3 Alasan Dunia Tidak Butuh Software Testing

Tugas

Matakuliah : Kapita Selekta

Dosen Pengampu : Roni Andarsyah, ST., M.KOM.,SFPC

Oleh :

ZIAN ASTI DWIYANTI (1204049)

No. Telepon : 081224122760

Email : ziandwiasti23@gmail.com

PROGRAM STUDI DIPLOMA IV TEKNIK INFORMATIKA

UNIVERSITAS LOGISTIK DAN BISNIS INTERNASIONAL

BANDUNG

2023

3 Alasan Dunia Tidak Butuh Software Testing

A. Pemahaman 7 prinsip testing, Functional testing, Non Functional testing, Structural testing dan Testing related to Change

1. 7 Prinsip Testing

Berikut adalah 7 prinsip pengujian perangkat lunak (software testing):

- Pengujian menunjukkan keberadaan bug: Prinsip ini menyatakan bahwa pengujian perangkat lunak dirancang untuk menemukan kesalahan atau bug dalam perangkat lunak.
- Pengujian lengkap tidak mungkin dilakukan: Prinsip ini menyatakan bahwa tidak mungkin untuk melakukan pengujian yang lengkap pada perangkat lunak.
- Early Testing: Prinsip ini menyatakan bahwa pengujian perangkat lunak harus dimulai sejak awal dalam siklus pengembangan perangkat lunak.
- Defect clustering: Prinsip ini menyatakan bahwa sebagian besar kesalahan dalam perangkat lunak terjadi pada area tertentu atau "cluster" dalam perangkat lunak.
- Pesticide Paradox: Prinsip ini menyatakan bahwa jika pengujian dilakukan dengan pengujian yang sama berulang-ulang, maka pengujian tersebut tidak efektif dalam menemukan bug yang baru.
- Testing is context dependent: Prinsip ini menyatakan bahwa pengujian perangkat lunak harus disesuaikan dengan konteks perangkat lunak dan kebutuhan pengguna.
- Absence-of-errors fallacy: Prinsip ini menyatakan bahwa menemukan sedikit atau tidak ada kesalahan dalam pengujian tidak menjamin bahwa perangkat lunak bebas dari kesalahan.

Prinsip-prinsip ini merupakan panduan penting untuk pengujian perangkat lunak yang efektif dan membantu dalam menemukan kesalahan dan memastikan bahwa perangkat lunak yang dihasilkan berkualitas baik.

2. Functional Testing

Functional testing adalah jenis pengujian perangkat lunak yang memeriksa perilaku dan fungsionalitas suatu aplikasi, sistem, atau komponen berdasarkan persyaratan atau spesifikasi yang telah ditentukan sebelumnya. Tujuan utama dari pengujian fungsional adalah untuk memastikan bahwa aplikasi atau sistem berfungsi seperti yang diinginkan dan memenuhi kebutuhan pengguna.

Pengujian fungsional umumnya melibatkan black-box testing, yang berarti bahwa pengujian tidak memiliki akses ke kode internal atau arsitektur aplikasi. Sebaliknya, pengujian difokuskan pada masukan dan keluaran sistem atau aplikasi, memverifikasi bahwa sistem berperilaku dengan benar ketika diberikan masukan tertentu dan menghasilkan keluaran yang diharapkan.

Pengujian fungsional dapat dilakukan secara manual atau menggunakan alat pengujian otomatis. Dalam pengujian manual, pengujian mengikuti serangkaian kasus pengujian yang telah ditentukan sebelumnya untuk memeriksa perilaku aplikasi. Dalam pengujian otomatis, pengujian menggunakan alat perangkat lunak untuk membuat dan menjalankan kasus pengujian secara otomatis.

Pengujian fungsional dapat dibagi menjadi beberapa kategori, seperti pengujian unit, pengujian integrasi, pengujian sistem, dan pengujian penerimaan. Setiap kategori ini berfokus pada level sistem atau aplikasi yang berbeda dan memiliki tujuan dan metode pengujian tersendiri.

Secara keseluruhan, pengujian fungsional merupakan bagian yang sangat penting dari proses pengembangan perangkat lunak, karena membantu memastikan bahwa aplikasi atau sistem memenuhi kebutuhan pengguna dan berfungsi seperti yang diinginkan.

