4 Validasi dan Verifikasi Software

1. Validasi dan Verifikasi Software

1.1. Verifikasi vs. Validasi

Verifikasi: “Are we building the product right “

Software seharusnya sesuai dengan spesifikasinya

Validasion:"Are we building the right product”.

Software seharusnya melakukan apa yang benar-benar disyaratkan oleh user.

1.2. Proses Verifikasi & Validasi

Proses Verifikasi & Validasi adalah keseluruhan proses daur hidup V & V harus diterapkan pada setiap tahapan dalam proses software yang mempunyai dua obyektif prinsipal yaitu:

- Menemukan kekurangan dalam sebuah sistem;
- Memperkirakan apakah sistem berguna dan dapat digunakan atau tidak dalam situasi operasional

1.3. Tujuan Verifikasi & Validasi

Verifikasi dan validasi harus memberikan kepastian bahwa software sesuai dengan tujuannya. Hal ini bukan berarti benar-benar bebas dari kekurangan. Harus cukup baik untuk tujuan penggunaannya dan tipe dari penggunaan akan menentukan derajat kepastian yang dibutuhkan

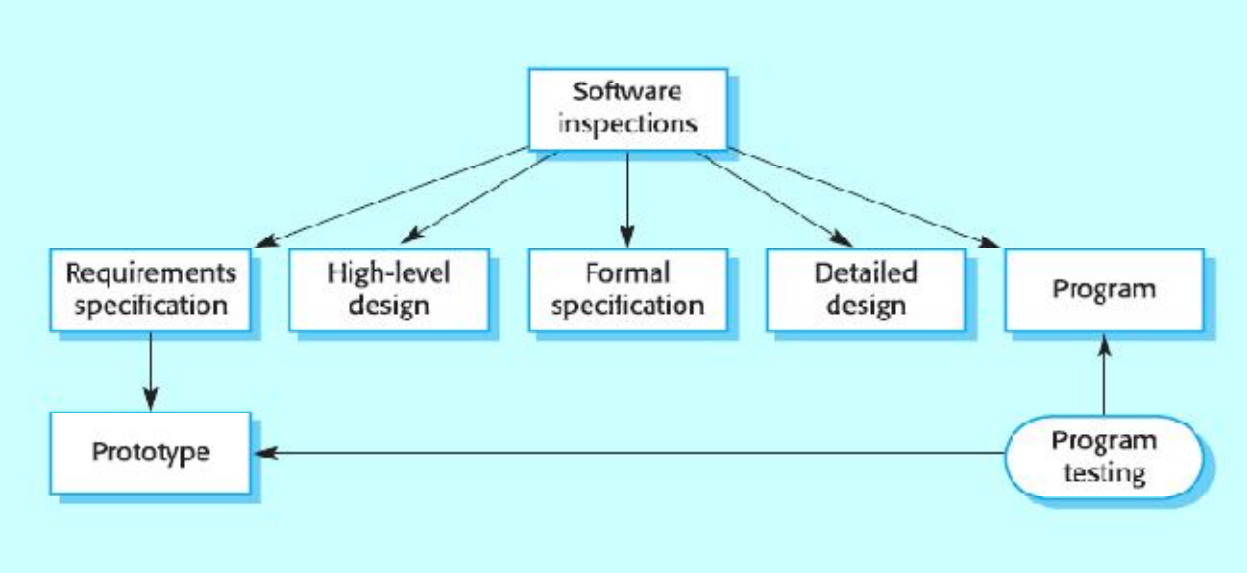
1.4. Kepastian Verifikasi & Validasi

- Tergantung pada tujuan sistem atau fungsi software yaitu Tingkat kepastian tergantung pada bagaimana kritikal software terhadapa sebuah organisasi.
- Tergantung pada Harapan User, user mungkin mempunyai harapan yang rendah terhadap software yang ada
- Tergantung pada lingkungan pemasaran, lebih awal melempar sebuah produk ke pasar lebih penting daripada menemukan kekurangan dalam program

