TUGAS

RANGKUMAN MEET BRAINMATRICS PART II

Mata Kuliah : Kapita Selekta

Dosen Pengampu : Roni Andarsyah, S.T., M.Kom., SFPC

Oleh:

Muhammad Ilman Aqilaa – 1204055

PROGRAM STUDI DIPLOMA IV TEKNIK INFORMATIKA

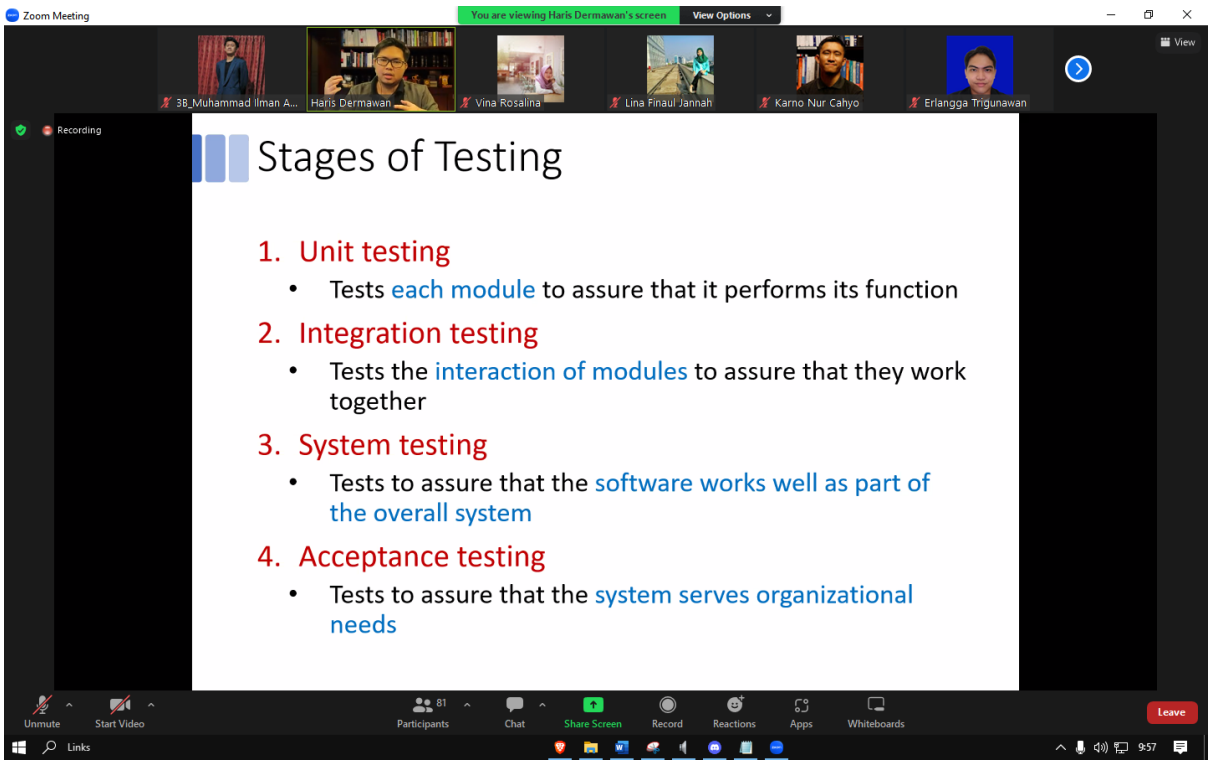
UNIVERSITAS LOGISTIK DAN BISNIS INTERNASIONAL

BANDUNG

2023

RANGKUMAN BRAINMATRICS

Bukti Kehadiran



BRAINMATRICSID LEARNING MANAGEMENT SYSTEM

- **Pemateri**

  Wahyu Utomo

- **Judul Materi**

  Haris Dermawan – Software Engineering

- **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 dimiki.

