster recovery)
  - Управление на потребителите (User management)
  - Задачи по поддръжката (Maintenance tasks)
  - Уязвимости на сигурността (Security vulnerabilities)



# Тестване на продуктова среда (Field Testing)

- ◎ Един софтуер може да работи на много среди (environments)
  - Всички варианти не могат да бъдат представени в един тест
- ◎ Тестване с представителни клиенти representative customers
  - Alpha тестване
    - Провежда се от програмистите на собствена среда
    - Представителна извадка от потенциалните потребители и членове на екипи от страна на организацията, която създава системата са поканени за тест
    - Програмистите наблюдават потребителите и отбелязват проблемите
  - Beta тестване
    - Системата се изпраща на представителна извадка от потенциалните потребители, които инсталират системата и я използват в реални условия
    - Потребителите изпращат данни за инцидентите със системата на организацията която е разработила софтуера



# Резюме

**Component testing** - The testing of individual software components.

**Компонентно тестване** – Тестване на индивидуални софтуерни компоненти.

**Integration testing** - Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.

**Интеграционно тестване** – Тестване изпълнено да разкрие дефекти в интерфейсите и в взаимодействието между интегрираните компоненти или системи

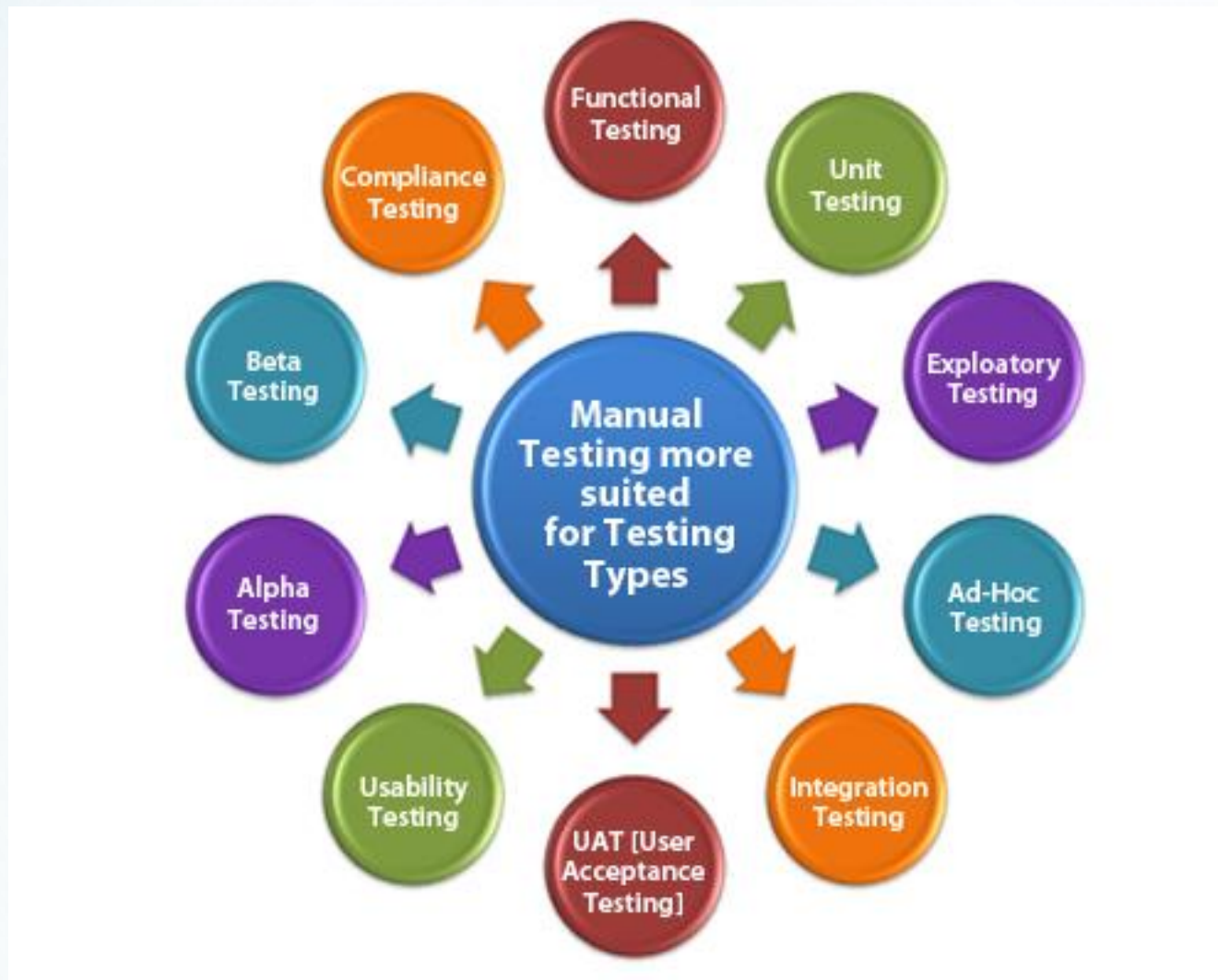
**System testing** - The process of testing an integrated system to verify that it meets specified requirements.

