ANALISIS dan PERANCANGAN SISTEM INFORMASI

RANGKUMAN ANALISIS DAN PERANCANGAN SISTEM

MANAJEM EN PROYEK & PROPOSAL SISTEM

A. ANALISIS KELAYAKAN

Kelayakan merupakan alat ukur seberapa besar manfaat atau praktikal suatu sistem informasi terhadap suatu organisasi. Analisis kelayakan menggunakan pendekatan *creeping commitment* (komitmen yang luwes), dimana pendekatan ini meninjau ulang kelayakan pada titik-titik tertentu selama daur hidup pengembangan berjalan.

- **Titik-titik pemeriksaan** (*checkpoints*):
  - Analisis sistem—Studi awal
  - Analisis sistem—Analisis masalah
  - Perancangan sistem—Analisis keputusan
- Cakupan analisis kelayakan :
  - **Kelayakan teknikal**: ukuran seberapa praktikal suatu solusi secara teknis serta seberapa siap dari sisi sumber daya maupun keahlian.
  - **Kelayakan operasional**: ukuran seberapa baik solusi itu bekerja dan bagaimana tanggapan orang-orang dalam organisasi terhadap sistem/proyek tersebut.
  - **Kelayakan dari sisi ekonomi** : ukuran biaya-efektifitas (*cost-effectiveness)* dari proyek atau solusi.
  - **Kelayakan dari sisi jadwal** : ukuran seberapa rasional jadwal proyek yang ditetapkan.

B. ANALISIS BIAYA - KEUNTUNGAN (*COST-BENEFIT ANALYSIS*)

- **Cost**
  - Biaya pengembangan : biaya yang diperlukan/dikeluarkan satu kali (*one time cost*) yang tidak akan muncul lagi setelah proyek selesai
  - Biaya operasional : biaya yang akan muncul selama sistem digunakan:
    - o Biaya tetap: Muncul dalam kurun waktu yang teratur dan relatif sama besarannya
    - o Biaya berubah: Muncul secara proporsional sesuai penggunaan
- **Benefits**
  - Keuntungan yang terlihat (*tangible*) : yang mudah untuk dikuantitatifkan
  - Keuntungan yang tidak terlihat (*intangible*) : yang sulit atau dianggap sulit untuk diukur

C. ANALISIS DARI SISI EKONOMI

- **Analisis "balik-modal" (*payback*)** untuk menetapkan apakah dan kapan investasi yang dikeluarkan akan kembali. Periode balik-modal merupakan periode waktu yang berjalan sebelum nilai kumulatif keuntungan akan melampaui nilai kumulatif biaya

- **Nilai bersih saat ini** (*net present value*). Contohnya: Satu juta Rupiah saat ini lebih bernilai dibandingkan satu juta rupiah satu tahun yang akan datang.
- **Return-on-investment (ROI)** : Teknik membandingkan keuntungan kumulatif dari beberapa alternatif solusi. ROI dari suatu solusi atau proyek merupakan rasio persentase untuk mengukur hubungan antara jumlah yang didapat dari suatu investasi dengan investasi
  - **Lifetime ROI** = (estimasi lifetime benefits - estimated lifetime costs) / estimated lifetime costs
  - **Annual ROI** = lifetime ROI / lifetime sistem

D. MANAJEMEN PROYEK VS. MANAJEMEN PROSES

Manajemen proyek adalah proses penentuan cakupan, perencanaan, penentuan tenaga kerja, pengorganisasian, pengarahan, dan pengendalian pengembangan dari suatu sistem menggunakan biaya yang minimal dalam waktu yang telah ditentukan

Manajemen proses adalah aktivitas pendokumentasian, pengelolaan, dan perbaikan secara terus menerus proses pengembangan sistem. Manajemen proses peduli dengan bagaimana aktivitas, hasil kerja, dan standar mutu diterapkan pada seluruh proyek.

E. DEFINISI PROYEK BERHASIL

- Hasil dari sistem informasi bisa diterima oleh konsumen
- Sistem diserahkan “tepat waktu”
- Sistem diserahkan “sesuai anggaran”
- Proses pengembangan sistem memberikan impak minimal terhadap operasi bisnis yang sedang berjalan

F. ALAT DAN TEKNIK MANAJEMEN PROYEK

- **PERT Chart** (Project Evaluation and Review Technique) adalah model jaringan grafis yang digunakan untuk menggambarkan interdependensi dan hubungan antar pekerjaan proyek. PERT chart lebih efektif digunakan untuk mempelajari hubungan antar pekerjaan proyek.
- **Gantt Chart** adalah diagram batang horisontal sederhana yang menggambarkan pekerjaan proyek terhadap kalender. Kelebihan dari Gantt chart adalah menunjukkan secara jelas pekerjaan yang overlapping, yaitu pekerjaan yang dilakukan pada saat bersamaan. Batang yang diarsir menunjukkan persentase penyelesaian dan kemajuan proyek. Gantt chart lebih efektif digunakan untuk mengkomunikasikan jadwal.

G. AKTIVITAS MANAJEMEN PROYEK

1. Negosiasi Cakupan : produk, kualitas, waktu, biaya, sumber daya

2. Mengidentifikasi pekerjaan/pekerjaan : WBS (Work Breakdown Structure) yaitu alat grafis yang digunakan untuk menggambarkan dekomposisi hirarkis suatu proyek menjadi fase, aktivitas dan pekerjaan.
3. Mengestimasi durasi pekerjaan
4. Menspesifikasikan ketergantungan antar pekerjaan

   Ketergantungan antar pekerjaan :
   - Finish-to-Start (FS) : Pekerjaan selesai memicu pekerjaan yang lain untuk mulai.
   - Start-to-Start (SS) : Satu pekerjaan dimulai memicu pekerjaan yang lain untuk mulai
   - Finish-to-Finish (FF) : Dua pekerjaan harus selesai di waktu yang sama
   - Start-to-Finish (SF) : Satu pekerjaan dimulai menandakan pekerjaan lain selesai.
5. Menetapkan sumber daya
6. Mengarahkan usaha tim
7. Mengawasi dan mengontrol progres
8. Menilai hasil dan pengalaman proyek

H. LAPORAN PROPOSAL SISTEM

Format Faktual	Format Administratif
Pendahuluan	Pendahuluan
Metode dan prosedur	Kesimpulan dan saran
Fakta dan detail	Ringkasan dan diskusi fakta dan detail
Diskusi dan analisis fakta & detail	Metode dan prosedur
Rekomendasi	Kesimpulan akhir
Kesimpulan	

SYSTEM DESIGN & ARCHITECTURE

A. PERANCANGAN SISTEM

Perancangan sistem informasi: Pekerjaan yang fokus pada spesifikasi detil dari solusi berbasis komputer atau disebut juga perancangan physical.

Perbedaan dengan **analisis sistem** : Analisis sistem menekankan pada permasalahan bisnis (WHAT), maka perancangan sistem menekankan pada pertimbangan teknis dan implementasi sistem (HOW)

Metode:

- Membangun sendiri (in-house), atau
- Membeli (COTS=commercial-off-the-shelf)

B. ARSITEKTUR APLIKASI

Menunjukkan teknologi yang akan digunakan untuk mengimplementasi satu atau lebih (atau mungkin seluruh) sistem informasi dalam hal data, proses, dan antarmuka, dan bagaimana komponen-komponen ini saling berinteraksi dalam jaringan. Berperan sebagai bagan/garis besar atau cetak biru (blueprint) dari perancangan detil dan implementasi.

C. PHYSICAL DATA FLOW DIAGRAM (DFD)

Memodelkan hal-hal teknis dan keputusan manusia yang akan diimplementasikan sebagai bagian dari sistem informasi (HOW, by WHOM). Sebagai alat untuk mengkomunikasikan pilihan hal-hal teknis dan keputusan perancangan lainnya kepada mereka (tim/orang) yang akan membangun atau mengimplementasikan sistem

- **Proses physical**

  Merupakan suatu "prosesor" seperti komputer atau seseorang, atau "implementasi teknis dari suatu pekerjaan" yang akan dikerjakan seperti program komputer atau proses manual

- **Aliran data physical**

  Dapat merepresentasikan:

  - Rencana implementasi input dan output yang akan masuk ke dalam dan keluar dari proses physical
  - Perintah kueri basis data seperti create, read, update, dan delete (CRUD)
  - Data yang diimport dari, atau dieksport ke sistem informasi dalam jaringan
  - Aliran data antar modul atau subroutine (sebagai suatu proses physical) dalam suatu program

D. SISTEM TERSENTRALISASI VS. TERDISTRIBUSI

- **Sistem tersentralisasi**

  Komputer multi-user terpusat menjadi host dari seluruh komponen data, proses, dan antarmuka dari suatu sistem informasi. Pengguna berinteraksi dengan sistem melalui berbagai terminal (atau emulator).