- **Kegagalan dalam project software**

  Project teknologi informasi gagal dengan presentase 50% dikarenakan:

  - Dibatalkan sebelum selesai;
  - Aplikasi sudah selesai namun tidak pernah digunakan;
  - Tidak memiliki manfaat bagi pengguna
  - Tidak sesuai dengan request client

- **Keunikan dari software**

  Kompleksitas:

Dalam Software = Buatkan saya beberapa pertanyaan dengan tema "Alasan dibutuhkannya software testing

Dalam Hardware = Tingkat kompleksitas produk lain rendah, dengan kemungkinan perubahan parameter dan fungsi tidak beragam

Sumber Daya Manusia:

Dalam Software = Kuantitas SDM tidak berhubungan dengan kualitas dengan kecepatan kerja.

Dalam Hardware = Kuantitas SDM berhubungan dengan kualitas dan kecepatan kerja

Visibilitas Produk:

Dalam Software = Produk tidak terlihat dengan kasat mata, termasuk bila ada cacat dari produk

Dalam Hardware = Produk terlihat dengan kasat mata, termasuk bila ada cacat dari produk

- **Software Errors vs Faults vs Failures**

  Software error:

  Kesalahan tata bahasa dalam baris kode;

  Kesalahan logis dalam menjalankan kebutuhan klien;

  Dari software error menyebabkan software faults:

  Fungsi perngkat lunak yang tidak benar melaksanakan umum atau khusus aplikasi.

  Dari software faults menyebabkan software failures:

  Terjadi ketika kesalahan perangkat lunak diaktifkan.

- **Early testing**

  Untuk menemukan cacat lebih awal, kegiatan pengujian harus dilakukan dimulai sedini mungkin dalam perangkat lunak atau siklus hidup pengembangan sistem, dan harus tujuan yang terfokus atau terdefinisi

  Proses testing:

  1. Unit Testing

     Menguji interaksi modul untuk memastikan modul berfungsi bersama.

     Tipe unit testing:

     - Black Box Testing

       Paling umum dan sering digunakan dengan hanya melakukan input output. Menguji apakah unit memenuhi syarat yang dinyatakan dalam spesifikasi.

- White-Box Testing

  Melihat modul untuk menguji elemen utamanya, kegunaan terbatas dalam desain karena unitnya sangat kecil.

2. Integration testing

   Menguji interaksi modul untuk memastikan modul berfungsi bersama.

   Empat jenis pengujian Integrasi:

   1. Pengujian antarmuka pengguna
      - Menguji setiap fungsi antarmuka
      - Menelusuri setiap menu/layar
   2. Pengujian kasus penggunaan
      - Memastikan bahwa setiap kasus penggunaan bekerja dengan benar
      - Telusuri setiap kasus penggunaan
   3. Pengujian interaksi
      - Mulailah dengan sebuah paket
      - Setiap metode adalah sebuah rintisan
      - Tambahkan metode satu per satu, uji sambil jalan
      - Setelah semua paket selesai, ulangi pada level paket
   4. Pengujian antarmuka sistem
      - Memastikan transfer data antar sistem

3. System testing

   Pengujian untuk memastikan bahwa perangkat lunak beerja dengan baik sebagai bagian dari sistem keseluruhan.

   Lima jenis pengujian sistem:

   1. Pengujian Persyaratan
      - Apakah persyaratan bisnis terpenuhi?
      - Memastikan bahwa integrasi tidak menimbulkan kesalahan baru
   2. Pengujian Kegunaan
      - Menguji seberapa mudah dan bebas kesalahan sistem yang digunakan
      - Informal atau formal
   3. Pengujian Keamanan
      - Memastikan bahwa fungsi keamanan ditangani dengan baik.
   4. Pengujian Kinerja

- Memastikan bahwa sistem bekerja di bawah volume aktivitas yang tinggi

5. Pengujian Dokumentasi
    - Analis memeriksa bahwa dokumentasi dan contoh berfungsi dengan baik

4. Acceptance testing

   Pengujian untuk memastikan bahwa sistem melayani kebutuhan organisasi. Yang dilakukan oleh pengguna dengan dukungan dari tim proyek untuk memastikan sistem memenuhi persyaratan.