1.5. Verifikasi Statik & Dinamik

Software inspection. Berhubungan dengan analisis representasi sistemstatik untuk menemukan masalah (verifikasi statik). Dapat menjadi tambahan dari tool-based document dan code analysis

Software testing. Berhubungan dengan pelaksanaan dan memperhatikan perilaku produk (dinamik verifikasi). Sistem dijalankan dengan data tes dan perilaku operasionalnya diperhatikan



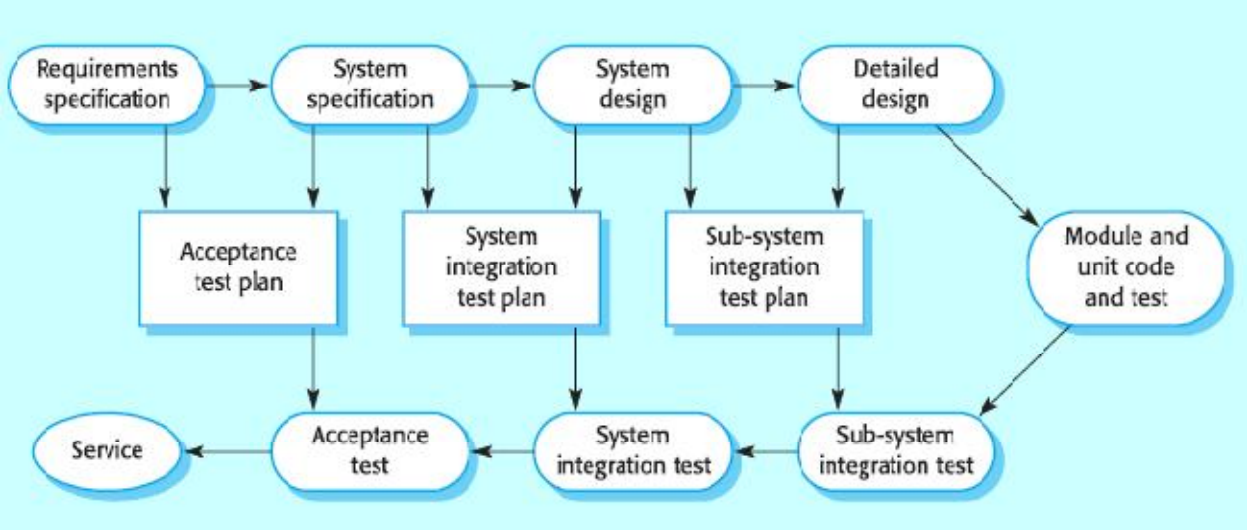
2. Pengujian Program

Pengujian Program dapat mengungkapkan keberadaan kesalahan bukan ketidak beradaannya. Hanya teknik validasi untuk persyaratan non-functional sebagai sebuah software dapat dijalankan untuk melihat bagaimana perilakunya. Harusnya digunakan dalam hubungannya dengan verifikasi statik untuk menyediakan penanganan Verifikasi & Validasi yang menyeluruh

2.1. Tipe Pengujian

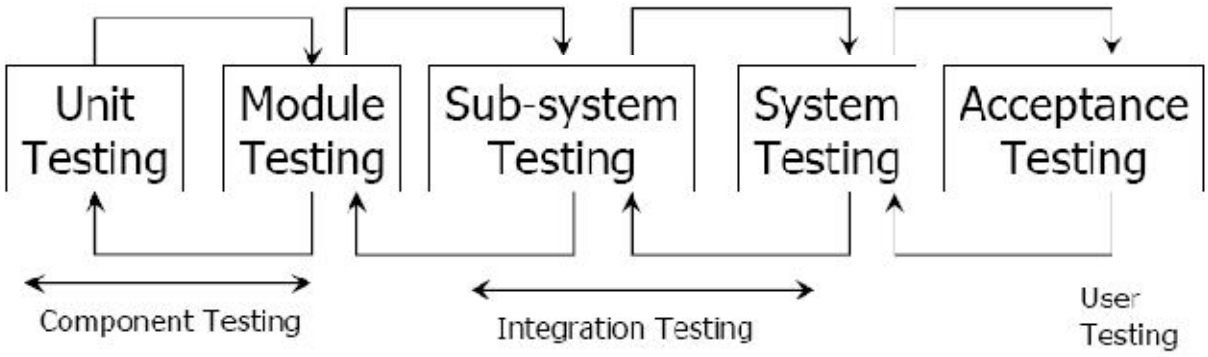
- Pengujian Kekurangan, dalam pengujian kekurangan test dirancang untuk menemukan kekurangan sistem. Uji kekurangan yang berhasil salah satunya adalah menunjukkan keberadaan kekurangan dalam sebuah sistem
- Pengujian Validasi ditujukan untuk memperlihatkan bahwa software sesuai dengan persyaratannya. Tes yang berhasil adalah salahsatu yang menunjukkan bahwa persyaratan telah diterapkan secara tepat

