

Tim Pengajar IF2250

IF2250 – Rekayasa Perangkat Lunak
**Penulisan Spesifikasi Kebutuhan
Perangkat Lunak**

SEMESTER II TAHUN AJARAN 2022/2023



KNOWLEDGE & SOFTWARE ENGINEERING

Kontraktor/
(System Analyst)



Membuat
Kebutuhan sistem

Kebutuhan User dan
Kebutuhan Sistem

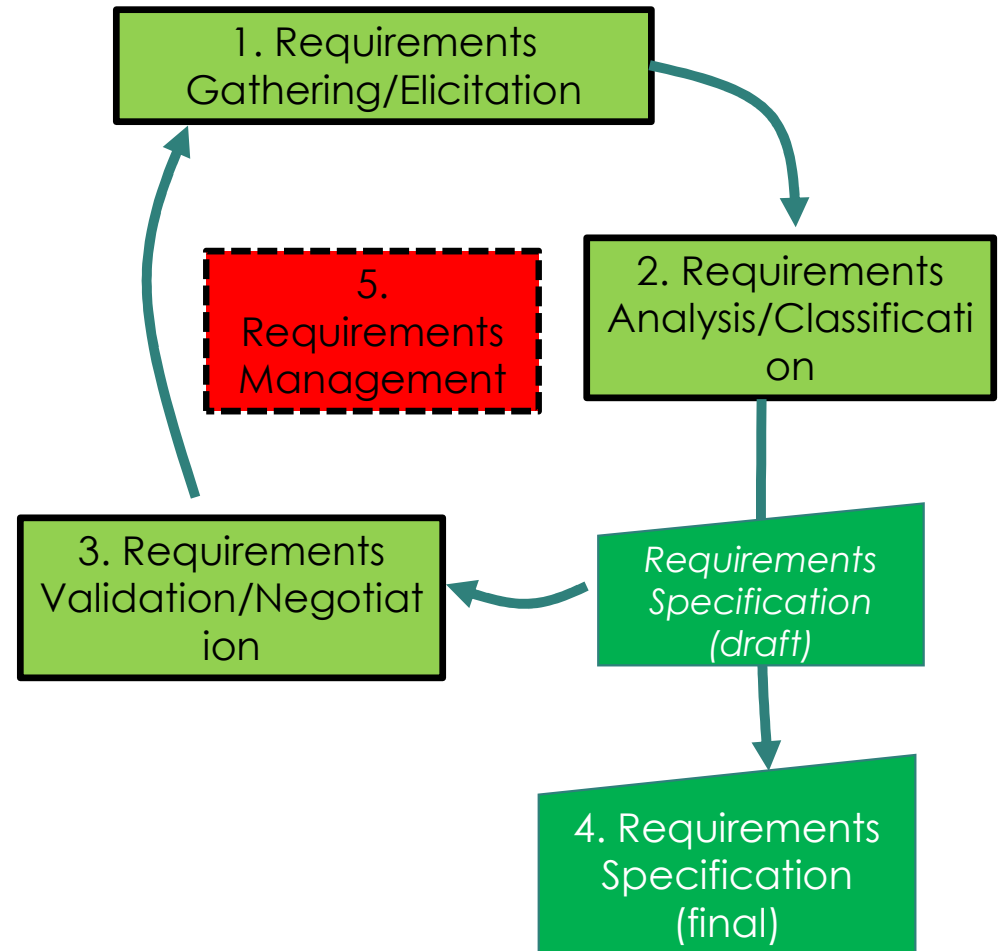
User memberikan
masukan lebih
rinci



User



KNOWLEDGE & SOFTWARE ENGINEERING



IF2250 Penulisan SKPL

Berbagai Karakteristik Pengguna (user)

- Pengguna yang memiliki suatu hak akses tertentu
 - Tamu, pengguna anggota organisasi, petugas pelayanan masyarakat, administrator jaringan, administrator sistem
- Pengguna yang hanya mengerti satu bagian operasional, sehingga kadang tidak mengerti secara lebih umum tentang sistem yang akan dikembangkan
 - Contoh: petugas loket bagian peminjaman buku, hanya mengerti bagian peminjaman buku saja, tetapi tidak tahu bahwa ada masalah pembuatan laporan pada software yang akan dibuat
- Pengguna yang hanya mengerti satu sistem
 - Hanya mengerti windows, hanya tahu menggunakan MS Word, hanya tahu 'facebook', hanya tahu menggunakan aplikasi statistik
- Pengguna yang terbiasa berkomunikasi dengan bahasa ibunya
- Pengguna yang secara langsung atau tidak langsung berhubungan dengan pelanggan
 - Contoh: pengguna yang berperan sebagai manajer akan berbeda perilakunya dengan pengguna yang berperan sebagai penghubung dengan pelanggan
- Pengguna yang sibuk atau yang tidak sibuk (sulit ditemui)
- Pengguna juga mungkin memiliki karakter yang berbeda-beda
 - Pengguna yang sulit atau lambat dalam mengungkapkan ide,
 - Pengguna yang 'sok tahu'
 - Pengguna yang suka emosional
 - Dan lain-lain

