Software Quality Assurance

Pertemuan 6
Software Requirements and SQA
PMPL pada Kebutuhan Perangkat Lunak

Tim Dosen PMPL INF UAJY

Materi Hari Ini

- Pada fase ini, kita akan membahas pentingnya, dampak, praktik, dan teknik penjaminan mutu perangkat lunak yang terkait dengan berbagai aktivitas dalam proses pengembangan perangkat lunak.
- Kita akan mulai dengan kebutuhan

Poin Penting

- Kita sedang mendiskusikan kebutuhan dengan mengacu pada penjaminan mutu dan manajemen
- Setiap item yang kita diskusikan, coba dikaitkan dengan masalah penjaminan mutu.

Pentingnya Kebutuhan

- Bagian tersulit dari membangun sistem perangkat lunak adalah memutuskan apa yang akan dibangun... Tidak ada bagian lain dari pekerjaan yang melumpuhkan sistem yang dihasilkan jika dilakukan dengan salah. Tidak ada bagian lain yang lebih sulit untuk diperbaiki nanti.
  - Fred Brooks

Permasalahan Kebutuhan - 1

- Kebutuhan tidak mencerminkan kebutuhan pelanggan yang sebenarnya untuk sistem
- Kebutuhan tidak konsisten dan/atau tidak lengkap
- Sangat mahal untuk membuat perubahan pada kebutuhan setelah disetujui

Permasalahan Kebutuhan - 2

- Ada kesalahpahaman antara pelanggan, orang yang mengembangkan kebutuhan sistem, dan orang yang mengembangkan atau memelihara sistem
- Kebutuhan ditulis menggunakan klausa kondisional yang kompleks (jika A maka B maka C…), yang membingungkan
- Menggunakan terminologi dengan cara yang ceroboh dan tidak konsisten
- Penulis kebutuhan berasumsi bahwa pembaca memiliki pengetahuan khusus tentang domain atau sistem, dan tidak menuliskan informasi penting pada dokumen kebutuhan

Dampak dari Kebutuhan yang Salah

- Sulit untuk memeriksa error dan kelalaian pada kebutuhan
- Interpretasi yang berbeda dari kebutuhan dapat menyebabkan ketidaksepakatan kontrak antara pelanggan dan pengembang sistem
- Ketika terjadi kesalahan pada kebutuhan, maka sistem jadi terlambat, tidak dapat diandalkan, dan tidak memenuhi kebutuhan pelanggan
- Hal ini mengakibatkan hilangnya waktu, pendapatan, pangsa pasar, dan kepercayaan pelanggan yang sangat besar
- Kesesuaian dengan kebutuhan yang salah akan menghasilkan sistem yang salah

Cacat Kebutuhan- 1

- Keempat kategori cacat ditemukan dalam kebutuhan
  - Kesalahan karena komisi
  - Kesalahan karena kelalaian
  - Kesalahan karena kejelasan dan ambiguitas
  - Kesalahan karena kecepatan dan kapasitas
- Jika tidak dicegah atau dihilangkan, cacat kebutuhan biasanya mengalir ke hilir ke dalam desain, kode, dan manual pengguna
- Secara historis, cacat kebutuhan adalah yang paling mahal dan sulit untuk dihilangkan
- Setiap upaya harus dilakukan untuk menghilangkan cacat kebutuhan
- Untuk kesalahan kebutuhan, pencegahan biasanya bekerja lebih baik daripada penghapusan

Cacat Kebutuhan - 2

- Kesalahan kelalaian adalah yang paling umum di antara cacat kebutuhan
- Contoh terkenal adalah masalah Y2K
- Kesalahan kedua yang paling umum adalah kesalahan kejelasan dan ambiguitas
- Terutama, karena bahasa alami (seperti bahasa Inggris) digunakan untuk menyatakan kebutuhan, sementara bahasa tersebut sendiri ambigu
- Misalnya: objek
- Kesalahan komisi juga dapat ditemukan dalam dokumen kebutuhan
- Kesalahan kecepatan dan kapasitas juga dapat ditemukan di dokumen kebutuhan

Pencegahan Vs Penghapusan

- Untuk kesalahan kebutuhan, pencegahan biasanya lebih efektif daripada penghapusan
- Joint application development (JAD), Quality Function Deployment (QFD), dan pembuatan prototipe lebih efektif dalam pencegahan cacat
- Inspeksi kebutuhan dan pembuatan prototipe memainkan peran penting dalam penghapusan cacat