**Системно тестване** – Процесът на тестване една интегрирана да потвърди, че тя среща специфичните изисквания

**Acceptance testing** - Formal testing with respect to user needs, 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.

**Тестване при предаване** – Официално тестване с зачитане на потребителските нужди, изисквания и бизнес процеси проведено да определи дали или не една система задоволява критериите за предаване и да даде възможност на потребителя, клиента или друг упълномощен обект да определи дали или не да приеме системата

# Видове тестване (Test Types)





# Тестване базирано на риска (Risk-Based Testing)





# Риск

## ⦿ Риск

- Възможността за един негативен или нежелан резултат или събитие
- Всеки проблем, който може да възникне
  - Би намалил възприятията за качеството на продукта или успеха на проекта



# Видове риск

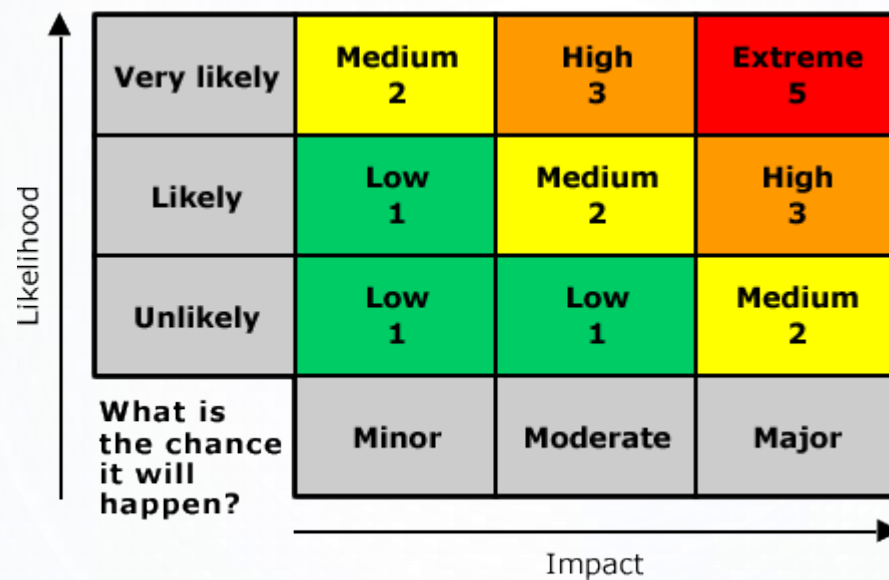
- ◎ Два главни вида риск се засягат
  - Продуктови (качествени) рискове (Product (quality) risks)
    - Първостепенният ефект на един потенциален проблем е върху продуктовото качество
  - Рискове на проекта (планиране) (Project (planning) risks)
    - Първостепенният ефект е върху успеха на проекта



# Нива на риск

- ◎ Не всички рискове са равни по важност
- ◎ Фактори за класифициране на нивото на риск:
  - Вероятност за възникване на проблема
    - Произтича от технически съображения
    - Например- използвани програмни езици, Internet връзка, и т.н.
  - Влияние на проблема, в случай че възникне
    - Произтича от бизнес съображения
    - Например- финансови загуби, брой засегнати потребители и т.н.