- **Sistem terdistribusi**

  Komponen data, proses, dan antarmuka dari suatu sistem informasi terdistribusi di beberapa lokasi dalam suatu jaringan komputer. Karena itu, beban kerja pengolahan juga terdistribusi dalam jaringan.

E. UNIT PERANCANGAN

Merupakan koleksi yang mencakup proses, penyimpanan data, dan aliran data yang menggunakan karakteristik desain yang mirip. Unit Perancangan merupakan subset dari sistem keseluruhan di mana input, output, file dan basis data, serta programnya dapat dirancang, dibangun, dan diuji coba sebagai satu kesatuan unit. Pada akhirnya, harus digabungkan dengan sistem keseluruhan

USER INTERFACE DESAIN

A. JENIS OUTPUT

- **Internal** output: suatu output yang ditujukan kepada pemilik atau pengguna dalam organisasi. Laporannya dapat detil tanpa atau dengan sedikit saringan, atau dapat juga dapat berupa ringkasan. Misalnya SIPADU memberikan informasi kepada dosen PA tentang mahasiswa bimbingannya saja, dan dosen lain tidak dapat melihatnya.
- **External** output: Ditujukan kepada konsumen, pemasok, rekanan, atau institusi lain di luar organisasi. Dan laporannya telah disaring sesuai peruntukannya. Misalkan pada website BPS yang menapilkan alumni STIS, disini informasi disaring sehingga yang dapat dilihat oleh orang luar hanyalah informasi umum saja.
- **Turnaround** output: Keluaran dari sistem yang membutuhkan umpan balik dari pengguna ke dalam sistem, contoh : tagihan, disini pengguna harus melakukan

➢ Prinsip Desain Output

- **Pengguna** yang tepat: Pendistribusian (atau akses ke) output kepada pengguna yang relevan
- **Format** yang tepat: Output harus mudah dibaca dan diterjemahkan (user friendly)
- **Informasi** yang tepat: Output harus dapat diterima (memenuhi kebutuhan bisnis) oleh pengguna sistem yang menerimanya
- **Waktu** yang tepat: Informasi bisa didapat pada saat dibutuhkan dan akurat (timely & currently

➢ Tahapan Perancangan Output

1. Identifikasi output sistem dan tinjau persyaratan logis.
2. Tentukan persyaratan output fisik.
3. Bila perlu, rancanglah formulir pracetak apapun.
4. Merancang, memvalidasi dan menguji output menggunakan beberapa kombinasi dari :
   - Layout tools (misalnya sketsa tangan, diagram jarak, atau alat CASE)
   - Prototyping tools (misalnya, spreadsheet, PC DBMS, 4GL)
   - Code generating tools (mis., Penulis laporan)

B. DESAIN INPUT

Desain input merupakan bagaimana cara mendapatkan data serta menjadikannya ke dalam suatu format yang dibutuhkan oleh komputer.

- **Data Capture** : Proses pengumpulan data dari sumbernya. Media : Kuesioner, lembar jawaban komputer.

- **Data Entry** : Proses menerjemahkan data (hasil dari Data Capture) menjadi format yang dapat dibaca oleh komputer. Data Entry akan menghasilkan suatu input data, dimana input data ini pun nantinya akan diproses.
- **Data processing**: Seluruh pengolahan terhadap data setelah menjadi bentuk yang bisa dibaca oleh komputer
  - Batch: Data yang telah di-entry, dikumpulkan dalam bentuk file yang disebut batch, lalu diolah sekaligus.
  - On-line: Data yang ditangkap langsung diolah begitu masuk ke dalam komputer.
  - Remote batch: Data di-entry dan di-edit secara on-line, namun dikumpulkan dulu dalam batch sebelum diolah sekaligus.

    Contoh : Presensi mahasiswa -> Absen fingerprint dikumpulkan di dalam satu komputer yang ada di ruang kelas, lalu data akan diproses (dikirim) ke server ketika sesi berakhir.

➢ **Pengendalian internal terhadap input**

Sangat penting dilakukan untuk memastikan data yang dimasukkan ke komputer akurat dan sistem terlindung dari *fraud*. Panduan untuk mengontrol input internal :

1. Monitoring input : Supaya tidak terjadi data yang salah letak, hilang, atau terlewat
2. Validasi semua data : Digunakan untuk mejaga konsistensi dan keamanan

➢ **Tahapan perancangan input**

1. Kenali input sistem dan tinjau kebutuhan logis
2. Pilih kontrol/komponen GUI yang cocok untuk input
3. Rancang dokumen sebagai sumber data (mis. formulir, kuesioner, ljk)
4. Rancang, validasi, dan uji input menggunakan teknik:
5. Layout tools (mis. sketsa, spasing charts, atau CASE tools)
6. Prototyping tools (mis. spreadsheets, IDE tools)

C. DESAIN USER INTERFACE

Desain User Interface merupakan integrasi dari desain input dan output.

Klasifikasi user :

- Expert User -> Paham dan terbiasa menggunakan komputer
- Novice User -> Pemahaman komputer masih rendah bahkan bisa tidak paham sama sekali

➢ **Masalah pada Interface**

- Penggunaan istilah-istilah dalam komputer dan akronim secara berlebihan
- Desain kurang intuitif
- Ketidakmampuan untuk membedakan antara tindakan alternatif ("apa yang harus saya lakukan selanjutnya?")
- Pendekatan pemecahan masalah yang tidak konsisten

- Desain yang tidak konsisten

➢ **Proses Mendesain User Interface**

1. Buat bagan dialog user-interface
2. Prototipe dialog dan user-interface
3. Dapatkan feedback (untuk menguji user-interface)
4. Jika perlu, ulangi kembali ke langkah 1 atau 2

SOFTWARE QUALITY

A. DEFINISI KUALITAS SOFTWARE

Software Quality adalah proses (pengembangan) software yang efektif yang diterapkan sehingga dapat menghasilkan produk yang berguna yang menyediakan nilai yang bisa diukur baik bagi pembuat maupun penggunanya

➢ **Rumus kualitas**

Kepuasan Pengguna = Produk sesuai spesifikasi/standar
+ Kualitas bagus
+ Serah terima sesuai waktu dan anggaran

—Robert Glass, 1998

B. FAKTOR KUALITAS—ISO 9126:2001

- **Fungsionalitas**: Sejauh mana perangkat lunak memenuhi kebutuhan yang dinyatakan dalam (kesesuaian, akurasi, interoperabilitas, kepatuhan, dan keamanan)
- **Reliability**: Jumlah waktu perangkat lunak tersedia untuk digunakan (jatuh tempo, toleransi kesalahan, dan pemulihan)
- **Usability**: Sejauh mana perangkat lunak mudah digunakan (dapat dimengerti, dipelajari, dan dioperasikan)
- **Efficiency**: Sejauh mana perangkat lunak mengoptimalkan penggunaan sumber daya sistem (perilaku waktu dan perilaku sumber daya)
- **Maintainability**: Kemudahan dalam melakukan perbaikan perangkat lunak (dapat dianalisis, dapat diubah, stabilitas, dan dapat diuji)
- **Portability**: Kemudahan perangkat lunak dapat dialihkan dari satu lingkungan ke lingkungan lain (kemampuan beradaptasi, dapat dipasang, kesesuaian, dan dapat diganti)

C. FAKTOR KUALITAS—ISO 25010:2011

Functional sutitability	Performance efficiency	Compatibility	Usability	Reliability	Security	Maintainability
Functional completeness	Time behavior	Co-existance	Appropriate-ness recognizability	Maturity	Confidentiality	Modularity
Functional correctness	Resource utilization	Interoperability	Learnability	Availability	Integrity	Reusability
Functional appropriate-ness	Capacity		Operability	Fault tolerance	Non-repudiation	Analyzability
			User error protection	Recoverability	Accountability	Modifiability
			User interface aesthetics		Authenticity	Testability
			Accessability			