Kebutuhan yang Berubah/Merayap

- Kebutuhan akan berubah, apa pun yang terjadi
- Masalah utama dalam rekayasa kebutuhan adalah tingkat perubahan kebutuhan setelah fase kebutuhan "resmi" berakhir (kebutuhan ‘merayap’)
- Tingkat perubahan ini rata-rata 3% per bulan pada fase desain berikutnya, dan akan turun setelah itu
- Tingkat perubahan ini harus turun menjadi 1% per bulan selama pengkodean
- Idealnya, seharusnya tidak menyebabkan perubahan dalam pengujian

Cacat dan Kebutuhan ‘Merayap’

- Penelitian telah menunjukkan bahwa persentase yang sangat signifikan dari cacat yang dikirim dapat ditelusuri kembali ke kebutuhan pengguna yang ‘merayap’
- Realisasi ini hanya dapat dilakukan, jika pelacakan cacat, ketertelusuran kebutuhan, efisiensi penghilangan cacat, dan tingkat cacat semuanya dipantau

Kontrol Kerusakan Kebutuhan 'Merayap'

- Mekanisme penjaminan mutu berikut dapat membatasi kerusakan yang dilakukan oleh kebutuhan 'merayap'
  - Prosedur manajemen perubahan yang formal
  - Alat kontrol konfigurasi canggih
  - Inspeksi desain dan kode yang formal

Masalah dengan Bahasa Alami - 1

- Kurang kejelasan
- Kebingungan kebutuhan
- Penggabungan kebutuhan

Masalah dengan Bahasa Alami - 2

- Pemahaman bahasa alami bergantung pada spesifikasi pembaca dan penulis yang menggunakan kata-kata yang sama untuk konsep yang sama
- Spesifikasi kebutuhan menggunakan bahasa alami terlalu fleksibel. Anda dapat mengatakan hal yang sama dengan cara yang sangat berbeda

Masalah dengan Bahasa Alami - 3

- Tidak mungkin untuk memodulasi kebutuhan dalam bahasa alami. Mungkin sulit untuk menemukan semua kebutuhan terkait
  - Untuk mengetahui dampak perubahan, setiap kebutuhan harus diperiksa

Dokumen Kebutuhan

- Dokumen kebutuhan adalah dokumen formal yang digunakan untuk mengkomunikasikan kebutuhan kepada pelanggan, engineers, dan manager
- Ini juga dikenal sebagai spesifikasi kebutuhan perangkat lunak atau SKPL
- Dokumen kebutuhan biasanya ditulis dalam bahasa alami (seperti, Inggris, Jepang, Prancis, dll.), yang ambigu

Apa yang Seharusnya Tidak Termasuk dalam SKPL?

- Kebutuhan proyek (misalnya, staf, jadwal, biaya, pencapaian, aktivitas, fase, prosedur pelaporan)
- Desain
- Rencana penjaminan produk (misalnya, rencana manajemen konfigurasi, rencana verifikasi dan validasi, rencana pengujian, rencana penjaminan mutu)

Pengguna Dokumen Kebutuhan

- System customers
- Managers
- System engineers
- System test engineers
- System maintenance engineers

Kebutuhan untuk Dokumen Kebutuhan - 1

- Seharusnya hanya menentukan perilaku eksternal
- Harus menentukan batasan pada implementasi
- Seharusnya mudah diubah
- Harus berfungsi sebagai alat referensi untuk memantain sistem

Kebutuhan untuk Dokumen Kebutuhan - 2

- Harus merekam pemikiran sebelumnya tentang siklus hidup sistem
- Harus mencirikan tanggapan yang dapat diterima untuk peristiwa yang tidak diinginkan

- Heninger (1980)

Organisasi Dokumen Kebutuhan?

- Klien/pengembang mungkin memiliki cara sendiri untuk mengorganisasi SKPL
- Departemen Pertahanan AS
- NASA
- Standar IEEE/ANSI 830-1998

IEEE/ANSI Standard 830-1998

Table 5.2. Organization of the SRS [IEEE98a]

