PERTEMUAN KE-12
VALIDASI

12.1 TUJUAN PEMBELAJARAN

Adapun tujuan pembelajaran yang akan dicapai sebagai berikut:

12.1. Perencanaan validasi

12.2. Dasar-dasar uji coba

12.2 URAIAN MATERI

Tujuan Pembelajaran 12.1:
Perencanaan validasi

Verifikasi adalah suatu usaha untuk memastikan bahwa produk dibangun dengan tepat, dalam pengertian bahwa produk keluaran dari suatu aktivitas sesuai dengan spesifikasi yang dibebankan pada mereka di dalam aktivitas sebelumnya. Sedangkan validasi adalah suatu usaha untuk memastikan bahwa produk yang dibangun sudah benar, yaitu produk yang memenuhi tujuan spesifik yang diharapkan.

Kualitas produk perangkat lunak menunjuk V&V (Verifikasi dan Validasi) secara langsung dan penggunaan teknik pengujian dapat menempatkan kerusakan sedemikian rupa sehingga mereka dapat ditujukan. Proses V&V menentukan izin bagi produk dari aktivitas pemeliharaan atau pengembangan yang ditentukan untuk kebutuhan dari aktivitas tersebut, dan izin produk perangkat lunak akhir memenuhi tujuan yang diharapkannya dan sesuai kebutuhan pengguna.

Proses verifikasi dan proses validasi memulai awal tahap pengembangan atau pemeliharaan. Tujuan dari perencanaan V&V adalah untuk memastikan bahwa masing-masing sumber daya, peran, dan tanggung jawab dengan jelas ditugaskan. Proses tersebut akan menghasilkan dokumen rencana V&V, menguraikan berbagai sumber daya, aktivitas, dan peran mereka, seperti halnya teknik dan tool yang dapat digunakan. Pemahaman dari tujuan yang berbeda dari

tiap aktivitas V&V akan membantu dalam perencanaan yang hati-hati dari teknik dan sumber daya yang diperlukan untuk memenuhi tujuan mereka.

Rencana juga menunjuk manajemen, komunikasi, kebijakan, prosedur aktivitas V&V, dan interaksinya, seperti kebutuhan dokumentasi dan pelaporan yang rusak atau cacat.

Tujuan Pembelajaran 12.2:

Dasar-dasar uji coba

Pengujian merupakan rangkaian aktivitas yang dapat direncanakan sebelumnya dan dilakukan secara sistematis. Strategi uji coba perangkat lunak memudahkan para perancang untuk menentukan keberhasilan sistem yang telah dikerjakan. Hal yg harus diperhatikan adalah langkah-langkah perencanaan dan pelaksanaan harus direncanakan dengan baik dan berapa lama waktu, upaya dan sumber daya yang diperlukan. Strategi uji coba mempunyai karakteristik sebagai berikut:

a. Pengujian mulai pada tingkat modul yg paling bawah, dilanjutkan dengan modul di atasnya kemudian hasilnya dipadukan
b. Teknik pengujian yang berbeda mungkin menghasilkan sedikit perbedaan (dalam hal waktu)
c. Pengujian dilakukan oleh pengembang perangkat lunak dan (untuk proyek yang besar) suatu kelompok pengujian yang independen.
d. Pengujian dan debugging merupakan aktivitas yang berbeda, tetapi debugging termasuk dalam strategi pengujian.

Meningkatnya visibilitas (kemampuan) perangkat lunak sebagai suatu elemen sistem dan “biaya” yang muncul akibat kegagalan perangkat lunak, memotivasi dilakukannya perencanaan yang baik melalui pengujian yang teliti. Pada dasarnya, pengujian merupakan satu langkah dalam proses rekayasa perangkat lunak yang dapat dianggap sebagai hal yang merusak daripada membangun.

Dalam melakukan uji coba ada 2 masalah penting yang akan dibahas, yaitu:

A. Teknik uji coba perangkat lunak
B. Strategi uji coba perangkat lunak

A. Teknik uji coba perangkat lunak

Pada dasarnya, pengujian merupakan suatu proses rekayasa perangkat lunak yg dapat dianggap (secara psikologis) sebagai hal yg destruktif daripada konstruktif.

Sasaran pengujian:

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2. Test case yg baik adalah test case yg memiliki probabilitas tinggi untuk menemukan kesalahan yg belum pernah ditemukan sebalumnya.
3. Pengujian yg sukses adalah pengujian yg mengungkap semua kesalahan yg belum pernah ditemukan sebelumnya.

Prinsip pengujian:

1. Semua pengujian harus dapat ditelusuri sampai ke persyaratan pelanggan.
2. Pengujian harus direncanakan lama sebelum pengujian itu dimulai.
3. Prinsip Pareto berlaku untuk pengujian perangkat lunak. Prinsip Pareto mengimplikasikan 80% dari semua kesalahan yg ditemukan selama pengujian sepertinya akan dapat ditelusuri sampai 20% dari semua modul program.
4. Pengujian harus mulai "dari yg kecil" dan berkembang ke pengujian "yang besar".
5. Pengujian yg mendalam tidak mungkin.
6. Paling efektif, pengujian dilakukan oleh pihak ketiga yg independen.

Testabilitas perangkat lunak adalah seberapa mudah sebuah program komputer dapat diuji. Karena pengujian sangat sulit, perlu diketahui apa yg dapat dilakukan untuk membuatnya menjadi mudah.