Pengembangan Model Verifikasi



2.2. Proses Testing

- System Testing, pengujian terhadap integrasi sub-system, yaitu keterhubungan antar sub-system
- Acceptance Testing adalah Pengujian terakhir sebelum sistem dipakai oleh user. Melibatkan pengujian dengan data dari pengguna sistem. Biasa dikenal sebagai "alpha test" ("beta test" untuk software komersial, dimana pengujian dilakukan oleh potensial customer)



1. Component testing adalah pengujian komponen- komponen program dan biasanya dilakukan oleh component developer (kecuali untuk system kritis)
2. Integration testing merupakan pengujian kelompok komponen-komponen yang terintegrasi untuk membentuk sub-system ataupun system. dilakukan oleh tim penguji yang independent. Pengujian berdasarkan spesifikasi sistem

2.3. Rencana Pengujian

- Proses testing

  Deskripsi fase-fase utama dalam pengujian

- Pelacakan Kebutuhan

  Semua kebutuhan user diuji secara individu

- Item yg diuji

  Menspesifikasi komponen sistem yang diuji

- Jadual testing

  Prosedur Pencatatan Hasil dan Prosedur kebutuhan akan Hardware dan Software

- Kendala-kendala

  Mis: kekurangan staff, alat, waktu dll.

2.4. Failures & Faults

- Failure : output yang tidak benar/tidak sesuai ketika sistem dijalankan
- Fault : kesalahan dalam source code yang mungkin menimbulkan failure ketika code yang fault tsb dijalankan

Failure Class	Deskripsi
Transient	Muncul untuk input tertentu
Permanent	Muncul untuk semua input
Recoverable	Sistem dapat memperbaiki secara otomatis
Unrecoverable	Sistem tidak dapat memperbaiki secara otomatis
Non-corrupting	Failure tidak merusak data
Corrupting	Failure yang merusak sistem data