1. Introduction
   - 1.1 Purpose0
   - 1.2 Scope
   - 1.3 Definitions, Acronyms and Abbreviations
   - 1.4 References
   - 1.5 Overview
2. The Overall Description
   - 2.1 Product Perspective
     - 2.1.1 System Interfaces
     - 2.1.2 Interfaces
     - 2.1.3 Hardware Interfaces
     - 2.1.4 Software Interfaces
     - 2.1.5 Communications interfaces
     - 2.1.6 Memory Constraints
     - 2.1.7 Operations
     - 2.1.8 Site Adaptation Requirements
   - 2.2 Product Functions
   - 2.3 User Characteristics
   - 2.4 Constraints
   - 2.5 Assumptions and Dependencies
   - 2.6 Apportioning of Requirements
3. Specific Requirements
   - 3.1 External interfaces
   - 3.2 Functions
   - 3.3 Performance Requirements
   - 3.4 Logical Database Requirements
   - 3.5 Design Constraints
     - 3.5.1 Standards Compliance
   - 3.6 Software System Attributes
     - 3.6.1 Reliability
     - 3.6.2 Availability
     - 3.6.3 Security
     - 3.6.4 Maintainability
     - 3.6.5 Portability
   - 3.7 Organizing the Specific Requirements
     - 3.7.1 System Mode
     - 3.7.2 User Class
     - 3.7.3 Objects
     - 3.7.4 Feature
     - 3.7.5 Stimulus
     - 3.7.6 Response
     - 3.7.7 Functional Hierarchy
   - 3.8 Additional Comments
4. Change Management Process
5. Document Approvals
6. Supporting Information

Verifikasi pada Dokumen Kebutuhan (SKPL)

- Verifikasi kebutuhan memiliki potensi tinggi untuk mendeteksi bug.
- Dokumen SKPL direview oleh orang menggunakan metode verifikasi apa pun (seperti walkthrough, inspeksi, dll.).
- Dapat menggunakan inspeksi karena efektivitas dan kemampuannya untuk menghasilkan hasil yang baik.
- Mungkin melakukan review dua kali atau bahkan lebih sering. Setiap review akan meningkatkan kualitas dokumen tetapi dapat menghabiskan sumber daya dan meningkatkan biaya pengembangan perangkat lunak.

Checklist

- Checklist adalah alat verifikasi populer yang terdiri dari daftar konten penting yang harus disertakan pada dokumen/program.
- Checklist digunakan untuk mencari duplikat informasi, informasi yang hilang, informasi yang tidak jelas, informasi yang salah, dll.
- Checklist digunakan ketika proses review dan membuat proses review lebih terstruktur dan efektif.

Atribut Kualitas Dokumen Kebutuhan - 1

- Correct (Benar)
- Unambiguous (Tidak ambigu)
- Complete (Lengkap)
- Consistent (Konsisten)
- Verifiable (Dapat diverifikasi)
- Modifiable (Dapat dimodifikasi)
- Traced (Dilacak)
- Traceable (Dapat dilacak)

Atribut Kualitas Dokumen Kebutuhan - 2

- Feasibility (Kelayakan)
- Dapat dipahami oleh pelanggan
- Desain independen
- Annotated (Beranotasi)
- Concise (Ringkas)
- Organized (Terorganisir)

Correct (Benar)

- SKPL disebut benar jika dan hanya jika setiap kebutuhan yang dinyatakan di dalamnya mewakili sesuatu yang diperlukan dari sistem yang akan dibangun
- SKPL disebut benar jika dan hanya jika setiap kebutuhan yang telah ditetapkan telah tercantum dalam dokumen.

Unambiguous (Tidak Ambigu)

- SKPL tidak ambigu jika dan hanya jika setiap kebutuhan yang disebutkan hanya memiliki satu interpretasi
- Semua istilah dengan banyak arti harus dituliskan dalam glosarium
- SKPL tidak boleh membingungkan baik bagi pembuat ataupun yang akan menggunakannya.
- Semua bahasa alami mengundang ambiguitas

Contoh Ambiguitas

- “Pesawat yang tidak bersahabat dan memiliki misi yang tidak diketahui atau berpotensi memasuki wilayah udara terbatas dalam waktu 5 menit akan menyalakan peringatan”
- Kombinasi "dan" dan "atau" menjadikan ini kebutuhan yang ambigu

Complete (Lengkap)

