

Тестване на обектно-ориентирани системи

Лекция 16

Доц. д-р Ася Стоянова-Дойчева



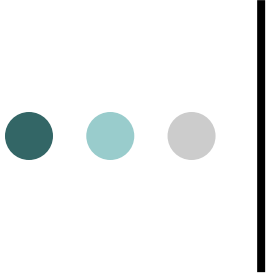
Съдържание

- Въведение
- Разширяване на виждането за тестване
- Тестване на обектно-ориентираните аналитични и проектни модели
 - Коректност
 - Консистентност
- Обектно-ориентирана стратегия за тестване
 - Unit тестове
 - Интеграционни тестове
 - Валидационни тестове
- Проектиране на тестови случаи за ОО софтуер
 - Fault-Based тестване
 - Проектиране на тестове базирани на сценарии
 - Методи за тестване приложими за класове
 - Random тестове за класове
 - Partition тестове за класове
- Проектиране на тестови случаи за вътрешни класове
 - Тестове базирани на моделите за поведение на системата
- Заключение



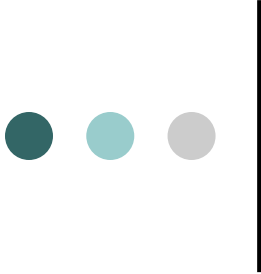
Въведение

- Въпреки, че при обектно-ориентираното тестване целта остава същата, естеството на обектно-ориентираните програми налага промяна на стратегията и подходите за тестване.
- Разширяване на виждането за тестването – аналитични и проектни модели



Тестване на обектно-ориентираните аналитични и проектни модели

- Коректност на ООА и ООП модели – синтактична и семантична
- Консистентност на ООА и ООП модели – CRC модела и модела на връзките между класовете.



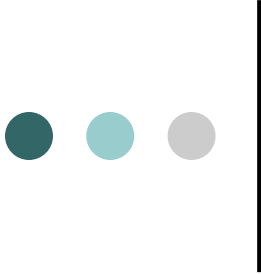
Обектно-ориентирана стратегия за тестване

- Класическата стратегия за тестване започва с “тестване в малкото” и продължава към “тестване в голямото”. т.е. :
- Започваме с unit testing
- Продължаваме с интеграционно тестване
- Завършваме с валидационни и системни тестове.



Unit тестове в контекста на ОО развой

- Каква ще е разликата в Unit тестовете при ОО развой?



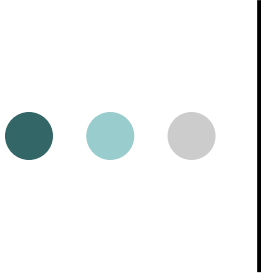
Интеграционни тестове в контекста на ОО развой

- Thread-based testing – интегрира множеството от класове, които са необходими за отговор на един вход или събитие към системата.
- Use-based testing – изграждането на системата започва чрез тестване на тези класове наречени независими класове (independent classes), които използват много малко (ако въобще има такива) сървърни класове. След тестването на независимите класове, следващият слой класове, които се тестват са зависимите класове (dependent classes), които използват тестваните вече независими класове. Последователно се тестват следващи слоеве от зависими класове докато се конструира цялата система.
- Cluster testing е една стъпка в тестването на обектно-ориентиран софтуер. При нея един клъстер от сътрудничащи си класове (определени чрез изследване на CRC и object-relationship модела) се анализира посредством тестови случаи, които се опитват да открият грешки в сътрудничеството.



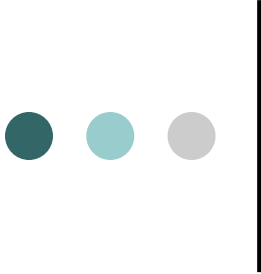
Валидационни тестове

- За подпомагане на получаването на валидационни тестове, тестващите трябва да ползва use-cases, които са част от аналитичния модел. Те доставят един сценарий, който помага за откриването на неоткрити грешки в изискванията за потребителско взаимодействие.



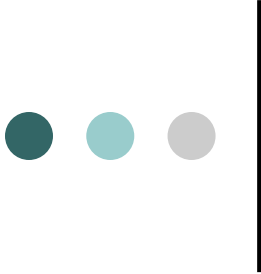
Проектиране на тестови случаи за ОО софтуер

- Всеки тестови случай трябва да бъде уникален и трябва да бъде явно свързан с класа, който ще се тества
- Трябва да бъде обявена целта на теста
- За всеки тест трябва да се създаде един списък от тестови стъпки, който трябва да съдържа:
 - Списък на специфицирани състояния за обекта, който ще се тества
 - Списък от съобщения и операции, които ще бъдат изследвани като резултат от теста
 - Списък от изключения, които могат да възникнат при тестване на обекта
 - Списък от външни условия (т.е. промени в средата, външна за софтуера, който трябва да съществува за правилното управление на теста)
 - Допълнителна информация, която ще подпомогне разбирането и изпълнението на теста.



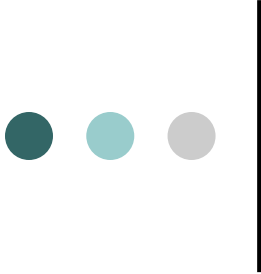
Приложение на конвенционалните методи за разработване на тестови случаи

- White-box тестове – подходящи за тестване на операциите на класовете
- Black-box тестове – use cases подходящи за разработване на тестови случаи
- Fault-Based тестове – търсят се аспекти на реализацията, където могат да възникнат грешки. За тази цел се използват аналитичните и проектни модели. Тестват се операциите и атрибутите на класовете.



Проектиране на тестове базирани на сценарии

- Некоректни спецификации – системата не прави това, което иска потребителя.
- Взаимодействие между подсистемите – поведението на една подсистема създава обстоятелства за грешки в други подсистеми.
- За създаване на тестови случаи се използват Use cases. Тестовите случаи са насочени към тестване на повече от една подсистема в един тест.



Тестване на повърхностните и вътрешните структури

- Тестване на повърхностните структури – интерфейсите се изследват за пропуснати сценарии. За създаване на тестови случаи се използват операциите и обектите.
- Тестване на вътрешните структури – тестват се механизмите на зависимостите, поведенията и комуникациите. За основа на този вид тестване се използват аналитичните и проектни модели – използват се моделите за връзка между обектите и между модулите.



Методи за тестване на класове

- Random тестване за ОО системи

Пример: Нека имаме клас X, който има следните операции: O1, O2, O3, O4, O5, O6. Всяка от тези операции може да бъде приложена за класа X, но с някакви ограничения зависещи от същността на проблема (например O2 не може да бъде изпълнена преди O1).

Минимална комбинация за тестване:

O1.O2.O4.O6

Тестването включва такива произволни вариации на последователности на операциите.



Partition тестване на класове

- Категоризират се входовете и изходите от операциите на класовете и тестовите случаи се разработват за изпълнение на всяка категория. Partition категориите се определят по следния начин:
 - State-based partitioning – категоризира операциите на класовете на основата на способността им да променят състоянието на класа.
 - Attribute-based partitioning – категоризира операциите на класа на основата на атрибутите, които използва.
 - Category-based partitioning – категоризира операциите на класовете на основата на използваните генетични функции.



Заклучение

- Общата цел на ОО тестване е да открие максимален брой грешки с минимум усилия – това е идентично с целта на конвенционалното тестване. Но стратегията за ОО тестване по смисъл се различава. Тя се разширява с тестване на аналитичните и проектни модели и фокуса се премества от процедурните компоненти при конвенционалното тестване, към класовете.