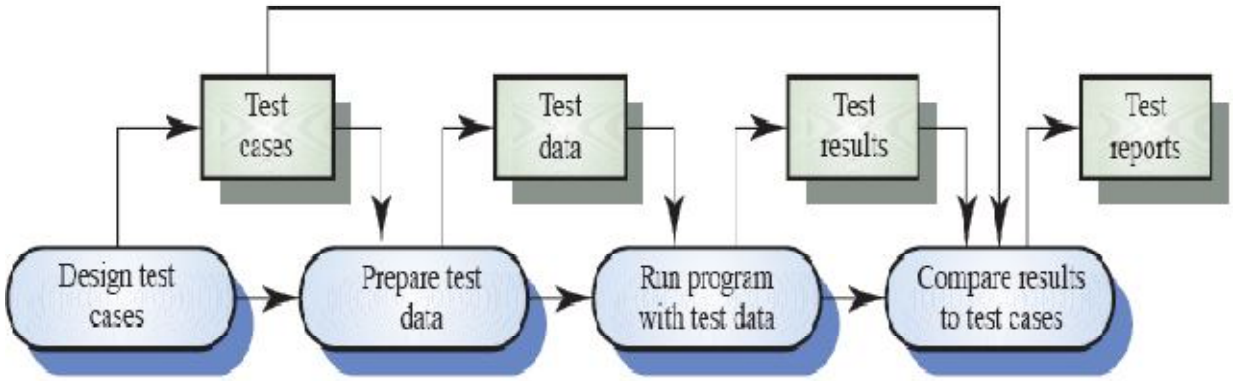
2.5. Prioritas Testing

Hanya test yang lengkap yg dapat meyakinkan sistem terbebas dari kesalahan, tetap hal ini sangat sulitd ilakukan. Prioritas dilakukan terhadap pengujian kemampuans istem, bukan masing-masing komponennya. Pengujian untuk situasi yg tipikal lebih penting dibandingkan pengujian terhadap nilai batas.

2.6. Test Data & Test Kasus

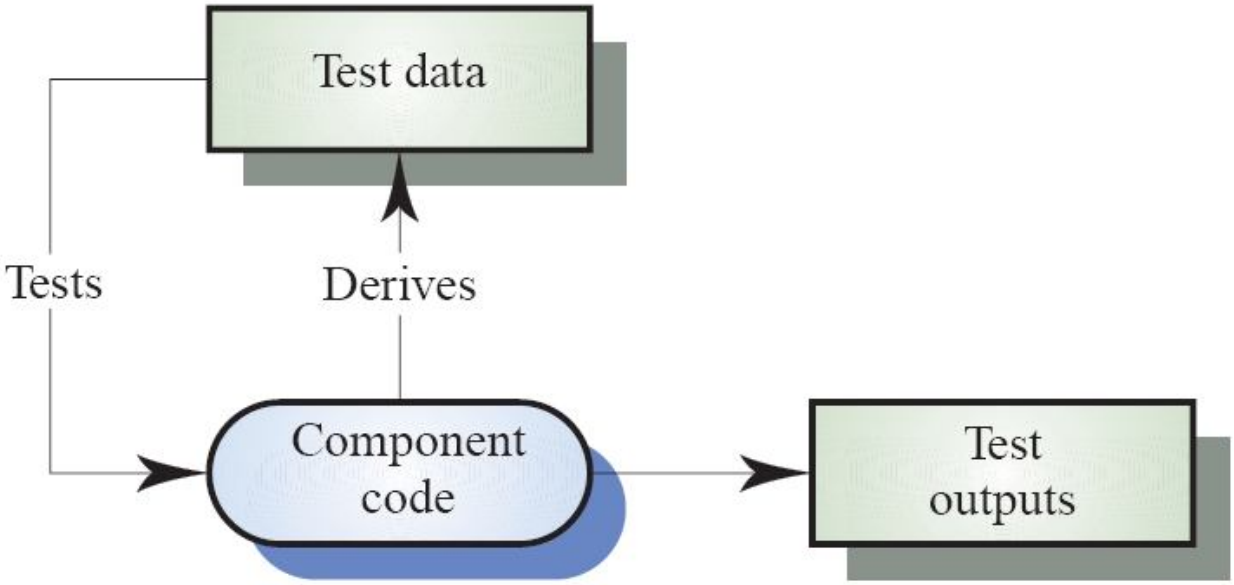
- Test data : Input yang yang direncankan digunakan oleh sistem.
- Test cases : Input yang digunakan untuk menguji sistem dan memprediksi output dari input jika sistem beroperasi sesuai dengan spesifikasi.

2.7. Proses Defect Testing



3. Struktural Testing

Struktural testing disebut juga white-box testing. Penentuan test case disesuaikan dengan struktur sistem. Knowledge program digunakan untuk mengidentifikasi test case tambahan. Tujuannya untuk menguji semua statement program (debug).