Portability
Adaptability
Installability
Replaceability

D. DILEMA KUALITAS : "GOOD ENOUGH" Software

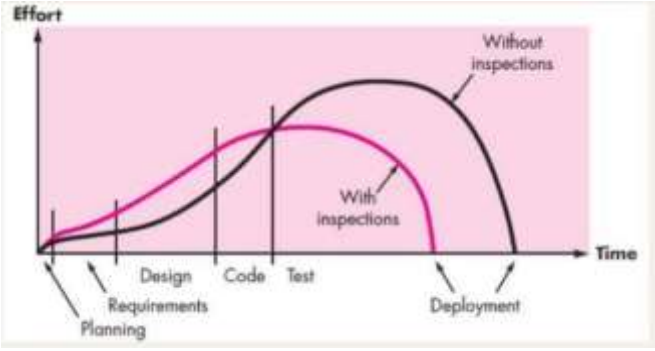
Jika kita memproduksi suatu software/sistem yang memiliki kualitas yang buruk, kita akan gagal karena tidak ada pengguna yang ingin membeli/menggunakan. Namun jika kita membuat suatu software/sistem yang sangat sempurna dengan meluangkan waktu pembuatan yang lama, modal uang yang banyak, dan usaha yang sangat keras maka akan menghasilkan suatu produk yang sangat mahal lalu kita akan gagal dalam persaingan pasar. Maka sebagai pembuat sistem diharapkan memiliki kemampuan untuk membuat sistem yang "GOOD ENOUGH". "GOOD ENOUGH" Software -> Software yang memberikan fungsi dan fitur dengan kualitas yang tinggi, namun dalam waktu yang sama juga memberikan fungsi dan fitur yang tidak mencolok serta tidak memiliki bugs

Perbaikan Error dan Defect

- **Error** (atau Bug) : Permasalahan kualitas yang ditemukan sebelum software dirilis
- **Defect** (atau fault) : permasalah kualitas yang ditemukan setelah software dirilis dan digunakan oleh end user.

E. TECHNICAL REVIEW

Tujuan utama review teknis (technical review) adalah untuk menemukan error selama proses (pengembangan) software sehingga tidak menjadi defect setelah rilis.



Grafik di atas menunjukan perbedaan dari perbaikan suatu sistem dengan teknik review dan perbaikan sistem tanpa teknik review. Dapat dilihat bahwa garis berwarna merah (menggunakan teknik review) menunjukan bahwa perbaikan suatu sistem memakan waktu (time) yang lebih singkat dan usaha (effort) yang lebih rendah dibanding garis hitam (tanpa menggunakan review). Maka dapat disimpulkan bahwa dengan teknik review dapat mempermudah dan mempersingkat kegiatan perbaikan suatu sistem.

F. CARA REVIEW TEKNIS

- **Informal** technical review : Dengan cara melakukan pemeriksaan pekerjaan secara sederhana sesama rekan kerja atau dengan melakukan pertemuan informal
- **Formal** technical review (FTR) : Pengaturan kualitas software yang dilakukan oleh software engineers (dan anggota tim lainnya) secara formal.

G. SOFTWARE QUALITY ASSURANCE (SQA)

Software Quality Assurance (Jaminan Kualitas Software) SQA merupakan aktivitas pelindung yang diaplikasikan pada seluruh proses perangkat lunak

- **Tujuan SQA** :
  1. Menjamin kualitas kebutuhan/permintaan pengguna software tsb
  2. Menjamin kualitas desain suatu software
  3. Menjamin kualitas kode program
  4. Menjamin efektifitas pengaturan kualitas software
- **Langkah-langkah SQA (1) :**
  1. Menyiapkan rencana untuk suatu proyek.
  2. Berpartisipasi dalam pengembangan proses proyek beserta deskripsinya.
  3. Melakukan review dan audit aktifitas pekerjaan software engineering beserta produk yang dihasilkan untuk memastikan kesesuaian (compliance) dengan definisi proses.
  4. Memastikan perbedaan baik dalam aktifitas pekerjaan maupun produk yang dihasilkannya terdokumentasi dan ditindaklanjuti.
  5. Merekam setiap ketidaksesuaian dan membuat laporan pada senior manajemen
- **Langkah-langkah statistical SQA -> lebih kuantitatif (2) :**
  1. Informasi tentang error dan defect dikumpulkan dan dikategorikan
  2. Upaya untuk melacak error dan defect ke akar permasalahannya
  3. Menggunakan prinsip Pareto: 80% dari permasalahan bisa dilacak dari 20% kemungkinan penyebabnya
  4. Setelah ditemukan penyebab utamanya, mulai dilakukan perbaikan

H. MENGUKUR KEANDALAN (RELIABILITY) DAN KEBERADAAN (AVAILABILITY)

- **Keandalan (reliability),** sederhananya adalah mean-time-between-failure (MTBF) dengan rumus: MTBF = MTTF + MTTR

  MTTF: mean-time-to-failure

  MTTR: mean-time-to-repair

- **Keberadaan (availability)** adalah probabilita suatu program beroperasi sesuai permintaannya dengan rumus:

$$Availability = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

SOFTWARE TESTING

A. PENDEKATAN STRATEGIS DALAM UJICOBA

- Untuk melakukan ujicoba secara efektif, perlu dilakukan review teknis yang efektif → banyak error bisa dihilangkan sebelum pelaksanaan ujicoba
- Ujicoba dimulai dari tingkat komponen menuju integrasi ke sistem secara keseluruhan
- Teknik ujicoba harus dibedakan sesuai pendekatan rekayasa perangkat lunak dan titik waktu pelaksanaannya
- Ujicoba dilakukan oleh pengembang software dan (untuk skala besar) dilakukan oleh kelompok independen (ITG)
- Ujicoba dan debugging adalah dua aktifitas yang berbeda, namun debugging merupakan rangkaian aktifitas dalam setiap strategi ujicoba

B. VERIFIKASI DAN VALIDASI

- **Verification** : "Are we building the product right?"

  Verification merupakan kumpulan kegiatan untuk memastikan software mengimplementasikan fungsi dengan benar.

- **Validation** : "Are we building the right product?"

  Validation merupakan kumpulan kegiatan yang berbeda untuk memastikan software yang telah dibangun dapat dilacak kembali sesuai permintaan konsumen

C. PIHAK YANG MELAKUKAN UJI COBA SOFTWARE

Seseorang yang membuat software tidak bisa diminta untuk mengujicoba software yang dibuatnya karna akan menimbulkan "conflict of interest". Sehingga butuh penguji coba software oleh pihak yang sama sekali tidak terlibat dalam pengembangan software terkait atau disebut Independent Test Group (ITG)

D. PANDUAN MELAKUKAN UJI COBA SOFTWARE

- Menentukan spesifikasi (requirements) produk dalam bentuk yang bisa diukur (quantitable)
- Menentukan tujuan ujicoba secara eksplisit
- Memahami pengguna serta membuat profil tiap kategori
- Mengembangkan rencana ujicoba yang mengutamakan segi waktu yang cepat
- Membangun software yang didesain untuk bisa mengujicoba dirinya sendiri(selftest)
- Menggunakan review teknis yang efektif sebagai filter
- Melakukan review teknis untuk menilai strategi ujicoba dan kasus ujicobanya
- Membangun perbaikan secara terus menerus sebagai proses ujicoba

E. STRATEGI UJI COBA SOFTWARE KONVENSIONAL

- **Unit testing (Uji coba unit)** : fokus kepada upaya verifikasi pada unit terkecil dari desai software, yaitu komponen/modul software. Apa saja yg ditest dalam uji coba unit?
  - Interface Modul dipastikan agar informasi dapat masuk dan keluar ke/ dari modul dengan cepat
  - Struktur data lokal dipastikan agar integritas data terjaga selama pengolahan berjalan
  - Semua alur melalui struktur pengendalian (control structure) dipastikan tercakup
  - Kondisi di perbatasan nilai data dipastikan tercakup
  - Seluruh pengendalian kesalahan tercakup

