Стратегия за тестване на софтуерни системи

Лекция 15

Доц. д-р Ася Стоянова-Дойчева

Съдържание

- Въведение
- Организация на софтуерното тестване
- Стратегия за тестване на софтуер
- Критерии за приключване на тестването
- Unit testing
- Процес за провеждане на unit tests
- Интеграционни тестове
  - Top-down интеграция
  - Bottom-Up интеграция
  - Регресионни тестове
- Валидационни тестове
  - Критерии за валидационни тестове
  - Alpha и Beta тестове
- Системни тестове
  - Recovery тестове(възстановителни)
  - Security тестове(защитни)
  - Stress тестове
  - Тестове за производителността (performance)
- Заключение

Въведение

- Тестването е множество от дейности, които могат да бъдат планирани предварително и проведени систематично
- Всяка стратегия за тестване трябва да съдържа планиране на тестовете, проект на тестовите случаи, изпълними тестове и резултатни данни, които се събират и оценяват.

Организация на софтуерното тестване

- Тестването се провежда от разработчиците на софтуера
- Специален екип за тестване се формира за по-големи проекти

Стратегия за тестване на софтуер

- Единични тестове (Unit тестове) – концентрират се върху всяка единица (компонента) от софтуера
- Интеграционни тестове – концентрират се върху изграждането на софтуерната архитектура.
- Валидационни тестове – валидират се изискванията към софтуера.
- Системни тестове – софтуера и други системни елементи се тестват като цяло.

Критерии за приключване на тестването

- Кога трябва да приключим с тестването?
  - Никога не приключвате тестването, тежестта пада от вас на потребителя.
  - Приключвате тестването, когато изразходите времето си или парите за проекта
  - Използвайки статистическо моделиране и теорията за надеждност на софтуера, модела за софтуерен неуспех (непокрити грешки по време на тестването) може да се представи като функция от времето на изпълнение.

Критерии за приключване на тестването

$$f(t) = (1/p) \ln [l_0 pt + 1],$$

където

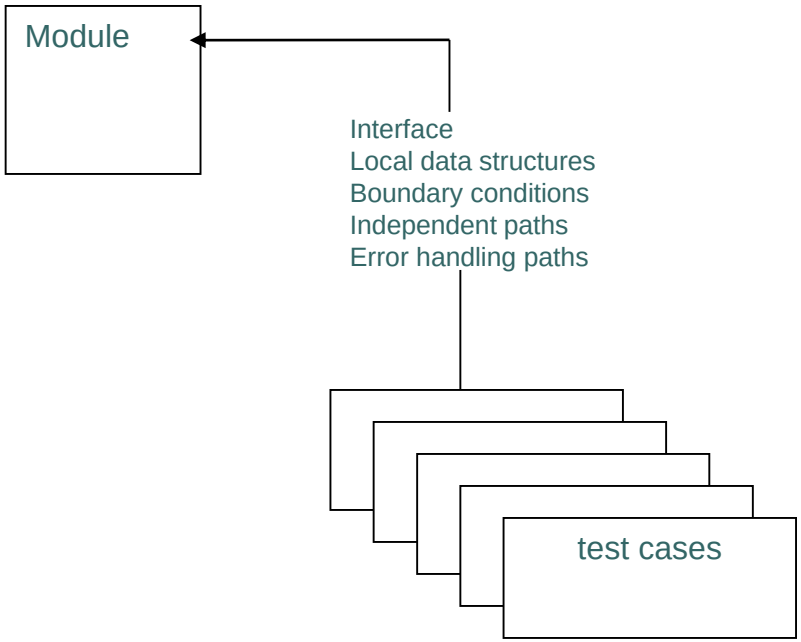
- f(t) – кумулативния (натрупващ се) брой на пропадания, които се очаква да се срещнат .
- l_0 – начален софтуерен интензитет на пропаданията (пропаданията за единица време) до започването на тестването.
- p – експоненциално намаляване на интензитета на пропаданията, като грешките са открити и поправките са направени.

Моментния интензитет на пропадане l(t) може да бъде получен като производна на f(t)