3.1. Path Testing

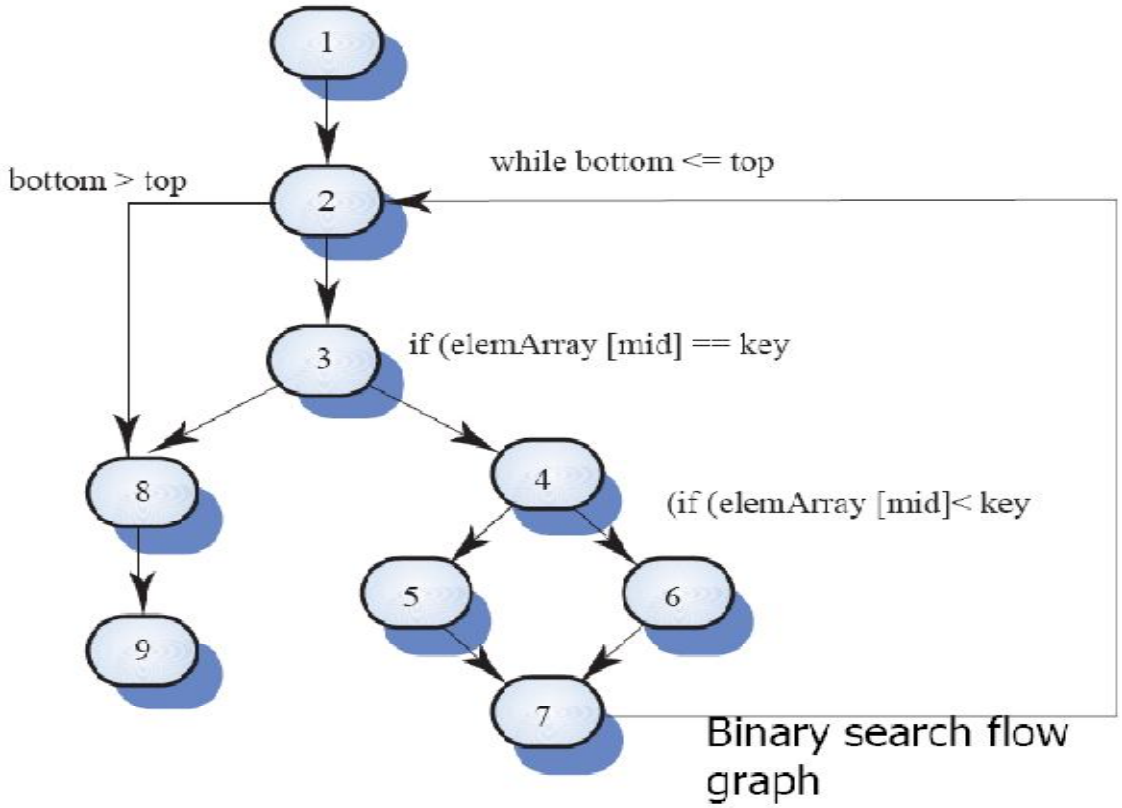
Tujuannya meyakinkan bahwa himpunan test case akan menguji setiap path pada suatu program paling sedikit satu kali. Titik awal untuk path testing adalah suatu program flow graph yang menunjukkan node-node yang menyatakan program decisions (mis.: if-then-else condition) dan busur menyatakan alur control. Statements dengan conditions adalah node-node dalam flow graf.

3.2. Program Flow Graph

Menggambarkan alur kontrol.Setiap cabang ditunjukkan oleh path yg terpisah dan loop ditunjukkan oleh arrows looping kembali ke loop kondisi node. Digunakan sebagai basis untuk menghitung cyclomatic complexity.

Cyclomatic complexity = (Jumlah edges –Jumlah Node) + 2.