# Нива на риск-таблица



A risk matrix diagram with 'Likelihood' on the vertical axis and 'Impact' on the horizontal axis. The vertical axis has three levels: 'Very likely', 'Likely', and 'Unlikely'. The horizontal axis has three levels: 'Minor', 'Moderate', and 'Major'. The matrix cells are color-coded and contain risk levels and scores. The colors are: Grey for 'Very likely' and 'Unlikely' rows; Yellow for 'Medium' risk; Orange for 'High' risk; Red for 'Extreme' risk; Green for 'Low' risk. The scores are: 2 for 'Medium', 3 for 'High', 5 for 'Extreme', 1 for 'Low', and 2 for 'Medium' in the 'Unlikely' row.

Very likely	Medium 2	High 3	Extreme 5
Likely	Low 1	Medium 2	High 3
Unlikely	Low 1	Low 1	Medium 2
What is the chance it will happen?	Minor	Moderate	Major
	Impact		



# Приоритизация на усилияето (Prioritization of Effort)

- ◎ Усилието се разпределя пропорционално на нивото на риск
  - По-важните тестове се тестват първи

# Продуктови рискове:

## Какво трябва да се вземе предвид?

- ⦿ Кои функции и атрибути са критични (за успеха на продукта)?
- ⦿ Колко видим е един проблем в една функция или атрибут?
  - (За клиентите, потребителите, хората навън)
- ⦿ Колко често една функция се използва?
- ⦿ Може ли без тази функционалност?

# Функционално тестване (Functional Testing)

## FUNCTIONAL TESTING



# Функционално тестване (Functional Testing)

- Функционалното тестване проверява поведение на системата базирано на входни и изходни данни
- Използват се методи на Black box тестване
- Основите за създаването на тестове са функционалните изисквания (functional requirements)





# Функционални изисквания (Functional Requirements)

- ◎ Те определят поведението на системата
  - “Какво” системата трябва да може да прави?
  - Определете ограничения на системата

# Requirements Specifications

## (Спецификация на изискванията)

- ◎ Функционалните изисквания трябва да бъдат документирани
  - Система за управление на изискванията (Requirements management system )
  - Разисани като текст Software Requirements Specification (SRS)
- ◎ Пример:  
<http://www.slideshare.net/KrishnasaiGudavalli/software-requirements-specification-17173967>

# Тестване основано на изискванията (Requirements-based Testing)

- ◎ Изискванията се използват, като база за тестване
  - Поне един тест за всяко изискване
  - Обикновено повече от един тест е нужен, за да покрие изискванията
- ◎ Главно използвани в:
  - Системното тестване (System testing)
  - Тестването при предаване (Acceptance testing)

**Requirements-based Testing**



# Нефункционално тестване (Non-functional Testing)





# Тестване на системните атрибути Testing the System Attributes

- ⦿ „Колко добре“ или с какво качество системата трябва да изпълни нейната функция
- ⦿ Атрибутивни характеристики:
  - Надеждност (Reliability)
  - Използваемост (Usability)
  - Ефективност (Efficiency)

# Способност за тестване на изискванията

## Testability of Requirements

- ◎ Не-функционалните изисквания често не са ясно дефинирани
- ◎ Как бихте тествали:
  - “Системата трябва да бъде лесна за работа“
  - „Системата трябва да бъде бърза“
- ◎ Изискванията трябва да бъдат изразени по начин, по който да могат да се тестват
  - Уверете се, че всяко изискване може да се тества
    - Направете го рано в процеса на разработка

# Нефункционални тестове Non-functional Tests

## ⦿ Performance тест

- Скоростта за обработване и времето за отговор (Processing speed and response time)

## ⦿ Load тест

- Поведение при повишаване на системното натоварване
  - Брой на едновременните потребители
  - Брой на транзакциите

## ⦿ Stress тест

- Поведение при претоварване (overloaded)

# Нефункционални тестове (Nonfunctional Tests) (2)

- ◎ Volume тест
  - Поведението зависи от количеството на данните
- ◎ Тестване на сигурността (Testing of security)
  - Срещу нерегламентиран достъп (unauthorized access)
  - Атаки върху предлаганите софтуерни услуги (Service attacks)
- ◎ Стабилност (Stability)
  - Средно време между повредите (Mean time between failures)
  - Колко често се случват проблеми с даден потребителски профил
  - И т.н.



