TUGAS KAPITA SELEKTA

Ditujukan sebagai salah satu syarat

Untuk memperoleh nilai pada kegiatan matakuliah Kapita Selekta

Program Studi DIV Teknik Informatika

Oleh

Hanan Destiarin Kishendrian (1204022)

Program Studi Diploma IV Teknik Informatika

Universitas Logistik & Bisnis Internasional

BANDUNG

2023

3 Alasan Dunia Tidak Butuh Software Testing

1. Alasan 1 (waktu dan biaya pengembangan menjadi lebih lama
    - Seven Testing Principles
        - Testing shows presence of defects = tidak dapat membuktikan bahwa tidak ada cacat. pengujian harus dirancang untuk menemukan cacat sebanyak mungkin
        - Exhaustive testing is impossible = menguji semuanya (semua kondisi masukan dan prasyarat) tidak dapat dilakukan kecuali untuk kasus sepele. alih-alih pengujian menyeluruh, analisis risiko dan prioritas harus digunakan untuk memfokuskan upaya pengujian.
        - Early testing = Untuk menemukan cacat lebih awal, kegiatan pengujian harus dimulai sedini mungkin dalam siklus hidup pengembangan perangkat lunak atau sistem, dan harus terfokus atau menetapkan tujuan.
        - Defect clustering = Sejumlah kecil modul biasanya berisi sebagian besar cacat yang ditemukan selama pengujian prarilis, atau bertanggung jawab atas sebagian besar kegagalan operasional. Prinsip Pareto 80/20
        - Pesticide paradox = Jika pengujian yang sama diulang terus menerus, pada akhirnya kumpulan kasus pengujian yang sama tidak akan lagi menemukan cacat baru. Kasus pengujian perlu ditinjau dan direvisi secara berkala
        - Testing is context dependent = ID pengujian dilakukan secara berbeda dalam konteks yang berbeda. Risiko dapat menjadi faktor besar dalam menentukan jenis pengujian yang diperlukan
        - Absence-of-errors fallacy = Menemukan dan memperbaiki cacat tidak membantu jika sistem yang dibangun tidak dapat digunakan dan tidak memenuhi kebutuhan dan harapan pengguna.

      Berapa banyak pengujian yang cukup?

      Tergantung pada risiko :

      - Kehilangan kesalahan penting
      - Menimbulkan biaya kegagalan
      - Merilis perangkat lunak yang belum teruji atau kurang teruji
      - Kehilangan kredibilitas dan pasar
      - Hilang jendela pasar
      - Pengujian berlebihan, pengujian tidak efektif

Gunakan risiko untuk menentukan :

- Apa yang harus di uji terlebih dahulu
- Apa yang paling sering diuji
- Seberapa teliti untuk menguji setiap item
- Apa yang tidak boleh diuji (kali ini)
- Alokasikan waktu yang tersedia untuk pengujian dengan memprioritaskan pengujian

- Software Testing Life Cycle (STLC)
  - Test planning = melibatkan kegiatan yang meentukan tujuan pengujian dan pendekatan untuk memenuhi tujuan pengujian dalam batasan yang ditentukan oleh konteks
  - Test monitoring and control = melibatkan perbandingan kemajuan actual yang sedang berlangsung terhadap rencana pengujian menggunakan apa pun metric pemantauan pengujian yang ditentukan dalam rencana pengujian
  - Test analysis = basis pengujian dianalisis untuk mengidentifikasi fitur yang dapat diuji dan menentukan kondisi pengujian terkait
  - Test design = merancang dan memprioritaskan test case dan set test case. Mengindentifikasi data pengujian yang diperlukan untuk mendukung kondisi pengujian dari kasus.
  - Text implementation = perangkat pengujian yang diperlukan untuk pelaksanaan pengujian dibuat dan di selesaikan. Termasuk mengurutkan kasus uji ke dalam prosedur uji
  - Test execution = mencatat id dan versi item uji atau objek uji, alat uji, dan perangkat uji. Menjalankan test secara manual atau dengan menggunakan alat eksekusi test
  - Test completion = memriksa apakah semua laporan cacar telat ditutup, memasukan permintaan perubahan atau item simpanan produk untuk setiap cacat yang masih belum terselesaikan diakhir pelaksanaan pengujian