3. Non Functional Testing

Non-functional testing adalah jenis pengujian perangkat lunak yang dilakukan untuk mengevaluasi bagaimana suatu sistem bekerja dalam kondisi tertentu yang tidak terkait dengan fungsionalitas utamanya. Fokusnya adalah pada performa, keandalan, kegunaan, keamanan, dan aspek lain dari sistem perangkat lunak yang tidak langsung terkait dengan fungsinya.

4. Structural Testing

Structural testing (pengujian struktural) adalah teknik pengujian perangkat lunak yang dilakukan dengan memeriksa struktur internal kode program. Tujuan dari pengujian struktural adalah untuk memastikan bahwa setiap bagian dari kode program telah diuji secara cermat dan efektif.

Pengujian struktural melibatkan analisis kode program dan pelaksanaan tes unit pada level kode. Tes unit adalah tes yang dilakukan pada bagian-bagian kecil kode program yang disebut unit, seperti fungsi atau prosedur. Dalam pengujian struktural,

tes unit dilakukan dengan memeriksa setiap jalur yang mungkin dilalui oleh kode program, sehingga memastikan bahwa setiap instruksi kode telah diuji.

5. Testing Related to Change

Testing related to change (pengujian terkait perubahan) adalah proses pengujian perangkat lunak yang dilakukan untuk memverifikasi perubahan pada perangkat lunak setelah dilakukan modifikasi atau pembaruan pada kode program. Tujuannya adalah untuk memastikan bahwa perubahan yang dilakukan tidak memengaruhi kinerja perangkat lunak secara keseluruhan dan tidak memunculkan kesalahan baru.

Proses pengujian terkait perubahan melibatkan beberapa langkah, antara lain:

- Analisis dampak: Langkah ini dilakukan untuk mengevaluasi dampak dari perubahan yang dilakukan pada perangkat lunak. Hal ini dapat dilakukan dengan menganalisis daftar perubahan dan menentukan bagian-bagian yang terkait dengan perubahan tersebut.
- Regresi testing: Pengujian regresi dilakukan untuk memastikan bahwa perubahan yang dilakukan tidak memengaruhi kinerja fungsi-fungsi yang telah diuji sebelumnya. Tes ini melibatkan pengulangan tes yang telah dilakukan pada versi sebelumnya dari perangkat lunak dan membandingkannya dengan hasil tes pada versi yang baru.
- Pengujian integrasi: Pengujian integrasi dilakukan untuk memastikan bahwa perubahan yang dilakukan pada satu bagian perangkat lunak tidak memengaruhi bagian lain dari perangkat lunak. Hal ini dilakukan dengan menguji bagian yang terkena dampak dan bagian-bagian yang terkait.
- Pengujian fungsional: Pengujian fungsional dilakukan untuk memastikan bahwa fungsi-fungsi utama dari perangkat lunak masih berfungsi dengan baik setelah dilakukan perubahan.
- Pengujian performa: Pengujian performa dilakukan untuk memastikan bahwa perubahan yang dilakukan tidak memengaruhi performa keseluruhan dari perangkat lunak.

Dalam pengujian terkait perubahan, penting untuk melakukan pengujian dengan cermat untuk memastikan bahwa perubahan yang dilakukan tidak memunculkan kesalahan baru atau memengaruhi kinerja perangkat lunak secara keseluruhan. Hal ini dapat

membantu memastikan bahwa perangkat lunak dapat berfungsi secara efektif setelah dilakukan perubahan.

B. Contoh Functional testing, Non Functional testing, Structural testing dan Testing related to Change

1. Contoh Functional Testing

- Pengujian interaksi antar komponen:

  Pengujian ini bertujuan untuk memastikan bahwa komponen-komponen yang berinteraksi dalam sistem berjalan dengan baik dan tidak saling mempengaruhi. Misalnya, jika sistem memiliki modul pembayaran dan modul pengiriman produk, maka pengujian interaksi antar komponen akan memastikan bahwa kedua modul tersebut dapat berinteraksi dengan baik.