# Нефункционални тестове (Nonfunctional Tests) (3)

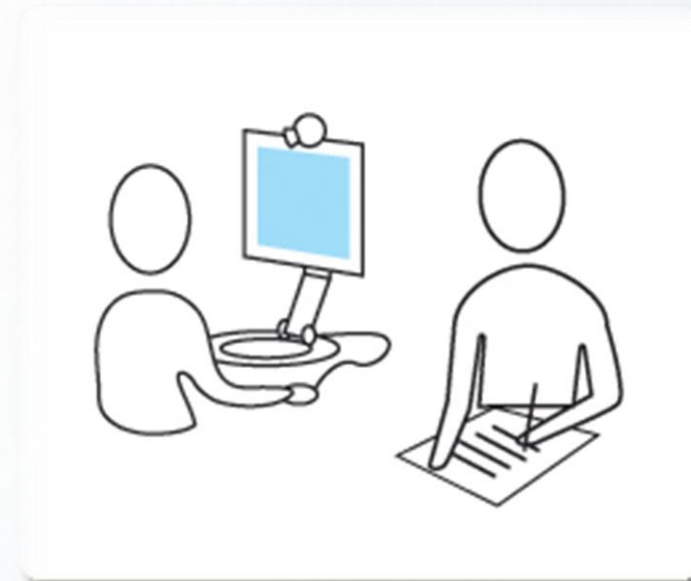
- ◎ Тестване на издръжливост (Robustness test)
  - Проверка как системата се държи при дадено изключение (exception) и възстановяване от грешки (recovery from errors)
- ◎ Съвместимост и превръщане на данни (Compatibility and data conversion)
  - Съвместимост на дадени системи (Compatibility to given systems)
  - Вход/ Изход на данни Import/export of data



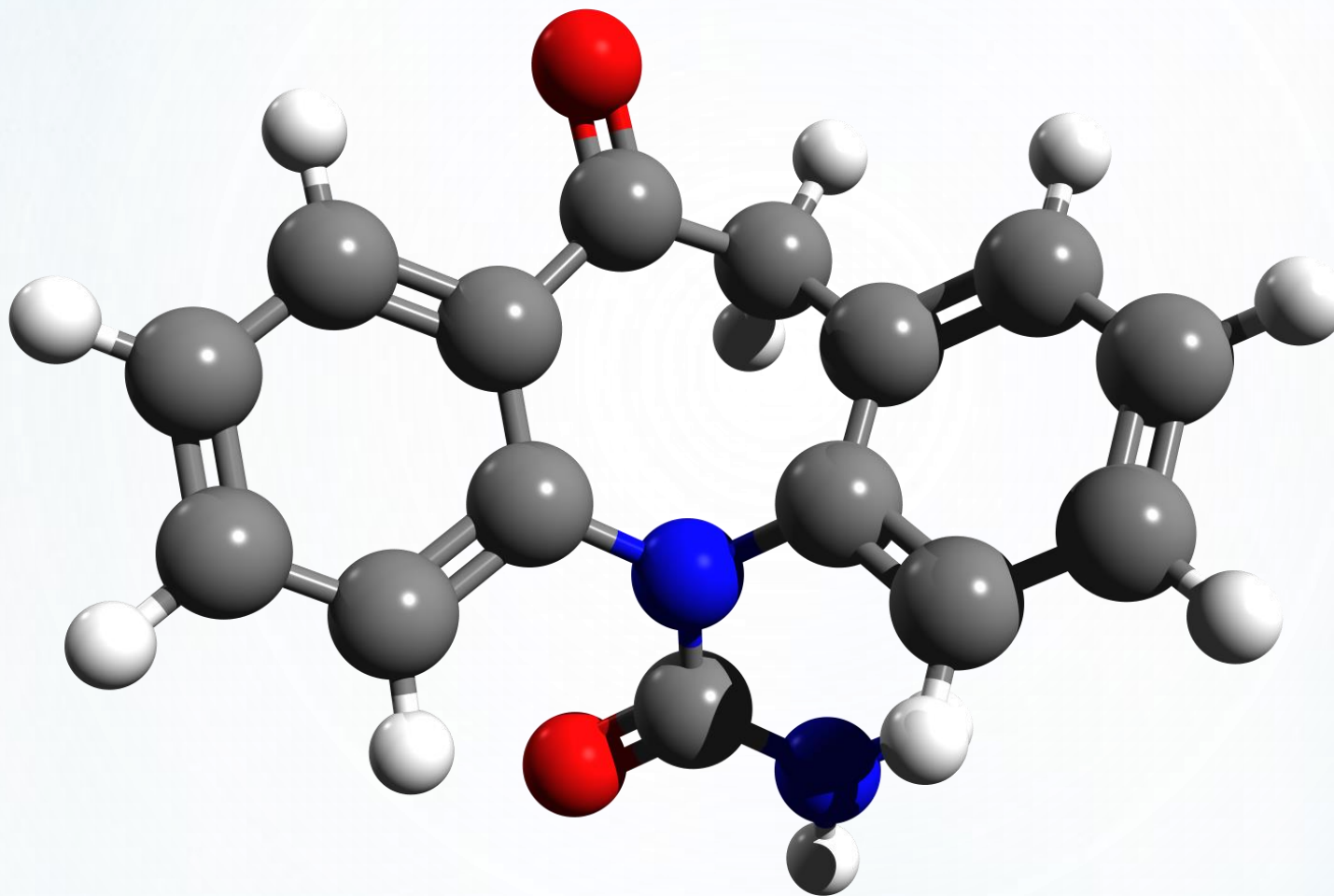
# Нефункционални тестове(4)