Karakteristik perangkat lunak yg diuji :

1. Operabilitas, semakin baik dia bekerja semakin efisien dia dapat diuji.
2. Observabilitas, apa yg anda lihat adalah apa yg anda uji.
3. Kontrolabilitas, semakin baik kita dapat mengontrol perangkat lunak semakin banyak pengujian yg adapat diotomatisasi dan dioptimalkan.
4. Dekomposabilitas, dengan mengontrol ruang lingkup pengujian kita dapat lebih cepat mengisolasi masalah dan melakukan pengujian kembali.
5. Kesederhanaan, semakin sedikit yg diuji semakin cepat pengujian.
6. Stabilitas, semakin sedikit perubahan semakin sedikit gangguan pengujian.
7. Kemampuan dipahami, semakin banyak informasi yg dimiliki semakin detail pengujiannya.

Atribut pengujian yg baik :

- Memiliki probabilitas yg tinggi menemukan kesalahan.
- Tidak redundan.
- Harusnya ‘jenis terbaik’.
- Tidak boleh terlalu sederhana atau terlalu kompleks.

Terdapat bermacam-macam rancangan metode test case yg dapat digunakan, semua menyediakan pendekatan sistematis untuk uji coba, yg terpenting metode menyediakan kemungkinan yg cukup tinggi menemukan kesalahan.

Terdapat 2 macam test case:

1. Pengetahuan fungsi yg spesifik dari produk yg telah dirancang untuk diperlihatkan, test dapat dilakukan untuk menilai masing-masing fungsi apakah telah berjalan sebagaimana yg diharapkan.
2. Pengetahuan tentang cara kerja dari produk, test dapat dilakukan untuk memperlihatkan cara kerja dari produk secara rinci sesuai dengan spesifikasinya.

B. Strategi uji coba perangkat lunak

Strategi uji coba perangkat lunak memudahkan para perancang untuk menentukan keberhasilan system yg telah dikerjakan. Hal yang harus diperhatikan adalah langkah-langkah perencanaan dan pelaksanaan harus direncanakan dengan baik dan berapa lama waktu, upaya dan sumber daya yg diperlukan.

Strategi uji coba mempunyai karakteristik sebagai berikut :

- Pengujian mulai pada tingkat modul yg paling bawah, dilanjutkan dgn modul di atasnya kemudian hasilnya dipadukan.
- Teknik pengujian yang berbeda mungkin menghasilakn sedikit perbedaan (dalam hal waktu)
- Pengujian dilakukan oleh pengembang perangkat lunak dan (untuk proyek yang besar) suatu kelompok pengujian yang independen.
- Pengujian dan debugging merupakan aktivitas yang berbeda, tetapi debugging termasuk dalam strategi pengujian.

Pengujian yang dapat dilakukan adalah:

1. Pengujian unit

   Unit testing (uji coba unit) fokusnya pada usaha verifikasi pada unit terkecil dari desain perangkat lunak, yakni modul. Uji coba unit selalu berorientasi pada white box testing dan dapat dikerjakan paralel atau beruntun dengan modul lainnya.

2. Pengujian integrasi

   Pengujian terintegrasi adl teknik yg sistematis untuk penyusunan struktur program, pada saat bersamaan dikerjakan uji coba untuk memeriksa kesalahan yg nantinya digabungkan dengan interface.

   Metode pengujian integrasi:

   a) Top- down integration

      Top down integration adalah pendekatan incremental dengan menggerakkan ke bawah melalui hirarki control, dimulai dengan control utama. Strategi intergrasi top-down memeriksa control mayor atau keputusan pada saat awal di dalam proses pengujian. Pada struktur program yang difaktorkan dengan baik, penarikan keputusan

terjadi pada tingkat hirarki yang lebih tinggi sehingga terjadi lebih dulu.

Strategi top-down kelihatannya tidak sangat rumit, tetapi di dalam praktenya banyak menimbulkan masalah logistic. Biasanya masalah ini terjadi jika dibutuhkan pemrosesan di dalam hirarki pada tingkat rendah untuk menguji secara memadai tingkat yang lebih tinggi.

b) Pengujian Integrasi Bottom-up

Bottom up integration memulai konstruksi dan pengujian dengan modul atomic (modul pada tingkat paling rendah pada struktur program). Karena modul diintegrasikan dari bawah ke atas, maka pemrosesan yang diperlukan untuk modul subordinate ke suatu tuingkat yang diberikan akan selalu tersedia dan kebutuhan akan stub dapat dieliminasi. Strategi integrasi bottom-up dapat diimplementasi dengan langkah-langkah:

1. modul tingkat rendah digabung ke dalam cluster (build) yang melakukan subfungsi perangkat lunak spesifik.
2. Driver (program control untuk pengujian) ditulis untuk mengkoordinasi input dan output test case
3. cluster diuji
4. driver diganti dan cluster digabungkan dengan menggerakkannya ke atas di dalam struktur program.

12.3 LATIHAN SOAL/TUGAS

1. Kenapa perlu dilakukan pengujian terhadap perangkat lunak yang dikembangkan?

12.4 DAFTAR PUSTAKA

1. Pressman, R. S. (2015). Software Engineering. A Practitioner's Approach (8th ed.). New York: McGraw-Hill Education.
2. Sommerville, I. (2011). Software Engineering (9th ed.). Boston: Addison-Wesley.