2. Alasan 2 (keyakinan berlebihan pada pengalaman proyek sebelumnya)

   Test development process

   - Analisis = tentukan kondisi pengujian
   - Desain = rancang kasus pengujian yang akan memverifikasi kondisi pengujian atau tujuan pengujian
   - Implementasi = tulis prosedur pengujian untuk menjalankan pengujian. Menulis jadwal pelaksanaan test

- Eksekusi = jalankan prosedur pengujian. Log hasil adalah log test. Laporkan setiap ketidak sesuaian kepada pengembang jika ada

Karakteristik blackbox

- Test condition, test case, dan data uji berasal dari test basis yang dapat mencakup persyaratan perangkat lunak, spesifikasi, kasus penggunaan, dan user story
- Test case dapat digunakan untuk mendeteksi kesenjangan anatara persyaratan dan implementasu persyaratan, serta penyimpangan dari persyaratan
- Cakupan diukur berdaserkan item yang dijuji dalam dasar pengujian dan teknik yang diterapkan pada daser pengujian

Structure-based or white box technique

- Pengujian berbasis struktur atau white box didasarkan pada sebuah mengidentifikasi struktur perangkat lunak atau system
- Teknik berbasis struktur melayani 2 tujuan : pengukuran cakupan uji dan desain kasus uji
- Mereka sering digunakan untuk menilai jumlah pengujian yang dilakukan dengan teknik berbasis spesifisitas bentuk turunan test, yaitu menilai cakupan. Mereka digunakan untuk merancang test tambahan dengan tujuan meningkatkan cakupan test

3. Alasan 3 (keterbatasan sumber daya yang dimiliki)
    - Menentukan Prioritas: Fokus pada pengujian pada area yang paling penting dan vital dalam perangkat lunak. Tentukan fitur atau fungsi mana yang paling mempengaruhi keseluruhan kinerja system dan focus pengujian pada area tersebut.
    - AutomatisasiPengujian: Otomatisasi pengujian perangkat lunak dapat membantu perusahaan menghemat waktu dan sumber daya yang diperlukan. Beberapa jenis pengujian, seperti pengujian regresi, dapat diotomatisasi dengan alat pengujian perangkat lunak.
    - Outsourcing Pengujian: Mempekerjakan layanan pengujian perangkat lunak pihak ketiga atau mengontrak pengujian keperusahaan yang spesialis dalam pengujian perangkat lunak dapat membantu perusahaan menghemat waktu dan sumber daya yang diperlukan
    - Menggunakan Metode Pengujian yang Efisien: Memilih metode pengujian yang efisien dan efektif dapat membantu perusahaan mengoptimalkan penggunaan sumber daya yang tersedia. Misalnya,

pengujian exploratory dapat membantu perusahaan menemukan kesalahan dengan cepat dan efisien.

- Menggunakan Tools Gratis atau Open-Source: Terdapat beberapa alat pengujian perangkat lunak gratis atau open-source yang dapat membantu perusahaan melakukan pengujian dengan biaya yang rendah.

Functional testing

- kebutuhan tentang fungsi software secara menyeluruh
- pemodelan dengan UML, ataupun penjelasan fitur-fitur dalam bentuk problem statements, adalah termasuk dalam funcyional requirement
    - diagrams : use case diagrams, activity diagrams
    - problem statements : harus mencari inventaris, harus melakukan perhitungan, harus menghasilkan laporan tertentu
- contoh : UI Test, API Test, Sanity Test

Non functional testing

- Operational – Physical/technical environment
- Performance – Speed and reliability
- Security – Who can use the system
- Cultural & Political – Company policies, legal issues
- Contoh : Scurity testing, stress test, load test

Structural testing

- Cakupan adalah sejauh mana suatu struktur telah dilaksanakan oleh pengujian, dinyatakan sebagai persentase item yang dicakup
- Jika cakupan tidak 100%, maka lebih banyak tes dapat dirancang untuk menguji item yang terlewatkan untuk meningkatkan cakupan
- Contoh : component testing, unit testing, database schema testing

Testing related to change

- Pengujian konfirmasi atau pengujian ulang, setelah cacat terdeteksi dan diperbaiki, perangkat lunak harus diuji ulang untuk memastikan bahwa cacat asli telah berhasil dihilangkan
- Pengujian regresi, adalah pengujian berulang dari program yang sudah diuji, setelah modifikasi, untuk menemukan kesalahan cacat diperkenalkan atau terungkap sebagai akibat dari perubahan
- Contoh : regression testing, confirmation testing