   Dua jenis pengujian penerimaan:

   1. Pengujian Alfa
      - Ulangi tes oleh pengguna untuk memastikan mereka menerima sistem, menggunakan data yang diketahui.
   2. Pengujian Beta
      - Menggunakan data nyata, bukan data uji.

- **Software yang berkualitas**
    1. Sejauh mana suatu sistem, komponen, sesuai kebutuhan.
    2. Sejauh mana suatu sistem, komponen, atay proses memenuhi pelanggan.
- **Alasan 1 Waktu dan Biaya pengembangan menjadi lebih lama**
- **Tujuh prinsip pengujian:**
    1. Pengujian untuk mencari kecacatan software.

       Pengujian harus dirancang untuk menemukan sebanyak mungkin.
    2. Pengujian lengkap sangat tidak mungkin

       Menguji semuanya (semua kombinasi input dan prasyarat) tidak dapat dilakukan kecuali untuk kasus sepele.

       Alih-alih pengujian menyeluruh, analisis risiko dan prioritas harus digunakan untuk memfokuskan upaya pengujian.
    3. Early testing

       Dalam Software Development Life Cycle (SDLC) kegiatan pengujian harus dimulai sedini mungkin dan harus difokuskan pada tujuan yang telah ditetapkan.
    4. Pengelompokkan kecacatan

Sejumlah kecil modul biasanya berisi sebagian besar cacat yang ditemukan selama pengujian prarilis, atau bertanggung jawab atas sebagian besar kegagalan operasional.

5. Pesticide Paradox

   Jika tes yang sama diulang terus-menerus sekali lagi, pada akhirnya kumpulan kasus uji yang sama akan melakukannya tidak lagi menemukan cacat baru.

   Uji kasus perlu ditinjau secara teratur dan diperbaiki

6. Testing is Context Dependent

   Menguji id dilakukan secara berbeda dalam konteks yang berbeda Risiko bisa menjadi faktor besar dalam menentukan jenis pengujian yang diperlukan.

7. Absence-of-errors-fallacy

   Menemukan dan memperbaiki cacat tidak membantu jika sistem yang dibangun tidak dapat digunakan dan tidak memenuhi kebutuhan dan harapan pengguna.

- **Functional Testing**

  Kebutuhan tentang fungsi software secara menyeluruh

  Pemodelan dengan UML, ataupun penjelasan fitur-fitur dalam bentuk problem statements.

  Contoh:

  1. Pengujian unit: pengujian terhadap unit kode program individu untuk memastikan bahwa masing-masing unit bekerja dengan benar.
  2. Pengujian integrasi: pengujian untuk memastikan bahwa berbagai komponen perangkat lunak yang berbeda dapat terintegrasi dengan baik dan bekerja secara bersama-sama.
  3. Pengujian sistem: pengujian untuk memastikan bahwa seluruh sistem atau aplikasi bekerja dengan baik dan memenuhi persyaratan fungsional.

- **Non Functional Testing**

  Merupakan jenis pengujian softwarte yang memiliki tujuan untuk evaluasi karakteristik non-fungsional dari perangkat lunak, seperti kinerja, keamanan, skalabilitas, keandalan, dan kompabilitias.

  Contoh:

  - Pengujian keamanan untuk mengevaluasi tingkat keamanan perangkat lunak dari serangan atau ancaman yang berpotensi.

- Pengujian kinerja untuk memastikan bahwa perangkat lunak bekerja dengan cepat dan responsif pada tugas tertentu.

- **Structural Testing**

  Structural testing adalah jenis pengujian perangkat lunak yang dilakukan untuk mengevaluasi kualitas struktur internal atau kode program dari suatu aplikasi. Tujuannya adalah untuk memastikan bahwa kode program telah dikembangkan dengan baik dan sesuai dengan standar kualitas yang ditetapkan.

  Contoh:

  1. Pengujian Cakupan Kode (Code Coverage Testing): Teknik pengujian yang digunakan untuk memastikan bahwa semua bagian kode program telah diuji dengan baik. Ada beberapa jenis cakupan kode, seperti cakupan baris, cakupan cabang, dan cakupan kondisi.