## ◎ Различни конфигурации на системата

- Тестване от тип (Back-to-back)
- Тест за използваемост (Usability test)
- Способността на софтуера лесно да бъде използван и разучаван
- Леснота и ефикасност на работа
- Разбираемост на системата



# Структурно тестване (Structural Testing)



# Изследване на структурата (Examining the Structure)

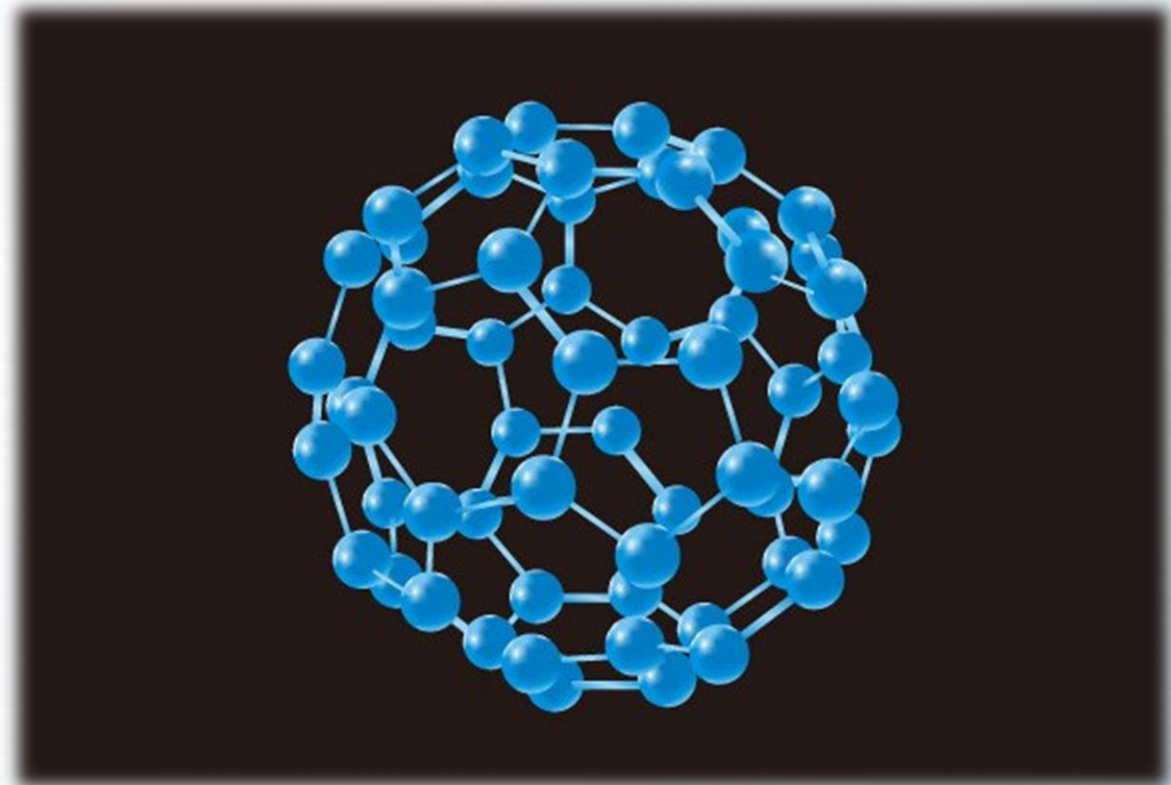
- Най често се казва 'white-box' или 'glass-box' тестване
- Използва информация за структурата на вътрешния код или архитектура
- Инструменти могат да бъдат използвани да измерят покритието на кода на елементите, такива като statements или decisions