Sistem analisis harus 'sabar', mencoba mengerti, tidak mudah emosional, dll, agar tujuan pekerjaannya dapat dilaksanakan, yaitu **mendapatkan 'kebutuhan pengguna'**



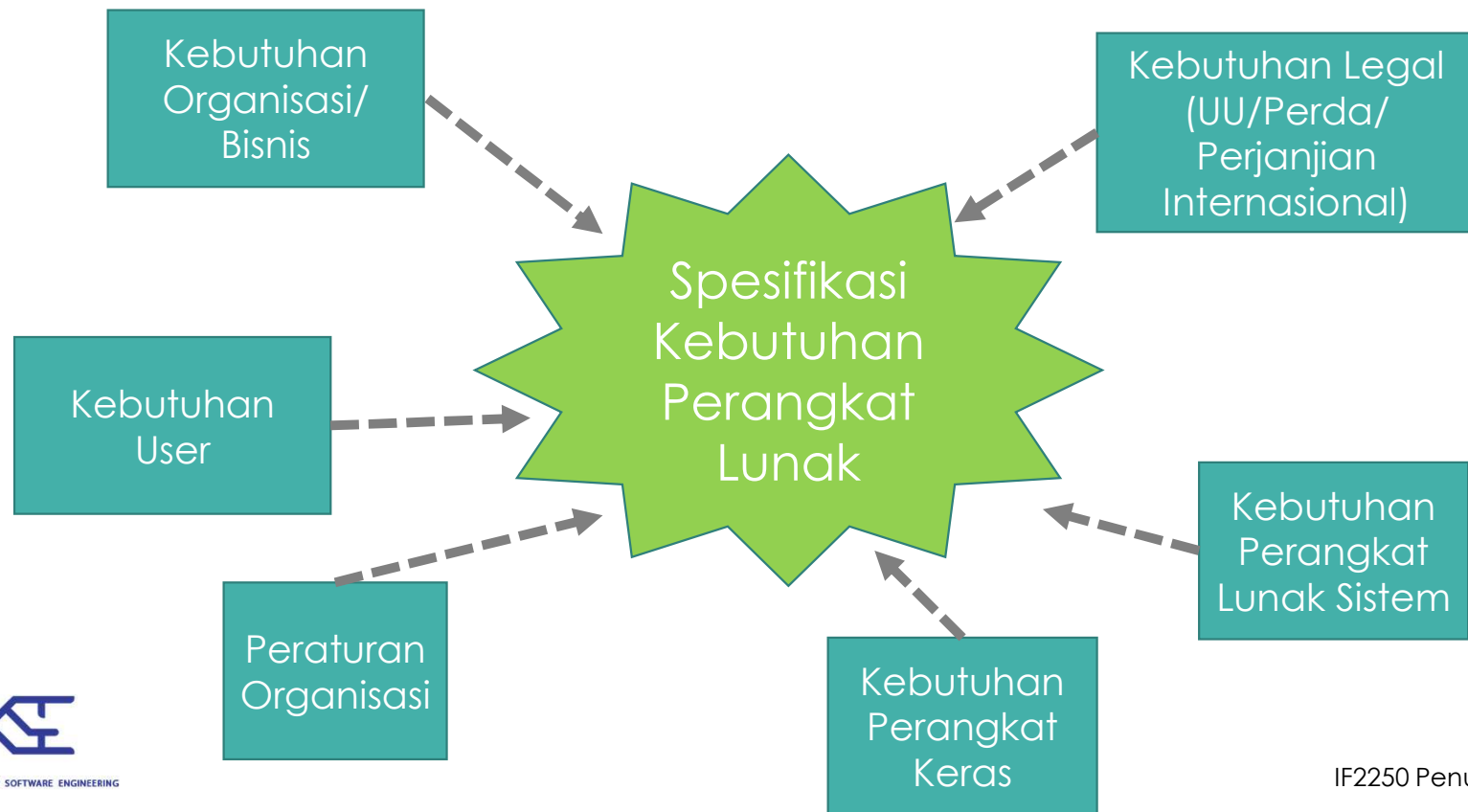
KNOWLEDGE & SOFTWARE ENGINEERING

Dengan Karakteristik tersebut, akibatnya pernyataan kebutuhan* menjadi:

- Terlalu umum, tidak rinci, tidak sederhana, terlalu 'mengawang-awang' atau bertele-tele
- Terlalu rinci, sehingga tidak menggambarkan sistem secara keseluruhan
- Terjadi konflik antara bagian yang berbeda dalam satu sistem
 - Perbedaan aturan, perbedaan tanggungjawab
- Tidak konsisten
 - *Hari ini bilang X, minggu depannya ingin Y*
- Mungkin kebutuhannya tidak realistis
 - Diluar anggaran, akan terlalu lama waktu pengembangannya



Spesifikasi Kebutuhan PL dibuat berdasarkan berbagai Kebutuhan Sistem



Penulisan Spesifikasi Kebutuhan Perangkat Lunak (SKPL)