  Karena unit/komponen bukan program yang bisa berjalan sendiri (standalone program), maka diperlukan program penggerak (driver) yang akan memicu fungsi yang ada dalam unit/komponen.
- **Integration testing (Uji coba integrasi)** : teknik yg sistematis untuk menyusun arsitektur software dengan dalam waktu bersamaan mencoba untuk menemukan error yang muncul saat proses interfacing

  INTEGRASI top-down
  - Modul utama berfungsi sebagai driver dan semua modul di bawahnya sebagai stubs
  - Stube akan digantikan dengan fungsi yang sebenarnya secara bertahap
  - Uji coba dilakukan pada waktu komponen digabungkan
  - Ujicoba ulang(regeression test dilakukan untuk memastikan tidak ada kesalahan diakibatkan karena adanya alur data dan/ atau kontrol baru

  INTEGRASI top-up
  - Unit/komponen pada level bawah digabungkan dalam satu kumpulan (yang disebut cluster atau build atau modul)
  - Untuk melakukan uji coba, program driver dibuat untuk mengkoordinasikan kasus ujicoba input dan output
  - Kumpulan unit/komponen diujicoba

- Driver dihapus, dan cluster digabungkan dengan cluster-cluster yang lain, terus ke atas sampai seluruhnya terhubung

- **Regression testing (Uji coba ulang)** : dilakukan ketika modul digabungkan sehingga muncul alur data, I/O atau control logic yang baru
- **Ujicoba validasi**
  - ✓ Merupakan puncak kegiatan setelah seluruh unit/komponen selesai digabung dan diujicoba
  - ✓ Ujicoba fokus pada tindakan dan hasil yang bisa dilihat dari sisi pengguna
  - ✓ Kriteria validasi diambil dari software requirements spesification yang ditentukan berdasarkan permintaan (requirement) pengguna
  - ✓ Hasil ujicoba validasi:
    - Karakteristik fungsi/perdorma sesuai dengan spesifikasi dan bisa diterima
    - Penemuan perbedaan antara realisasi dengan spesifikasi dan daftar kekurangan dibuat kemudian diperbaiki atau negosiasi

SYSTEM IMPLEMENTATION AND SUPPORT

A. KONSTRUKSI DAN IMPLEMENTASI SISTEM

- **Konstruksi** sistem: Pembangunan, instalasi, dan ujicoba komponen sistem
- **Implementasi** sistem: Penyerahan sistem menuju produksi, hidup dan berjalan

B. TUGAS DALAM TAHAPAN KONSTRUKSI

1. Membangun dan ujicoba jaringan

Umumnya menggunakan jaringan yang ada. Jika memerlukan jaringan yang baru, harus diujicoba dulu sebelumnya. Peran: Designer jaringan, administrator jaringan, analis sistem.

2. Membangun dan ujicoba basis data

Ujicoba dilakukan dengan sampel data. Peran: Pengguna sistem, designer/programmer db, administrator db.

3. Instalasi dan ujicoba s/w baru

Jika perlu s/w baru (beli/sewa), harus di-install dan diujicoba dulu. Peran: Analis sistem, designer sistem, administrator jaringan, vendor/konsultan s/w, programmer.

4. Menulis dan ujicoba program baru
   - Membangun program secara in house
   - Mendaur ulang komponen yang ada
   - Menulis komponen baru
   - Ujicoba dan mendokumentasikan

Peran: Analis sistem, designer sistem, programmer.

C. TUGAS DALAM TAHAPAN IMPLEMENTASI

1. Melakukan ujicoba sistem

Ujicoba jaringan, basis data, s/w yang dibeli, s/w in-house baru, s/w yang sudah ada bisa bekerja bersama. Peran: Analis sistem, pembangun sistem (db, jaringan, programmer), pemilik dan pengguna sistem

2. Menyiapkan rencana konversi

Rencana konversi dari sistem lama ke baru. Peran: Analis sistem/manajer proyek, komite pengarah/pimpinan

3. Instalasi basis data

Memasang sistem basis baru dengan data dari sistem lama dan memastikan data telah dipindahkan dengan benar. Peran: Programmer, Analis/desainer sistem.

4. Melatih pengguna

Pengguna sistem diberikan pelatihan dan dokumentasi. Peran: Analis sistem, pemilik dan pengguna sistem

5. Berpindah ke sistem baru

Memindahkan kepemilikan sistem baru dari pengembang kepada pemilik/pengguna sistem. Peran: Analis sistem/manajer proyek, Pemilik sistem, Pengguna sistem.

D. UJICOBA PENERIMAAN SISTEM (USER ACCEPTANCE TEST)

- **Definisi**: Ujicoba yang dijalankan pada sistem final di mana pengguna melakukan ujicoba verifikasi, validasi, dan audisi
- **Ujicoba verifikasi**: Mencoba sistem dalam lingkungan simulasi (dengan data simulasi)
    - Memeriksa kesalahan dan kelalaian terhadap spesifikasi desain
    - Alpha testing
- **Ujicoba validasi:** Mencoba sistem dalam lingkungan yang sebenarnya (menggunakan data yang sebenarnya)
    - Memeriksa kinerja, perilaku pengguna, prosedur, backup dan recovery
    - Beta testing
- **Audisi**: Mengesahkan bahwa sistem bebas dari kesalahan dan kelalaian dan laik untuk dioperasikan

E. BEDA DUKUNGAN DAN OPERASI SISTEM

- Dukungan sistem: Dukungan sistem terhadap pengguna, termasuk pemeliharaan yang diperlukan untuk memberbaiki kesalahan, kelalaian atau permintaan baru
- Operasi sistem: Eksekusi proses bisnis sistem dan program aplikasi dari hari ke hari, minggu ke minggu, bulan ke bulan dan tahun ke tahun

F. AKTIVITAS DUKUNGAN SISTEM

- Pemeliharaan program: Memperbaiki “bugs” atau kesalahan yang terselip dalam proses pengembangan

- Pemulihan sistem: Pemulihan sistem dan data setelah terjadi kegagalan sistem
- Dukungan teknis: Bantuan kepada pengguna yang kurang memahami permasalahan teknis atau pada situasi yang belum terantisipasi
- Peningkatan sistem: Peningkatan sistem untuk menangani permasalahan bisnis atau teknis yang baru serta permintaan terhadap teknologi yang baru

Cara cepet jago koding lalu bikin aplikasi kayak Gojek, Tokped?

1. Sebelum belajar ambil air hangat
2. Siramkan ke wajah
3. Ingatkan diri sendiri bahwa belajar itu harus sabar.

UJIAN AKHIR SEMESTER GANJIL TAHUN AKADEMIK 2017/2018

==========

MATA KULIAH	: Analisis dan Perancangan Sistem Informasi
KELAS	: 3KS1, 3KS2, 3KS3
DOSEN PENGAMPU	: Yunarso Anang, Ph.D.
TANGGAL	: Kamis, 8 Februari 2018
WAKTU	: 110 menit
SISTEM	: Tutup Buku