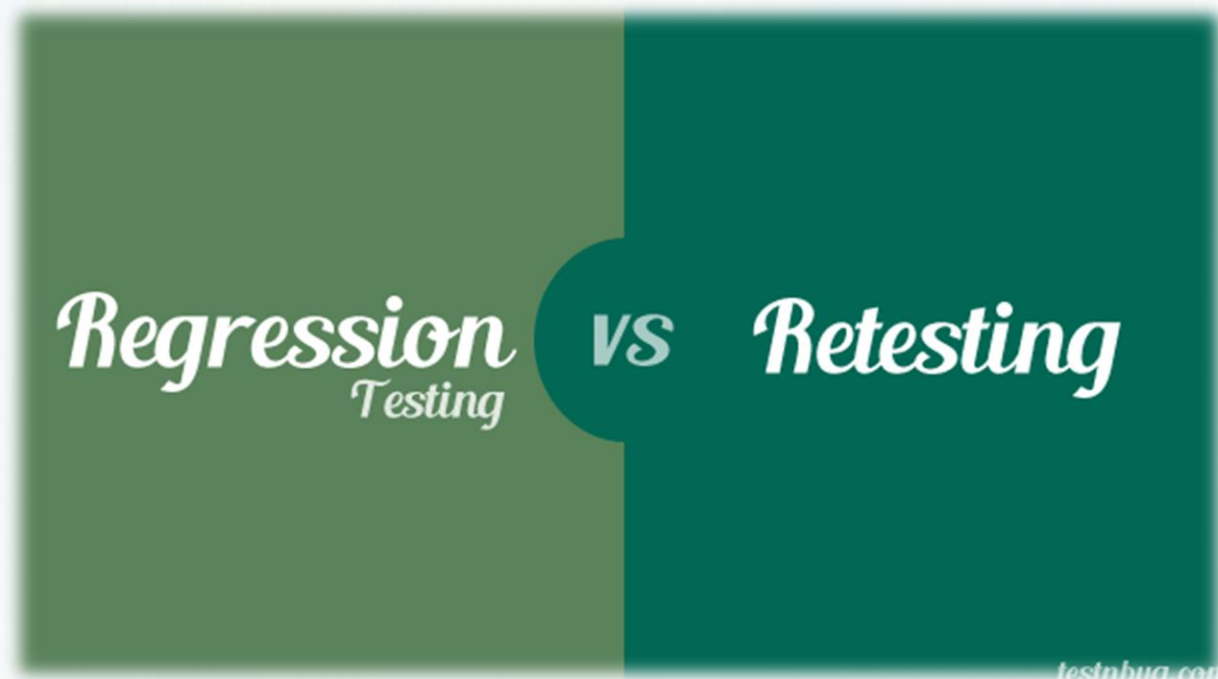


# Структурно тестване (Structure Testing Application )

- ◎ Използва се при:
  - Компонентно тестване
  - Интеграционно тестване
- ◎ Може да си използва и при:
  - Системна интеграция
  - Acceptance тестване



# Тестване свързано с промени: Повторно тестване и регресионно тестване



# Повторно тестване (Re-testing)

- ◎ След като даден дефект бива поправен, софтуера трябва да бъде изтестван повторно
  - За да се потвърди че този дефект наистина е бил премахнат
- ◎ Това се казва потвърждение (confirmation)

**RETESTING  
REQUIRED**

# Какво е регресионно тестване

- Повторно тестване на предварително тестван софтуер или програма
  - Необходимо е след модификации върху софтуера или програмата
- Тестване за това дали не са се появили нови проблеми
  - Като резултат от промените направени по системата
- Може да се използва при всички нива на тестване





# Преизползваемост на тестовете (Tests Reusability)

- ◎ Тестове използвани при регресионното тестване, които се изпълняват многократно
  - Те трябва да са добре документирани и преизползваеми
  - Често стават обект на автоматизирано тестване