- Pengujian integrasi:

  Pengujian integrasi bertujuan untuk memastikan bahwa semua komponen dalam sistem dapat berinteraksi dengan baik dan menghasilkan hasil yang diharapkan. Misalnya, jika sistem terdiri dari beberapa modul, maka pengujian integrasi akan memastikan bahwa semua modul dapat bekerja secara harmonis.

- Pengujian keamanan:

  Pengujian keamanan bertujuan untuk memastikan bahwa perangkat lunak terlindungi dari serangan dan dapat menjaga kerahasiaan dan integritas data. Misalnya, pengujian keamanan dapat melibatkan pengujian penetrasi untuk menemukan kerentanan pada sistem atau pengujian keamanan fungsional untuk memastikan bahwa sistem dapat mencegah akses yang tidak sah.

2. Contoh Non Functional Testing

- Pengujian Ketersediaan:

  Pengujian ketersediaan bertujuan untuk memastikan bahwa sistem dapat beroperasi secara konsisten dan tersedia setiap saat. Contoh pengujian ketersediaan adalah pengujian pemulihan bencana (disaster recovery testing) dan pengujian ketersediaan (availability testing).

- Pengujian Kehandalan:

  Pengujian kehandalan bertujuan untuk memastikan bahwa aplikasi dapat bekerja dengan baik dan tidak mengalami kegagalan yang sering. Contoh pengujian

kehandalan adalah pengujian pemulihan kesalahan (fault recovery testing) dan pengujian toleransi kesalahan (error tolerance testing).

- Pengujian Usability:

  Pengujian usability bertujuan untuk memastikan bahwa aplikasi dapat digunakan dengan mudah oleh pengguna dan memberikan pengalaman pengguna yang baik. Contoh pengujian usability adalah pengujian antarmuka pengguna (user interface testing), pengujian pengalaman pengguna (user experience testing), dan pengujian aksesibilitas (accessibility testing).

3. Contoh Structural Testing

- Pengujian Jalur:

  Pengujian jalur adalah teknik pengujian struktural yang digunakan untuk memastikan bahwa setiap jalur atau percabangan dalam kode telah diuji dengan benar. Contoh pengujian jalur adalah pengujian path coverage, branch coverage, dan statement coverage.

- Pengujian Mutasi:

  Pengujian mutasi adalah teknik pengujian struktural yang digunakan untuk menguji keandalan kode dengan memodifikasi kode asli dan melihat apakah tes masih berjalan dengan baik. Contoh pengujian mutasi adalah mengubah operator logika dari && menjadi || untuk melihat apakah tes masih berhasil.

4. Contoh Testing Related to Change

- Pengujian Regresi:

  Pengujian regresi dilakukan untuk memastikan bahwa perubahan pada kode tidak memengaruhi fungsi yang sudah diuji sebelumnya. Ini dilakukan dengan menjalankan kembali tes yang sudah ada setelah dilakukan perubahan pada kode.

- Pengujian Integrasi:

  Pengujian integrasi juga dapat dilakukan untuk memastikan bahwa perubahan pada satu bagian kode tidak mempengaruhi interaksi antara bagian kode lainnya.

- Pengujian Kinerja:

  Pengujian kinerja dilakukan untuk memastikan bahwa perubahan yang dilakukan tidak memengaruhi kinerja perangkat lunak secara negatif. Hal ini dilakukan

dengan menguji performa perangkat lunak sebelum dan setelah perubahan dilakukan.

C. Intisari dari Pelatihan

Kegagalan Project Software :

- Dibatalkan sebelum selesai
- Selesai tapi tidak pernah dipakai
- Tidak bermanfaat bagi pengguna
- Tidak sesuai dengan keinginan pengguna

Stages of Testing

1. Unit testing

   Tipe unit testing : Black Box testing dan White Box testing.
2. Integration testing

   Tipe integration testing : User interface testing, use-case testing, interaction testing, dan system interface testing.
3. System testing

   Tipe system testing : Requirements testing, usability testing, security testing, performance testing, dan documentation testing.
4. Acceptance testing

   Tipe Acceptance testing : alpha testing dan beta testing

3 Alasan Dunia Tidak Butuh Software Testing :

1. Waktu dan Biaya Pengembangan Menjadi Lebih Lama
2. Keyakinan Berlebihan pada pengalaman proyek sebelumnya
3. Keterbatasan sumber daya yang dimiliki.