==========

A. Soal pilihan ganda : Pilih satu jawaban yang Anda anggap paling tepat. (20 poin @ 2 poin)

1. Dari sudut pandang manajemen proyek, suatu proyek dikatakan berhasil jika :
   a. Sistem dibangun menggunakan pendekatan produk atau model
   b. Sistem dibangun menggunakan dana dalam jumlah yang telah dianggarkan
   c. Sistem selesai dibangun dalam waktu yang telah direncanakan
   d. Sistem dapat diterima oleh konsumen
   e. Semua yang ada di atas
2. Manakah di antara berikut ini yang menunjukkan pekerjaan dalam proyek dan hubungan di antara pekerjaan-pekerjaan tersebut dalam bentuk grafik jaringan?
   a. WBS
   b. *Gantt chart*
   c. *Line chart*
   d. PERT *chart*
   e. Bar *chart*
3. Berikut ini adalah panduan untuk memilih dan merekrut anggota tim proyek kecuali :
   a. Pertimbangan rencana ke depan
   b. Pertahankan agar jumlah anggota tim tetap sedikit untuk meminimalisir miskomunikasi antar sesama
   c. Pilih pekerjaan untuk tiap orang
   d. Pilih anggota tim dengan pengalaman pengembangan sistem yang sangat luas
   e. Upayakan keharmonisan tim dengan cara memilih anggota yang bisa bekerja dengan baik dengan yang lain
4. Dalam proses pengadaan atau pelelangan *software*. Manakah yang harus dilakukan pertama oleh pengembang?
   a. Meminta proposal dan penawaran harga
   b. Memastikan pernyataan dari vendor dan performanya
   c. Memberikan penghargaan pada vendor dan meminta laporan

d. Mempelajari kriteria teknis dan pilihannya
e. Bukan satu dari yang di atas

5. Manakah di antara berikut yang merupakan langkah pertama dalam perancangan sistem yang dibangun sendiri (*in-house development*)?
   a. Merancang arsitektur aplikasi
   b. Merancang interface sistem
   c. Merancang database sistem
   d. Membuat paket spesifikasi desain
   e. Memperbarui rencana proyek
6. Manakah di antara berikut yang mendefinisikan arsitektur dari suatu sistem informasi?
   a. Teknologi yang digunakan untuk merealisasikan *user interface*
   b. Teknologi yang digunakan untuk merealisasikan seluruh *software* yang akan dibangun sendiri (*in-house development*)
   c. Tingkat di mana sistem informasi akan terdistribusi atau terpusat
   d. Distribusi data yang tersimpan dalam jaringan
   e. Semua yang ada di atas
7. Manakah di antara berikut yang bukan merupakan *layer* aplikasi dalam sistem?
   a. *Database layer*
   b. *Presentation layer*
   c. *Data layer*
   d. *Application logic layer*
   e. *Data manipulation layer*
8. Manakah di antara berikut yang merupakan pilihan distribusi data?
   a. Simpan semua data dalam satu server
   b. Simpan beberapa *table* tertentu dalam server berbeda
   c. Simpan *subset* dari *table* tertentu dalam server berbeda
   d. Buat replikasi dari *table* tertentu atau *subsetnya* dalam server berbeda
   e. Semua yang di atas
9. Manakah di antara berikut yang bukan merupakan prinsip utama dalam rancangan *output*?
   a. *Ouput* komputer sebaiknya dirancang menggunakan alat otomatis
   b. Distribusi (atau akses) ke *output* komputer harus cukup untuk memenuhi kebutuhan semua pengguna terkait
   c. *Output* komputer harus sederhana dan mudah untuk dipahami
   d. *Output* komputer harus bisa diterima oleh pengguna sistem
   e. *Timing ouput* komputer harus tepat. Informasi yang dikeluarkan harus bisa didapat oleh pengguna pada saat yang diperlukan untuk pengambilan keputusan atau transaksi berikutnya

10. Manakah di antara berikut yang merupakan langkah-langkah dalam perancangan *output*?
    a. Perancangan semua *form* yang perlu dicetak lebih dahulu
    b. Perancangan, validasi, dan uji coba *output*
    c. Identifikasi *output* sistem dan *review* permintaan *logic*
    d. Menentukan permintaan *output* fisik
    e. Semua yang ada di atas

B. Soal benar salah : Tulis B jika Anda menganggap pertanyaan berikut benar atau S jika salah. (20 poin @ 2 poin)

1. Entri data (*data entry*) adalah proses mengubah sumber data atau dokumen menjadi bentuk yang bisa dibaca oleh komputer.
2. Contoh yang paling umum dari entri data adalah menggunakan *keyboard*.
3. Salah satu di antara prinsip utama dalam perancangan *input* adalah bagaimana menangkap variabel dan data konstan.
4. Pengguna pemula (*novice user*) umumnya akan meningkat menjadi pengguna ahli (*expert user*) melalui latihan dan pengalaman
5. Salah satu pertimbangan dalam desain *user interface* adalah dengan menghindari pemberian nilai awal atau jawaban (*default value*) yang harus di entri oleh user.
6. Diagram transisi status (*state transition diagram*) merupakan alat dalam perancangan *user interface* untuk mengkoordinasikan tampilan-tampilan dalam *user interface.*
7. Fase implementasi sistem mencakup pekerjaan pengembangan (*development*), instalasi (*installation*) dan uji coba (*testing*) komponen sistem.
8. Uji coba validasi sistem dijalankan dalam lingkungan yang menggunakan data yang sebenarnya.
9. Dalam konversi sistem secara paralel. Sistem lama dan baru dioperasikan bersamaan selama beberapa waktu.
10. Satu di antara uji coba validasi yang dilakukan dalam uji coba penerimaan sistem (*system acceptance testing*) adalah *human engineering test.*

C. Soal Esai

1. Sebutkan karakteristik besar dan karakteristik kecil dalam model kualitas yang diatur dalam ISO 2051:2011. Berikan penjelasan untuk masing-masing karakteristik besar. (15 poin)
2. Untuk mengukur karakteristik kualitas *reliability* atau kendala suatu sistem, kita dapat menggunakan metrik *mean-time-to-failure* (MTIF), *men-time-to-recover* (MITR), dan menghitung *mean-time-between-failure* (MTBF). Sebutkan definisi masing-masing metrik tersebut dan tuliskan rumus menghitung MTBF beserta simulasi perhitungannya menggunakan gambar. (15 poin)
3. Dalam uji coba *software*, Teknik yang digunakan adalah dengan membuat *driver* dan *stub.* Sebutkan definisi masing-masing alat tersebut beserta gambar yang menunjukkan kapan perlu membuat *driver* kapan perlu membuat *stub.* (15 poin)

4. Jelaskan apa yang dimaksud dengan *cyclomelic complexity,* untuk apa menghitungnya, dan apa kaitannya dengan kualitas program, serta tuliskan 3 rumus cara perhitungan dengan menggunakan alur program. (15 poin)

PEMBAHASAN UAS GANJIL 2017/2018

ANALISIS DAN PERANCANGAN SISTEM INFORMASI

A. Pilihan Ganda

1. E
2. D
3. D
4. D
5. A
6. E
7. A
8. D
9. A
10. E

B. Soal Benar Salah

1. B
2. B
3. S
4. B
5. S
6. B
7. B
8. B
9. B
10. B

C. Esai

1. Model kualitas yang diatur dalam ISO 2051:2011
   - *Functional suitability* : Sejauh mana produk atau sistem menyediakan fungsi yang memenuhi kebutuhan ketika digunakan dalam kondisi tertentu
     - *Functional completeness*
     - *Functional correctness*
     - *Functional appropriateness*
   - *Performance efficiency* : Performa relative terhadap jumlah sumber daya yang digunakan dalam kondisi tertentu
     - *Time behavior*

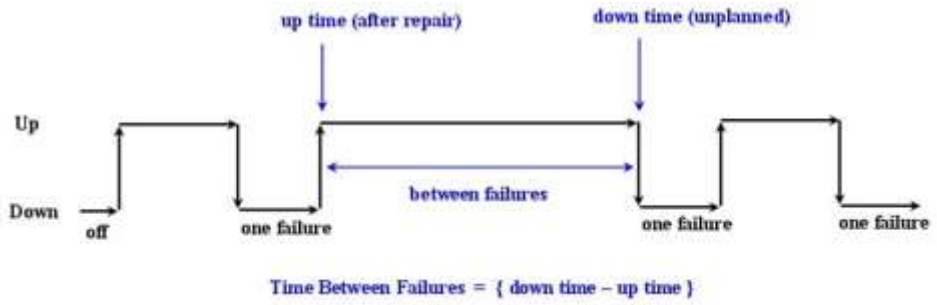
- *Resource utilization*
- *Capacity*

- *Compatibility*: Sejauh mana suatu produk, sistem atau komponen dapat bertukar informasi dengan produk, sistem atau komponen lain dan/atau melakukan fungsi yang diperlukan, ketika berbagi hardware atau software environment yang sama.
  - *Co-existence*
  - *Interoperability*
- *Usability*: sejauh mana suatu produk atau sistem dapat digunakan oleh pengguna tertentu untuk mencapai tujuan tertentu dengan efektivitas, efisiensi dan kepuasan dalam konteks penggunaan.
  - *Appropriateness recognizability*
  - *Learnability*
  - *Operability*
  - *User error protection*
  - *User interface aesthetics*
  - *Accessibility*
- *Reliability*: sejauh mana suatu sistem, produk atau komponen melakukan fungsi tertentu dalam kondisi yang ditentukan untuk jangka waktu tertentu
  - *Maturity*
  - *Availability*
  - *Fault tolerance*
  - *Recoverability*
- *Security*: sejauh mana suatu produk atau sistem melindungi informasi dan data sehingga orang atau produk atau sistem lain memiliki tingkat akses data yang sesuai dengan jenis dan tingkat otorisasinya.
  - *Confidentiality*
  - *Integrity*
  - *Non-repudiation*
  - *Accountability*
  - *Authenticity*
- *Maintainability* : tingkat efektivitas dan efisiensi suatu produk atau sistem dapat dimodifikasi oleh pengelola yang dimaksud
  - *Modularity*
  - *Reusability*
  - *Analyzability*
  - *Modifiability*
  - *Testability*