- **Testing Related to Change**

  Testing Related to Change adalah pengujian yang dilakukan untuk memastikan bahwa perangkat lunak masih berfungsi dengan baik setelah dilakukan perubahan pada kode atau lingkungan sistem.

  Contoh:

  1. Regression Testing: pengujian ini dilakukan setelah perubahan kode atau lingkungan sistem, untuk memastikan bahwa fitur yang telah diuji sebelumnya masih berfungsi dengan baik setelah perubahan.

- **Berapa banyak pengujian dikatakan cukup?**

  - **Tergantung pada risiko:**

  Kehilangan kesalahan penting

  Menimbulkan biaya kegagalan

  Merilis perangkat lunak yang belum teruji atau kurang teruji

  Kehilangan kredibilitas dan pangsa pasar

  Kehilangan jendela pasar

  Pengujian berlebihan, pengujian tidak efektif

  - **Gunakan risiko untuk menentukan:**

  Apa yang harus diuji terlebih dahulu

  Apa yang paling sering diuji

  Seberapa teliti menguji setiap item

  Apa yang tidak boleh diuji (kali ini)

Alokasikan waktu yang tersedia untuk pengujian oleh memprioritaskan pengujian

- **Test Process**

  Software testing life cycle

  1. Test planning
  2. Test monitoring and control
  3. Test analysis
  4. Test design
  5. Test implementation
  6. Test execution test completion

- **Siklus Pengembangan Software:**

  1. User yang sebagai product owner melakukan request software ke system analyst.
  2. System analyst membuatkan analisis kelayakan dari sistem request tersebut
  3. Jika layak, sistem analis melakukan analisis dan desain terhadap sistem yang akan dibuat lalu menghasilkan sistem spesification. Dimana sistem analis dibantu oleh bisnis anlalis dalam memahami proses bisnis dari software yang akan dibangun.
  4. Spefikasi sistem akan diserahkan ke programmer untuk dilakukan pembangunan (Coding)
  5. Hasil Codingan berupa kode program akan diserahakan oleh software tester untuk dilakukan pengujian (Unit, Integration. System, User Acceptance Testing)
  6. Instalasi software dan manajemen perubahan (software yaitu kode program dengan dokumentasi)
  7. Siklus kembali pada proses 1 jika ada request perubahan atau permintaan perubahan software

- **ALASAN 2 Keyakinan Berlebihan pada Pengalaman proyek sebelumnya**

  Solusi 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 sistem dan fokus pengujian pada area tersebut.

  • Automatisasi Pengujian: 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 ke perusahaan 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.

- **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 antara persyaratan dan implementasi persyaratan, serta penyimpangan dari persyaratan

  • Cakupan diukur berdasarkan item yang diuji dalam dasar pengujian dan teknik yang diterapkan pada dasar pengujian

- Alasan 3 Keterbatasan sumber daya yang dimiliki

  Jadi mengapa testing diperlukan?

  - Karena perangkat lunak cenderung memiliki kesalahan
  - Untuk mempelajari tentang keandalan perangkat lunak
  - Untuk mengisi waktu antara pengiriman perangkat lunak dan tanggal rilis
  - Untuk membuktikan bahwa perangkat lunak tidak memiliki kesalahan
  - Karena pengujian termasuk dalam rencana proyek
  - Karena kegagalan bisa sangat mahal
  - Untuk menghindari dituntut oleh pelanggan
  - Untuk bertahan dalam bisnis

- **Intisari dalam pelatihan**

  Walaupun dengan tema yang menggambarkan jika software testing tidak perlukan merupakan pendapat yang salah karena software testing merupakan bagian penting dari

pengembangan perangkat lunak. Testing memastikan bahwa perangkat lunak yang dibangun berfungsi dengan benar, aman, dan dapat diandalkan. Tanpa testing, perangkat lunak mungkin mengalami bug dan kecacatan yang dapat merusak kinerja dan keandalannya. Testing juga membantu mengurangi risiko dan biaya kerusakan dan perbaikan yang mungkin terjadi di kemudian hari. Oleh karena itu, software testing adalah bagian penting dari pengembangan perangkat lunak dan harus dilakukan secara konsisten untuk memastikan kualitas produk yang dihasilkan.