Cyclomatic complexity menyatakan jumlah test untuk menguji control statements



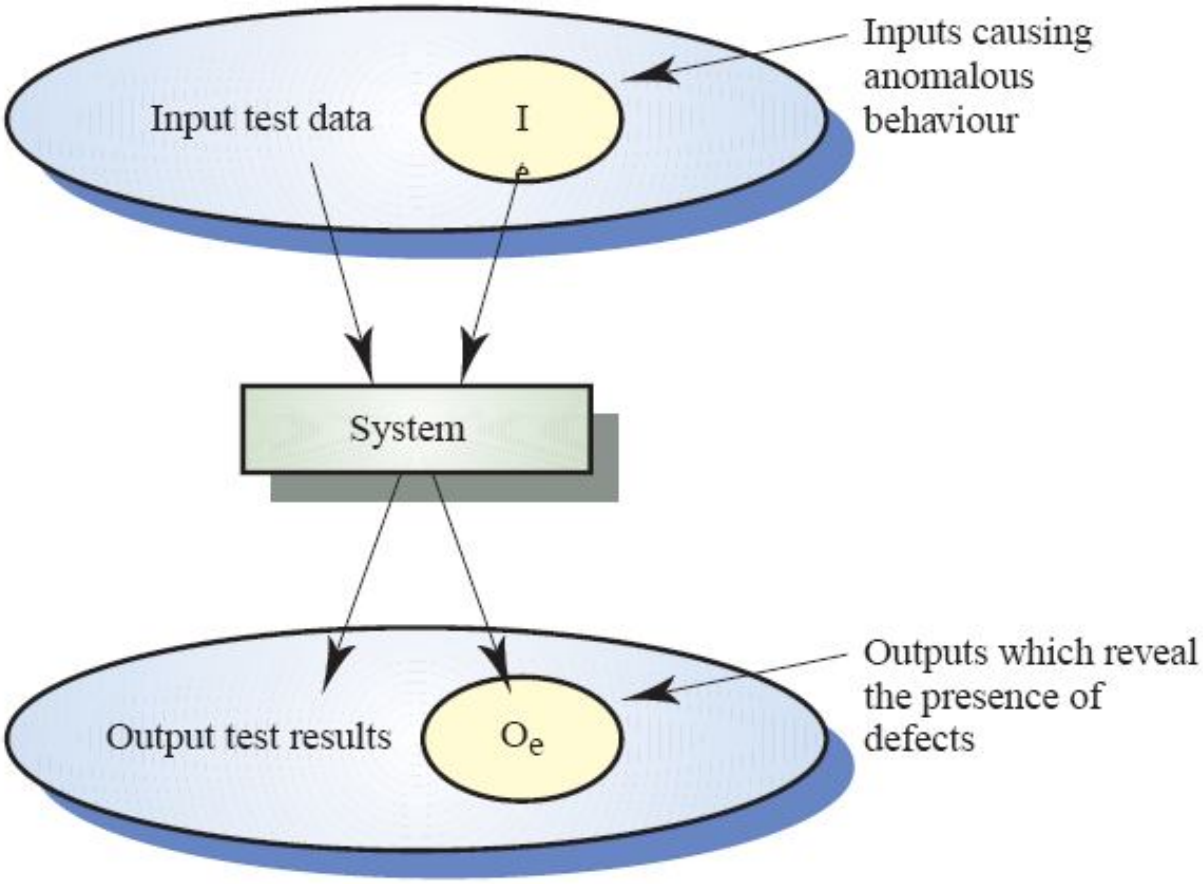
Binary search flow graph

3.3. Independent Path

- 1, 2, 3, 8, 9
- 1, 2, 3, 4, 6, 7, 2
- 1, 2, 3, 4, 5, 7, 2
- 1, 2, 3, 4, 6, 7, 2, 8, 9
- Test cases harus ditentukan sehingga semua path tsb tereksekusi.

4. Black Box Testing

Pendekatan pengujian dimana program dianggap sebagai suatu'black-box' ('kotak hitam'). Program test case berbasiskan spesifikasi. Test planning dapat dimulai sejak awal proses pengembangan sistem



Pengujian black box berusaha menemukan kesalahan dalam kategori:

- Fungsi-fungsi yang tidak benar atau hilang
- Kesalahan interface
- Kesalahan dalam struktur data atau akses database eksternal
- Kesalahan kinerja
- Inisialisasi dan kesalahan terminasi

Referensi

1. Roger S Pressman, "Software Enginering: A Practitioners Approach"

2. Bob Hughes, "Software Project Management"