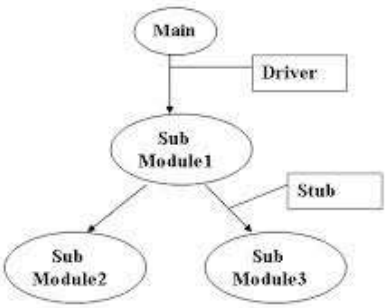
- *Portability* : tingkat efektivitas dan efisiensi suatu sistem, produk atau komponen dapat dikirim dari satu hardware, software atau lingkungan operasional ke yang lain.
  - *Adaptability*
  - *Installability*
  - *Replaceability*

2. Definisi masing-masing metrik
   - MTTF mengukur rata-rata waktu masa pakai sistem atau waktu yang dibutuhkan sampai sistem gagal (failure) dalam periode waktu tertentu.
   - MTTR mengukur rata-rata waktu yang dibutuhkan untuk memperbaiki sistem dan memulihkan secara utuh fungsionalitasnya dalam periode waktu tertentu.
   - MTBF mengukur rata-rata waktu yang dibutuhkan antara satu kegagalan dengan kegagalan berikutnya dalam periode waktu tertentu.

$$MTBF = \frac{\sum(downtime - uptime)}{banyak\ kegagalan}$$



3. Definisi *stub* dan *driver*
   - *Stub* digunakan dalam pendekatan top - down testing , yaitu ketika modul utama siap untuk diuji tetapi sub-modulnya belum siap diuji. Stub sering disebut "*called program* ", yaitu program yang dipanggil dalam menguji fungsi modul utama.
   - *driver* digunakan dalam pendekatan bottom - up testing , yaitu ketika sub-modul siap untuk diujii tapi modul utamanya belum siap diuji. Driver sering disebut "*calling program*"

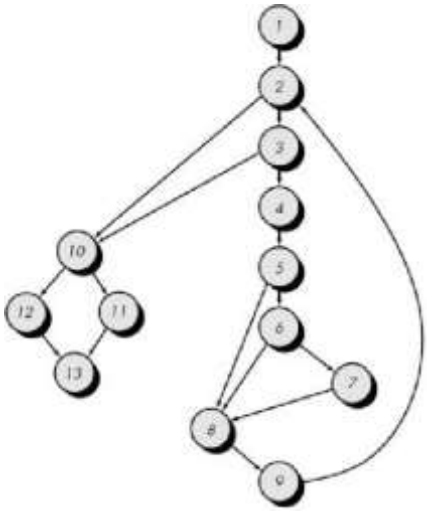


4. *Cyclomatic complexity* adalah sebuah *software metric* yang menyediakan ukuran kuantitatif dari kompleksitas logika dari suatu program. Dengan menggunakan hasil perhitungan *cyclomatic complexity*, kita dapat menentukan apakah sebuah program merupakan program yang sederhana atau kompleks berdasarkan logika yang diterapkan pada program tersebut. Apabila dikaitkan dengan *software testing*, *cyclomatic complexity* dapat digunakan untuk menentukan berapa minimal *test case* yang harus dijalankan untuk menguji sebuah program dengan menggunakan teknik basis *path testing*. Contoh: program penghitungan rata-rata 100 angka atau kurang dari array

```
INTERFACE RETURNS average, total.input, total.valid;
INTERFACE ACCEPTS value, minimum, maximum;

TYPE value[1:100] IS SCALAR ARRAY;
TYPE average, total.input, total.valid;
     minimum, maximum, sum IS SCALAR;
TYPE I IS INTEGER;
I = 1;
Total.input = total.valid = 0;
Sum = 0;
DO WHILE value[i] <> -999 AND total.input < 100
   Increment total.input by 1;
   IF value[i] > = minimum AND value[i] < maximum
        THEN increment total.valid by 1;
                sum = sum + value[i]
        ELSE skip
   ENDIF
   Increment i by 1;
ENDDO
IF total.valid > 0
     THEN average = sum/total.valid;
   ELSE average =-999;
ENDIF
```

Berdasarkan program diatas dibuatlah flowgraph seperti berikut



Cara penghitungan *Cyclomatic complexity* :

1) V(G) = E – N + 2

   V(G) : *Cyclomatic complexity*

E : jumlah edge

N : jumlah Node

V(G) = 17 edges – 13 nodes + 2 = 6

2) Menentukan jumlah *region* dari *flow graph*

V(G) = 6 region

3) Menentukan jumlah node bercabang ditambah dengan angka 1.

Node bercabang = 5 (node 2, 3, 5, 6, 10)

V(G) = 5 + 1 = 6

UJIAN AKHIR SEMESTER GANJIL TAHUN AKADEMIK 2018/2019

==========

MATA KULIAH	: Analisis dan Perancangan Sistem Informasi
KELAS	: 3KS1, 3KS2, 3KS3
DOSEN PENGAMPU	: Yunarso Anang, Ph.D.
TANGGAL	: Jumat. 21 Desember 2018
WAKTU	: 110 menit
SISTEM	: Boleh membuka buku dan Internet kecuali sosial media dan sistem pengiriman online

==========

TIDAK DIPERKENANKAN MENAMBAH LEMBAR JAWABAN

Anda ditunjuk untuk memimpin proyek pengembangan sistem ujian masuk berbasis komputer (selanjutnya disebut UMBK) untuk seleksi penerimaan mahasiswa baru di STIS (selanjutnya disebut SPMB). Seperti diketahui bersama, SPMB sebelumnya menggunakan lembar jawaban komputer (LJK), di mana lembar ujian dan LJK dicetak sebanyak peserta ujian plus cadangannya, lalu didistribusikan ke lokasi ujian di seluruh ibukota provinsi plus dua kabupaten di seluruh Indonesia. Ujian diselenggarakan serentak dalam satu waktu di seluruh Indonesia. Setelah ujian selesai, LJK dikumpulkan lalu dibawa kembali ke STIS untuk diolah dan diumumkan hasilnya. Anda diminta untuk merencanakan, merancang jadwal kegiatan mulai dari persiapan, pengembangan sistem, uji coba, sampai pelaksanaan, merancang dan implementasi sistem UMBK, dan memimpin seluruh jalannya proyek.

Tuliskan semua yang Anda akan dan Anda kerjakan pada setiap fase, mulai dari perencanaan sampai dengan pelaksanaan, dengan rinci namun ringkas dan jelas, memperhatikan seluruh keterbatasan dan keunikan dari penyelenggaraan ujian berbasis komputer ini. Anda harus menuliskan semua itu dalam 1 lembar jawaban folio yang diberikan. **Tidak diperkenankan menambah lembar jawaban.**

PEMBAHASAN UAS GANJIL 2018/2019

ANALISIS DAN PERANCANGAN SISTEM INFORMASI

Dalam perencanaan dan penerapan sebuah sistem informasi dalam hal ini Sistem Ujian Masuk Berbasis Komputer (UMBK) SPMB STIS sebaiknya disusun menjadi empat tahapan, di antaranya Inisiasi Sistem, Analisis Sistem, Desain Sistem, dan Implementasi Sistem.