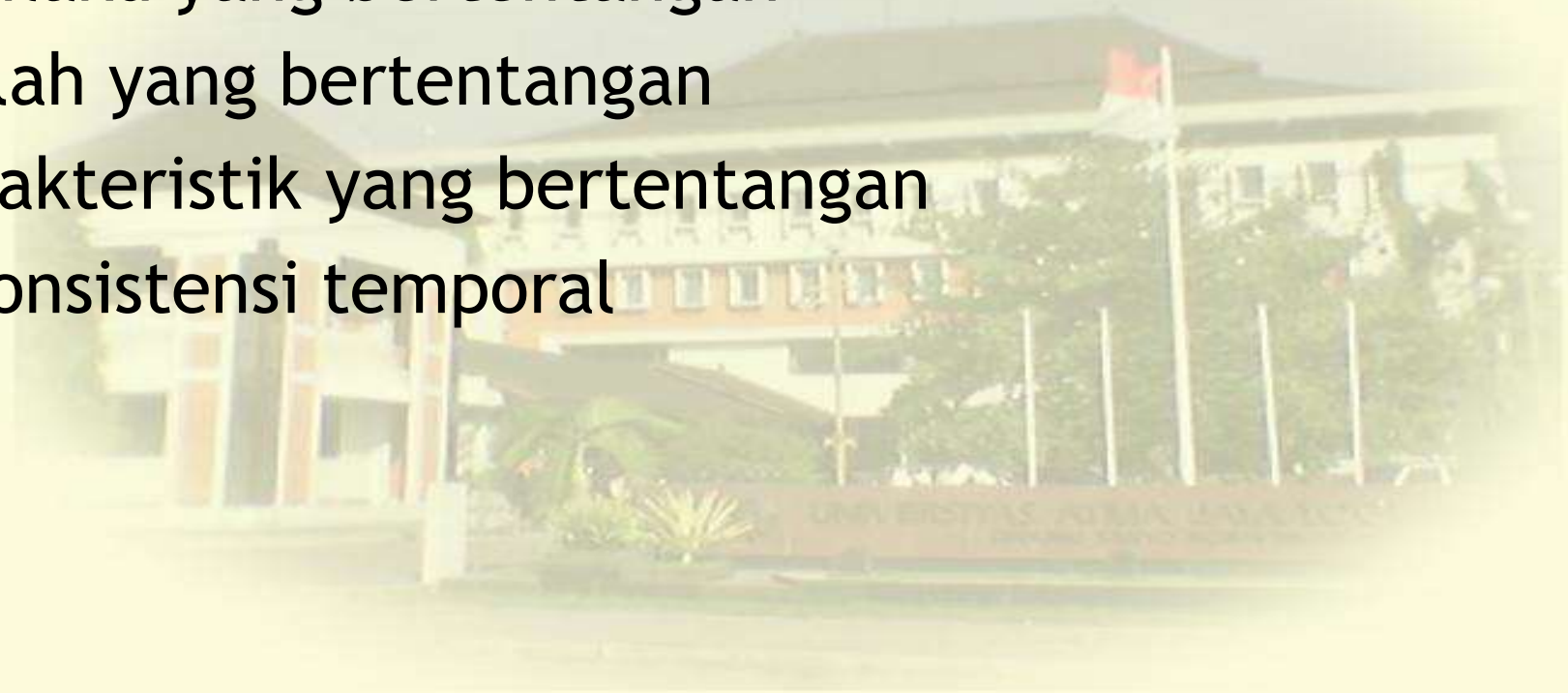
- Segala sesuatu yang seharusnya dilakukan oleh perangkat lunak tertulis dalam SKPL. Semua kebutuhan-kebutuhan sudah tercakup (fungsional & non fungsional).
- Semua definisi masukan pada berbagai kondisi sudah terpenuhi, tanggapan dari setiap input yang mungkin (valid & tidak valid) ke perangkat lunak telah ditentukan.
- Semua halaman diberi nomor; semua gambar dan tabel diberi nomor, nama, dan referensi; dan disertai dengan semua istilah yang digunakan dan unit yang digunakan sebagai pengukuran (bila ada).
- Tidak ada bagian yang ditandai “To Be Determined” (TBD)

Consistent (Konsisten) - 1

- SKPL disebut konsisten jika dan hanya jika:
  - Tidak ada kebutuhan yang disebutkan yang bertentangan dengan dokumen lain sebelumnya, seperti spesifikasi atau pernyataan kerja
  - Tidak ada bagian dari kebutuhan yang bertentangan dengan kebutuhan lainnya yang sudah dituliskan.

Consistent (Konsisten) - 2

- Konflik dapat berupa salah satu dari berikut ini:
  - Perilaku yang bertentangan
  - Istilah yang bertentangan
  - Karakteristik yang bertentangan
  - Inkonsistensi temporal



Verifiable (Dapat Diverifikasi)

- Suatu SKPL disebut dapat diverfikasi, jika dan hanya jika setiap kebutuhan sudah terverifikasi.
- Suatu kebutuhan dapat diverifikasi jika dan hanya jika menggunakan istilah yang dapat diukur, untuk memeriksa apakah suatu produk perangkat lunak sudah memenuhi kebutuhan.
- Harus menghindari istilah yang tidak dapat diverifikasi seperti: UI yang baik, waktu respon yang cepat, banyak user, biasanya, dll…
- Jadi secara umum, kebutuhan yang ambigu tidak dapat diverifikasi.