# Обем на регресионното тестване

- ◎ Колко трябва да е обхвата и продължителността на регресионното тестване?
- ◎ Има няколко нива на обхват на теста:
  1. Повторно тестване свързано с дефект(Тест за потвърждаване)
    1. Повторно изпълнение на тестове, които са успели да хванат дефекти
    2. Тестване на променена функционалност
      - Проверка само на променените и коригирани части от софтуера

# Обем на регресионното тестване (2)

◎ Има няколко нива на обхват на теста:

3. Тестване на нова функционалност

- Тестване само на нови интегрирани части от програмата или софтуера

4. Пълно регресионно тестване

- Тестване на цялата система

# Неочаквани странични ефекти

- ◎ Основния проблем със софтуера
  - Сложността на самия код
- ◎ Променени или нови парчета код, които могат да променят непроменен досега код
  - Тестването само на кода, който е променен не достатъчно





# Пълно регресионно тестване

- ⦿ Единствения начин да бъдем сигурни (доколкото е възможно)
- ⦿ Промени в средата на системата
  - Необходимо е регресионно тестване
  - Може да окаже влияние във всяка част на системата
- ⦿ Прекалено скъпо и отнемащо време
  - Не е постижимо на достъпна и разумна цена
  - Необходим е анализ на щетите и влиянието им

# Тестване във фаза на поддръжка (Maintenance Testing)



# Какво поддържаме?

- ◎ Софтуера не се износва
  - Някои проблеми в дизайна вече съществуват
  - Нови дефекти чакат да бъдат открити
- ◎ Софтуерния проект не спира с първата доставка до клиента
  - Един път инсталиран и наложен, често ще се налага да бъде поддържа с години
  - Ще бъде променян, актуализиран и удължаван много пъти

# Какво поддържа? (2)

- ◎ Нови версии
  - Всеки път когато е направена корекция – нова версия бива създавана
- ◎ Тестването на промените може да се окаже трудна задача
  - Липсващи или стари системни спецификации



# Основни типове поддръжка

- ◎ Адаптивна поддръжка
  - Продукта е адаптиран към нови условия на работа
- ◎ Коригираща поддръжка
  - Премахването на забелязани дефекти



# Основни причини за поддръжка

- ⦿ Системата оперира при нови обстоятелства и условия
  - Такива, които не са планирани и прогнозирани първоначално
- ⦿ Клиентите има нови изисквания
- ⦿ Рядко срещани специални случаи
  - Които не са имплементирани в първоначалния дизайн
  - Нови методи и класове са необходими
- ⦿ Рядко срещани тотални счупвания на системата



# Тестването след поддръжка

- Всичко ново или променено трябва да бъде тествано
- Регресионното тестване е необходимо
  - Останалата част от софтуера трябва да бъде изтестван за странични ефекти
- Какво се случва ако системата не се променя?
  - Дори и само средата да е сменена пак трябва да се тества
- Обхвата е пряко свързан с риска на промяната, обема на съществуващата система и обема на промяната
- Може да се изпълнява на всеки или всички тестови нива и типове
- Основната дейност е анализ на действието и щетите (impact analysis)

# Събития които предизвикват Maintenance тестване

- ◎ Планирани промени
  - Подобряващи промени (адаптирането на софтуера според желанията на клиента, като се добавят нови функционалности и се подобрява производителността на продукта)
  - Промени в средата (планирани промени в системата или промени в базата данни)
  - Коригиращи и спешни промени
- ◎ Миграция (от една платформа на друга)
  - Трябва да включва тест дали системата може да работи адекватно на новата среда
- ◎ „Пенсиониране“ на системата
  - Може да включва тестването на миграцията на информация от старата към новата система или архивиране на информация, която ще бъде използвана за в бъдеще



# Определения

***Maintenance тестване*** – Тестване на промените върху операционната система или влиянието на промяната в средата на операционната система.

***Maintainability тестване*** – Процес на тестване за да се определи до каква степен даден продукт може да бъде поддържан.

# Links

- ◎ <http://softwaretestingfundamentals.com/differences-between-black-box-testing-and-white-box-testing/>
- ◎ <http://technologyconversations.com/2013/12/11/black-box-vs-white-box-testing/>