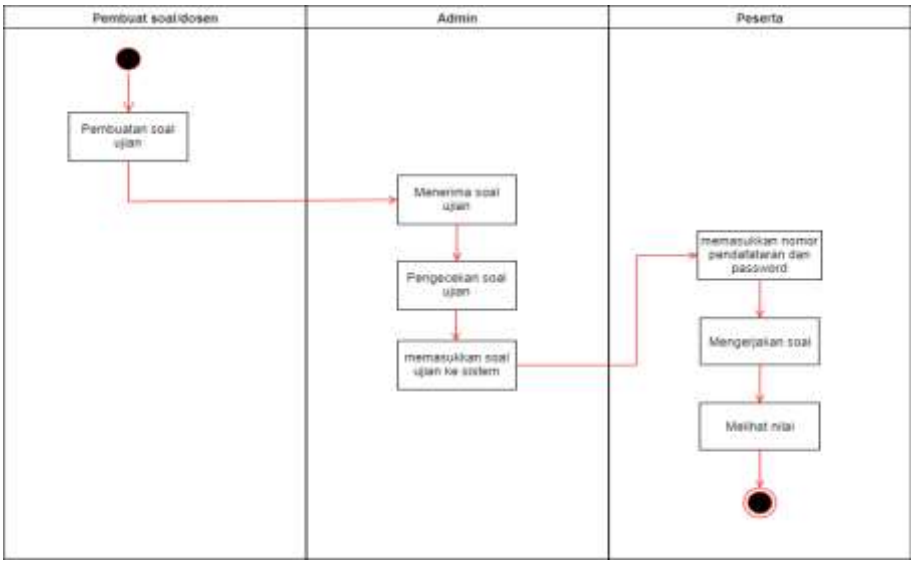
Tahap yang paling awal adalah inisiasi sistem. Tahap ini melibatkan tiga pemangku kepentingan, yaitu pemilik sistem (dalam hal ini adalah Direktur Polstat STIS), manajer proyek (saya), analis sistem, peserta ujian, pihak bank, dan kemenpan(sebagai tempat pertama registrasi pendaftaran). Ketiga pemangku kepentingan seperti pemilik sistem, manajer proyek, dan analis sistem berdiskusi untuk menentukan cakupan, tujuan sistem, serta jadwal dan biaya proyek pembangunan sistem ini ke depan. Cakupan dan tujuan ditentukan berdasarkan masalah yang terjadi pada sistem lama. Dalam hal ini digunakan metode PIECES.

Dalam hal *performance,* sistem lama yang menggunakan LJK memerlukan waktu yang lama untuk mendapatkan hasil ujian SPMB. Dari sisi *information,* sistem lama hanya bisa mendapatkan informasi yang sedikit dari hasil ujian peserta, yakni data diri dan jawaban yang dipilih tanpa ada informasi lama waktu peserta menjawab yang dapat digunakan sebagai evaluasi kualitas soal. Dari aspek *economy,* jelas memerlukan biaya yang besar dalam hal pencetakan, pendistribusian, sampai pengiriman LJK kembali ke pusat untuk dikoreksi. Sementara dalam hal *control,* terjaminnya soal dan jawaban sampai dengan selamat dan aman ke tangan peserta tanpa dibuka di tengah jalan masih kurang. Dari sisi *efficiency,* jelas sistem yang lama masih tidak efisien. Dari aspek *service*, peserta SPMB dengan sistem lama masih terdapat kemungkinan jawaban tidak masuk ke sistem akibat kesalahan teknis seperti jawaban tidak dapat di-scan, LJK yang rusak, atau salah mengisi biodata diri. Dari paparan pemilik sistem tentang permasalahan yang ada lalu ditentukan cakupan/batasan sistem yang akan dibangun, biaya yang ditawarkan, serta kapan proyek ditargetkan selesai.

Dari cakupan yang didapat pada tahap sebelumnya, manajer proyek bersama dengan analis sistem dan pengguna sistem (dalam hal ini panitia SPMB) membahas cakupan tersebut. Cakupan-cakupan tersebut dibuat menjadi lebih rinci lagi, lalu dianalisis apakah benar-benar dibutuhkan pengguna sistem (dalam hal ini panitia SPMB) atau tidak. Setelah itu, kebutuhan pengguna diurutkan kembali berdasarkan prioritas, mana yang lebih didahulukan. Cakupan-cakupan tersebut kemudian dianalisis untuk dicari pemecahan masalah lewat sebuah sesi JRP (*Joint Requirement Planning*).

Hasil dari JRP yang berupa dokumen kebutuhan pengguna kemudian dibuat diagram *use case* yang menggambarkan interaksi antara pengguna sistem (panitia dan peserta SPMB) dan sistem UMBK itu sendiri. Tujuan pemodelan *use case* agar kebutuhan sistem lebih mudah dimengerti. Selain diagram *use case,* dibuat pula *Entity Relationship Diagram (ERD)* yang menggambarkan hubungan antara entitas, *Data Flow Diagram (DFD)* yang menggambarkan proses atau aliran data dalam sistem UMBK,

dan juga *Unified Modelling Language (UML)* berupa *Activity Diagram* yang menggambarkan bagaimana aktivitas pengguna UMBK dari mulai hingga selesai.



Langkah terakhir dari analisis sistem adalah menganalisis kelayakan sistem tersebut. Manajer proyek bersama analis sistem mempertimbangkan kebutuhan pengguna yang ada dengan jadwal proyek serta biaya yang ditentukan apakah layak untuk dilanjutkan, perlu dilakukan revisi cakupan, biaya, dan jadwal, atau tidak dilanjutkan. Apabila proyek sistem UMBK ini dinilai layak kemudian dibuat sebuah proposal yang akan diserahkan ke pemilik sistem.

Setelah proposal disetujui, maka tahap selanjutnya adalah mendesain sistem secara lebih teknis. Pada tahap ini juga dipertimbangkan apakah sistem menggunakan *software* yang dibangun sendiri, menggunakan *open source software*, atau harus membeli *software* yang sudah ada. Dalam hal ini akan dipertimbangkan apakah membuat *software* ujian online sendiri atau menggunakan *open source* software misalnya TCExam. Misalkan setelah mempertimbangkan banyak hal, manajer proyek memutuskan untuk menggunakan TCExam. Maka langkah selanjutnya, desainer sistem tinggal mengembangkan rancangan arsitektur aplikasi, desain *database*, serta *user interface* yang sudah dibuat oleh pengembang TCEXam. Desain-desain tersebut dilakukan penyesuaian sesuai kebutuhan dari proses bisnis sistem UMBK.

Apabila desain sistem UMBK telah rampung, langkah selanjutnya adalah membangun sistem tersebut. Pembangunan sistem meliputi tiga tahap, yaitu membangun dan menguji jaringan, membangun dan menguji database, menginstall dan mengkonfigurasi software. Pada tahap pertama, jaringan dibangun lalu diuji apakah jaringan yang dibuat aman dari serangan. Selanjutnya, membangun database dan menguji apakah terdapat kesalahan dalam struktur database. Langkah terakhir adalah menginstall TCExam dan melakukan beberapa konfigurasi.

Setelah sistem dibangun, maka langkah terakhir adalah mengimplementasi sistem. Pada tahap ini database diinstall serta melatih pengguna sistem agar dapat menyesuaikan dengan sistem yang baru. Lalu menyiapkan beberapa perangkat yang dibutuhkan seperti komputer server yang memiliki spesifikasi *processor* minimal *quad core* 64 bit, RAM minimal 8 GB, DDR 3, kapasitas *hard disk* minimal

250 GB, dan 1 server cadangan. Lalu menyiapkan komputer peserta dengan *processor* minimal *dual core*, RAM minimal 2GB, dan sistem operasi windows XP/Windows 7/Windows 8/Linus/MAC. Hak akses yang mempunyai hak dalam melakukan pengaksesan sistem hanya admin sistem.

UJIAN AKHIR SEMESTER GANJIL TAHUN AKADEMIK 2019/2020

==========

MATA KULIAH	: Analisis dan Perancangan Sistem Informasi
KELAS	: 3SD1, 3SD2, 3SI1
DOSEN PENGAMPU	: Yunarso Anang, Ph.D.
TANGGAL	: Selasa, 17 Desember 2019
WAKTU	: 110 menit
SISTEM	: Tutup Buku

==========

Instruksi: rencanakan baik-baik apa yang akan dituliskan pada lembar jawaban folio yang disediakan. DILARANG menambah lembar jawaban. Mengerjakan soal tidak harus berurutan, namun tuliskan jawabannya pada tempat yang berurutan dalam lembar jawaban.