Modifiable (Dapat Dimodifikasi)

- SKPL dapat dimodifikasi, jika dan hanya jika strukturnya memungkinkan setiap perubahan terhadap kebutuhan dapat dibuat dibuat secara mudah, lengkap dan konsisten.
- Memiliki indeks, daftar isi, referensi silang, dan penomoran halaman yang sesuai (berkaitan dengan format dan gaya SKPL).
- Tidak ada duplikasi yang tidak perlu. Jadi suatu kebutuhan tidak perlu muncul lebih dari satu tempat di SKPL.

Traced (Dilacak)

- SKPL dilacak jika sumber kebutuhannya jelas. Itu berarti SKPL menyertakan referensi ke dokumen pendukung sebelumnya.

Traceable (Dapat Dilacak)

- SKPL dapat dilacak jika asal dari kebutuhan sudah jelas dan memberikan fasilitas untuk mereferensi setiap kebutuhan dalam pengembangan masa depan dan perbaikan dokumen.

Teknik untuk Traceability

- Beri nomor setiap paragraf secara hierarkis
- Beri nomor setiap paragraf secara hierarkis dan jangan pernah menyertakan lebih dari satu kebutuhan dalam paragraf mana pun
- Beri nomor setiap kebutuhan dengan nomor unik dalam tanda kurung segera setelah kebutuhan muncul di SKPL
- Gunakan kesepakatan untuk menunjukkan kebutuhan, misalnya, gunakan pernyataan *wajib*

Traced dan Traceability

- Dapat ditelusuri ke belakang (Backward Traceability) atau ke tahapan sebelumnya. Kita tahu mengapa setiap kebutuhan di SKPL ada. Hal ini tergantung pada setiap kebutuhan yang mereferensi ke sumber pada dokumen sebelumnya.
- Dapat ditelusuri ke depan (Forward Traceability) atau semua dokumen yang dihasilkan setelah SKPL. Setiap kebutuhan dalam SKPL diberi nomor yang khas (nomor referensi) sehingga semua dokumen setelah SKPL dapat merujuk ke SKPL.

Feasibility (Kelayakan)

- Beberapa kebutuhan mungkin tidak fisibel untuk diimplementasikan karena alasan teknis atau sumber daya yang terbatas. Kebutuhan tersebut harus diidentifikasi dan selanjutnya dihapus dari SKPL.

Dapat Dipahami oleh Pelanggan

- Pembaca utama SKPL dalam banyak kasus adalah pelanggan atau pengguna, yang cenderung ahli dalam penerapan/pemakaian tetapi tidak harus terlatih dalam ilmu komputer

Desain Independen

- SKPL disebut desain independen jika tidak menyiratkan arsitektur atau algoritma perangkat lunak tertentu

Annotated (Beranotasi)