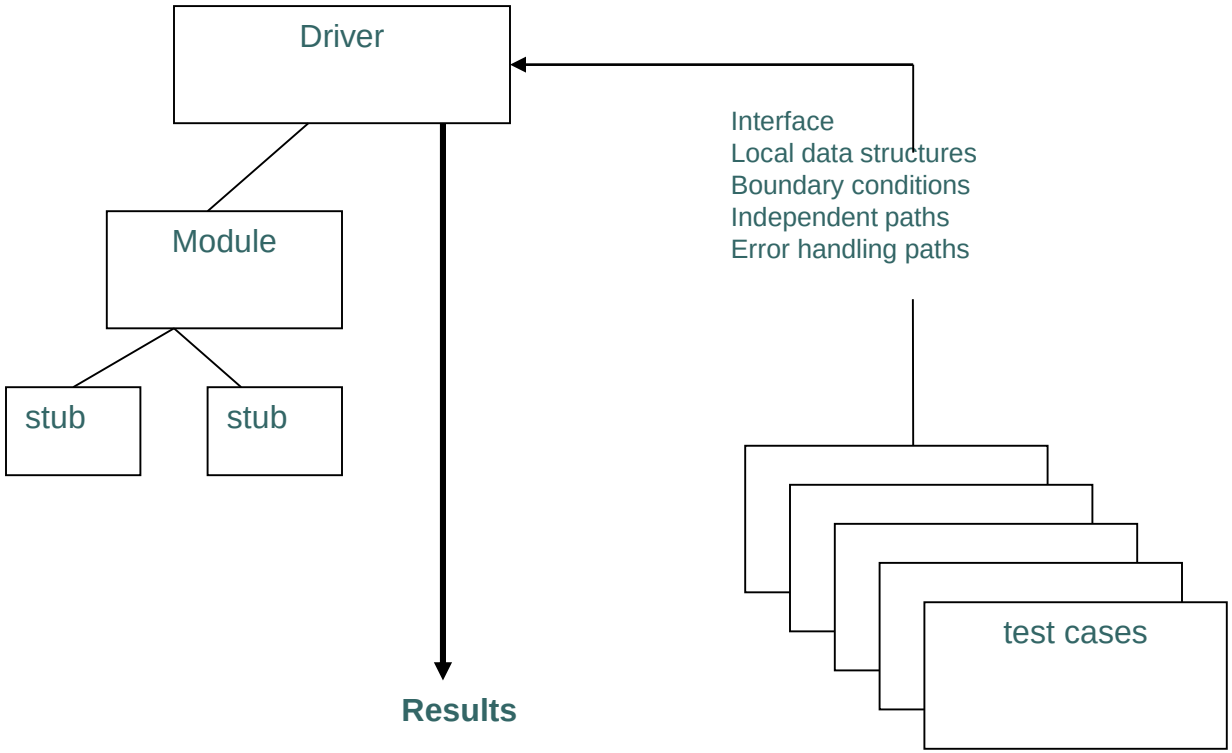
$$l(t) = l_0 / (l_0 pt + 1)$$

Unit тестове

- Unit тестове – съсредоточават се върху най-малката единица на софтуера – софтуерните компоненти или модули.



Процес на провеждане на unit тестове



Интеграционни тестове

- Целта е да се вземат unit тествани компоненти и да се изгради една програмна структура, която е наложена от развоя.

Причини за интеграционно тестване са:

- между интерфейсите могат да се губят данни;
- определен модул може да има непреднамерено нежелано въздействие върху другите модули;
- когато се комбинират подфункции (subfunctions) може да не се получи желаната главна функция (main function);
- индивидуално акцентирана неточност може да бъде усилена до неприемливо ниво;
- глобални структури от данни могат да предизвикат проблеми.

Интеграционни тестове

- Неинкрементална интеграция – “big-bang” подход
- Инкрементална интеграция
  - Top-Down интеграция
  - Bottom-Up интеграция

Top-Down интеграция

- Модулите се интегрират посредством преместване надолу през контролната йерархия, като се започне от главния контролен модул – главната програма (main control module – main program). Модулите, подчинени на главния контролен модул, се интегрират в структурата по един от двата начина:
- Първо в дълбочина (depth-first)
- Първо в широчина (breadth-first).

Top-Down интеграция

- Главният контролен модул се използва като тестови драйвер, а всички компоненти, директно подчинени на него, се заместват временно със стубове (stubs).
- В зависимост от избрания подход за интеграция заместващите стубове (subs) се заместват последователно и поотделно от съответните актуални компоненти.
- Тестовете се направляват, с интеграцията на всеки компонент.
- При завършване на всяко множество от тестове, следващ стуб (stub) се замества от реален компонент.
- Регресионното тестване може да се използва за осигуряване, че няма да се въведат нови грешки.
- Процесът продължава във втора стъпка, докато се създаде цялата програма.

Bottom-Up интеграция

- Bottom-Up integration testing започва конструирането и тестването с атомарни модули (atomic modules), т.е. компоненти на най-ниско ниво в структурата на програмата. Понеже компонентите се интегрират от долу нагоре, обработката необходима за субкоординация на компонентите на едно дадено ниво е винаги налична и не са необходими стубове(stubs)

Bottom-Up интеграция

- Компонентите на ниско ниво се комбинират в клъстери (наричани също така builds), които извършват една специфична софтуерна подфункция.
- Един драйвер (контролна програма за тестване) се създава за координиране на входа и изхода на тестовите случаи.
- Тества се клъстерът
- Драйверите се отстраняват и клъстерите се комбинират, преместваики се нагоре в програмната структура.

Регресионни тестове

- В контекста на една интеграционна стратегия за тестване regression testing е повторно изпълнение на едно подмножество от вече изпълнени тестове, целта на което е да се осигури, че промените няма да разпространят непредвидени странични ефекти.

Валидационни тестове

- Валидационните тестове установяват, дали са изпълнени валидационните критерии описани в документа “Спецификация на изискванията”.
- Критерии за валидационните тестове :
  - всички функционални изисквания са удовлетворени;
  - всички поведенчески характеристики са постигнати;
  - всички производителни изисквания са постигнати;
  - документацията е коректна;
  - изпълнени са допълнителни изисквания (преносимост, съвместимост, ...).

Валидационни тестове

- Тестове за приемане на софтуера.
- Alpha и Beta тестове.

Системни тестове

- Това са тестове, които съпровождат интегрирането на разработения софтуер в една голяма система – хардуер, други системи, персонал, информация и др.
  - Recovery тестове
  - Security тестове
  - Stress тестове
  - Performance тестове

Заключение

- Целта на тестването на софтуер е да се открият допуснатите грешки при разработката му. За да бъде постигната тази цел се планират и изпълняват серия от тестови стъпки – unit, integration, validation и system тестове.