1. Agar karakteristik kualitas dari suatu produk piranti lunak (*software*) dapat ditaksir atau dinilai (*to be assessed*), misalnya *usability* atau "kemudahan untuk digunakan", maka karakteristik tersebut harus diturunkan terlebih dahulu, kemudian anda harus dapat menunjukkan karakteristik kualitas tersebut secara spesifik dan dapat diukur (atau minimal bisa dikenali). sebagai contoh, *intuitiveness* atau seberapa mudah suatu produk software memiliki *user interface* yang mudah diikuti sehingga pengguna pemula pun dapat menggunakan tanpa perlu melakukan belajar atau latihan, dapat ditunjukkan sebagai:
   - Apakah layoutnya konduktif sehingga mudah untuk dipahami?
   - Apakah pengoperasiannya mudah ditemukan dan dilakukan?
   - Apakah menggunakan metafor yang dikenal?
   - Apakah penggunaan *key stroke* dan *mouse click* ekonomi (tidak banyak)?
   - Apakah estetika membantu dalam pemahaman dan penggunaan?

   Kerjakan perintah berikut:

   Untuk karakteristik kualitas ***maintainability*** atau "kemudahan untuk dilakukan pemeliharaan", turunkan menjadi salah **satu faktor** yang dapat menunjukkan *maintainability* (seperti faktor *intuitiveness* menunjukkan *usability* dalam contoh di atas). Lalu, seperti contoh di atas, tunjukkan bagaimana faktor tersebut dapat ditaksir secara spesifik dan dapat diukur (atau minimal bisa dikenali). Tuliskan minimal 3 item.

2. Anda pasti sudah ditentukan keanggotaannya dalam kepanitiaan PKL, di mana anda terlibat dalam salah satu seksi atau kelompok. Agar kegiatan dapat berjalan dengan lancar, anda harus tahu tentang proyek atau kegiatan dalam seksi atau kelompok anda.

   Kerjakan perintah berikut:

a. Identifikasi *task* apa saja yang harus dikerjakan

b. Untuk tiap *task*, tentukan awal dan akhir atau durasi

c. Tentukan *precedence* atau urutan dan hubungan antar task di atas

d. Buat PERT dan Gantt Chart

3. Diberikan potongan *source code* program berikut:

```
public void GenerateId(int start, int end){
  if (start < 0) {
    throw new ArgumentOutRangeExeption("start", start, "should not less than zero");
  }
  if (end < 0) {
    throw new ArgumentOutRangeExeption("end", end, "should not less than zero");
  }
  int paramLength = (int)Math.Max(Math.Floor(Math.Log10(start) + 1), Math.Log10(end) + 1);
  if (paramLength > this.NumberOfRandomDigits) {
    throw new ArgumenException("invalid number of digits");
  }
  if (start > end) {
    int tmp = start;
    start = end;
    end= tmp;
  }
  int numberOfDigits = this.NumberOfRandomDigits;
  string format = this.GetAllDigitsFormat();
  long idpart = start;
  while (idpart <= end) {
    long id = GenerateId((long) Math.Pow(10, numberOfDigits) * _prefix + idpart);
    Console.WriteLine("{0:"+format+"}", id);
    idpart++;
  }
}
```

Kerjakan perintah berikut:

a. Buat *flow-chart* program di atas. Beri nomor untuk setiap elemen (proses atau cabang) alur.

b. Buat *flow-graph* yang sesuai dengan *flow-chart* di atas. Pada setiap *node*, tuliskan nomor elemen yang sesuai.

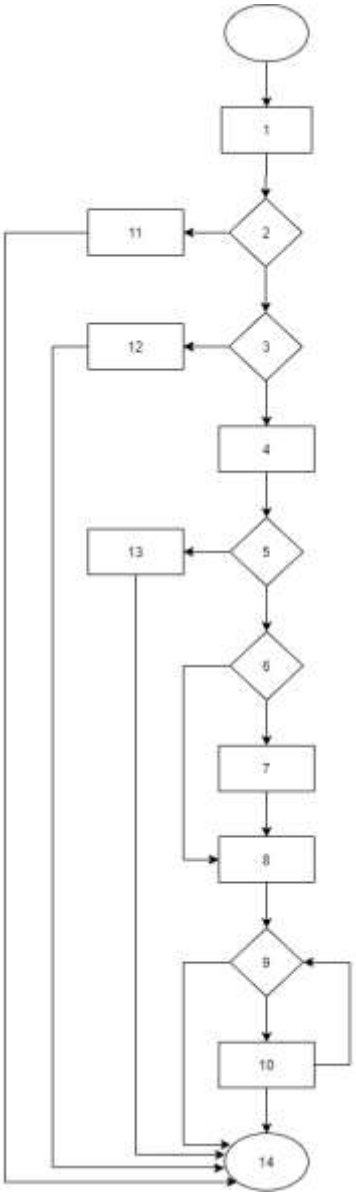
c. Hitung *cyclomatic complexity* menggunakan 3 rumus yang ada.

PEMBAHASAN UAS GANJIL 2019/2020

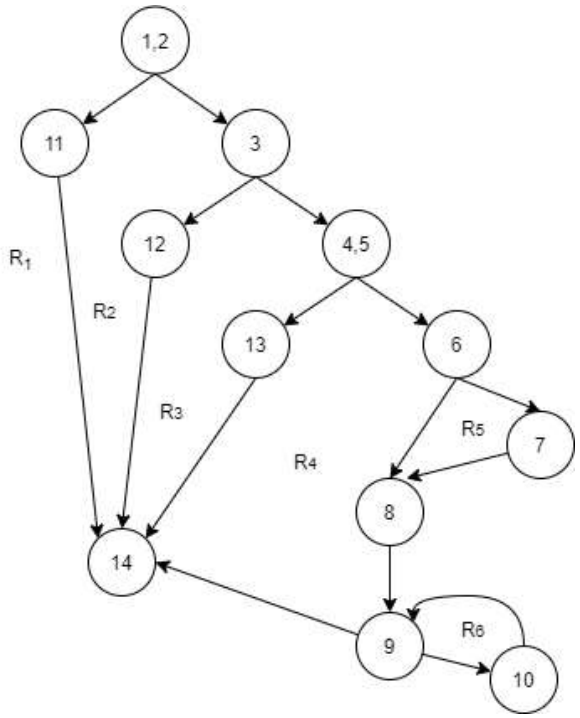
ANALISIS DAN PERANCANGAN SISTEM INFORMASI

1. Karakteristik maintainability atau "kemudahan untuk dilakukan pemeliharaan" dapat diturunkan menjadi salah satu faktor yang dapat menunjukkan *maintainability*, yaitu testablity. Faktor tersebut dapat ditaksir atau diukur secara spesifik sebagai berikut:
   - Apakah perangkat lunak yang memungkinkan modifikasi dapat divalidasi?
   - Apakah program dapat melakukan fungsi yang diharapkan?
   - Apakah perangkat lunak memfasilitasi pelaksanaan keberhasilan tes (yaitu, tes yang akan menyebabkan kegagalan yang disebabkan semua cacat yang ada)?

3a. *flow chart* dari *source code* pada soal



3b. *flow graph*



- Independent program path

  Path 1 : 1-2-11-14

  Path 2 : 1-2-3-12-14

  Path 3 : 1-2-3-4-5-13-14

  Path 4 : 1-2-3-4-5-6-8-9-14

  Path 5 : 1-2-3-4-5-6-7-8-9-14

  Path 6 : 1-2-3-4-5-6-8-9-10-9-14

3c. Menghitung *cyclomatic complexity*

4) V(G) = E – N + 2

   V(G) : *Cyclomatic complexity*

   E : jumlah edge

   N : jumlah Node

   V(G) = 16 edges – 12 nodes + 2 = 6

5) Menentukan jumlah *region* dari *flow graph*

   V(G) = 6 region

6) Menentukan jumlah node bercabang ditambah dengan angka 1.

   Node bercabang = 5 (node 2, 3, 5, 6, 9)

   V(G) = 5 + 1 = 6