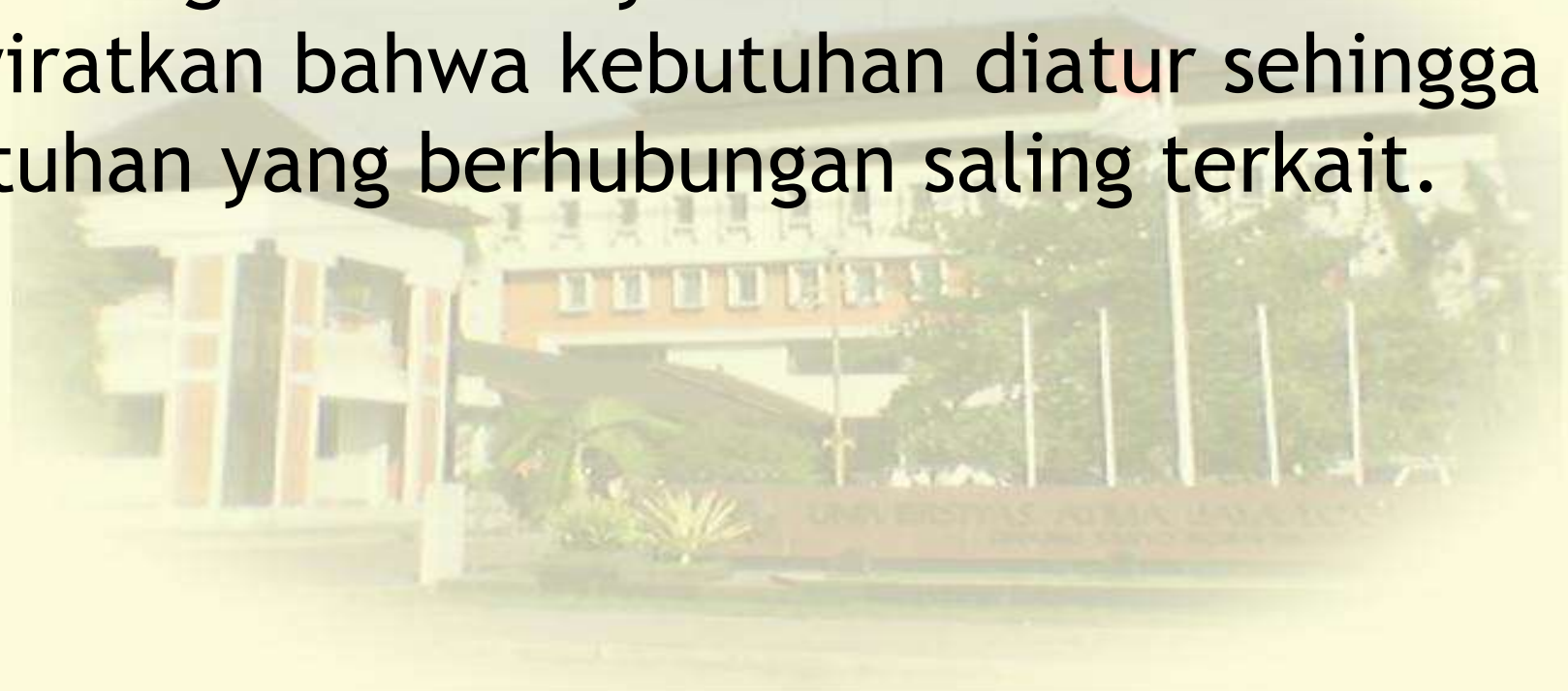
- Tujuan dari anotasi kebutuhan yang terkandung dalam SKPL adalah untuk memberikan panduan kepada organisasi pengembangan
- Kebutuhan relatif (E/D/O)
- Stabilitas relatif

Concise (Ringkas)

- SKPL yang lebih pendek lebih baik, jika sudah memenuhi semua karakteristik

Organized (Terorganisir)

- SKPL disebut terorganisir jika kebutuhan yang terkandung di dalamnya mudah ditemukan. Ini menyiratkan bahwa kebutuhan diatur sehingga kebutuhan yang berhubungan saling terkait.



Frasa yang Harus Dicari dalam SRS

- Always, Every, All, None, Never
- Certainly, Therefore, Clearly, Obviously, Evidently
- Some, Sometimes, Often, Usually, Ordinarily, Customarily, Most, Mostly
- Etc., And So Forth, And So On, Such As
- Good, Fast, Cheap, Efficient, Small, Stable
- Handled, Processed, Rejected, Skipped, Eliminated
- If…Then…(but missing Else)

Seni Keseimbangan

- Mencapai semua atribut sebelumnya dalam SKPL tidak mungkin
- Setelah Anda terlibat dalam menulis SKPL, Anda akan mendapatkan wawasan dan pengalaman yang diperlukan untuk melakukan tindakan penyeimbangan
- Tidak ada SKPL yang sempurna

Referensi

- Software Quality: Analysis and Guidelines for Success by Capers Jones
- Requirements Analysis and Specification by Alan M. Davis
- A Practitioner's Approach to Software Engineering by Roger Pressman
- Software Engineering 6th Edition, by I. Sommerville, 2000
- 'Software Engineering Quality Practices' by R. K. Kandt, Auerbach Publications, 2006
- 'Requirements Engineering: Processes and Techniques' by G. Kotonya and I. Sommerville, John Wiley & Sons, 1998
- 'Software Requirements: Objects, Functions, and States' by A. Davis, PH, 1993
- 'Software Engineering Quality Practices' by R. K. Kandt, Auerbach Publications, 2006
- Requirements Analysis and Specification by Alan M. Davis
- Yoges Singh; Software Testing, Cambridge Univerity Press; 2011
- Naresh Chauhan; Software Testing – Principles and Practices, Oxford University Press; 2010