KNOWLEDGE & SOFTWARE ENGINEERING

IF2250 Penulisan SKPL

Spesifikasi Kebutuhan

- Membuat spesifikasi kebutuhan artinya
 - Proses penulisan kembali kebutuhan pengguna (user) dan sistem menjadi dokumen spesifikasi kebutuhan perangkat lunak
- Spesifikasi Kebutuhan akan menjadi **KONTRAK** bagi Pengguna juga Pengembang
 - Pengguna dan pelanggan (customer) harus dapat mengerti
 - Pengguna umumnya tidak memiliki latar belakang teknis
 - Pengembang system harus dapat mengerti untuk membuat Perancangan dan Implementasi Program



Penulisan Spesifikasi Kebutuhan PL (SKPL)

Spesifikasi kebutuhan dapat diungkapkan dengan cara berikut:

- Bahasa sehari-hari
- Penulisan yang lebih terstruktur
- Bahasa yang mirip dengan deskripsi perancangan
 - Algoritma
- Notasi Grafis
- Notasi matematis (metode formal)

**Mudah
dituliskan**



**Sulit
dituliskan**



Contoh:

Bahasa Sehari-hari

Seorang diijinkan untuk meminjam dokumen

Bahasa Terstruktur

Siapa: Seorang

Aksi: diijinkan untuk meminjam

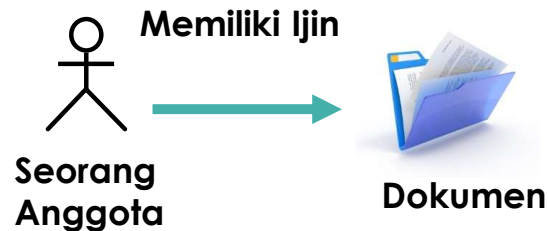
Objek: dokumen

Bahasa Deskripsi (mirip algoritma)

Jika (seseorang punya hak) maka
akan diijinkan untuk meminjam
dokumen

Metode Formal (Notasi Z)

Notasi Grafis



CheckOut

Δ Documents
 $p?: \text{PERSON}$
 $d?: \text{DOCUMENT}$

$d? \notin \text{dom checked_out}$
 $(d?, p?) \in \text{permission}$
 $\text{checked_out}' = \text{checked_out} \cup \{(d?, p?)\}$

Penulisan SKPL yang baik

1. Jelas/Lengkap (Clear, Complete, Concise)
2. Konsisten (Consistent)
3. Tepat (Correct)
4. Mudah di ubah (Modifiable)
5. Terurut (Ranked)
6. Dapat diuji (Testable)
7. Dapat ditelusuri (Traceable)
8. Tidak Ambigu (Unambiguous)



1. Jelas/Lengkap (Clear, Complete, Concise)

- Sudah mencakup semua aspek yang terkait pada situasi dunia nyatanya
- Hanya 'satu' kebutuhan (tidak tercampur aduk), dan tidak terduplikasi
- Mudah dibaca dan tidak meragukan
- Tidak menggunakan istilah yang tidak jelas atau tidak ada penjelasannya atau arti yang sama menggunakan istilah yang berbeda-beda
- Tidak menuliskan situasi yang tidak akan ditemui atau fitur yang tidak perlu



2. Konsisten (Consistent)

- Tidak ada kebutuhan yang konflik antara setiap kebutuhan
 - Misalnya konflik bisa terjadi karena ada suatu perilaku yang diharapkan terjadi tetapi mungkin menyebabkan perilaku lain menjadi tidak terjadi
 - Contoh:
 - Pegawai di bagian peminjaman: Peminjaman buku dapat dipinjam untuk 10 buku/anggota
 - Pegawai di bagian pengembalian: Sistem membolehkan pengembalian maksimum 5 buku



3. *Benar (Correct)*

- Spesifikasi kebutuhan yang **benar**, harus **akurat** dan **presisi** dalam mengidentifikasi setiap situasi dan **keterbatasannya**, termasuk suatu kemampuan yang mungkin dihadapi, dan termasuk mendefinisikan respons yang terkait dengan kemampuan tadi
 - Spesifikasi harus mendefinisikan lingkungan operasional pada dunia nyata, bagaimana interface dan interaksinya dengan lingkungan
- Kebutuhan mungkin **benar** di **satu masa**, tetapi mungkin menjadi **tidak benar** di waktu yang **berbeda**, atau dengan kata lain kebutuhan pada dunia nyata sering di selaraskan menjadi **spesifikasi** yang **benar**.
 - User membutuhkan perangkat lunak yang dapat diakses setiap saat
 - Pada jaman **sebelum** era internet, kebutuhan ini mungkin perlu diperjelas apakah akses setiap saat ini hanya **diakses** di semua **cabang perusahaan**?
 - Di **era internet**, maka kebutuhan ini dapat diterjemahkan sebagai pengembangan aplikasi **berbasis web** dengan menggunakan **client browser** untuk aksesnya
 - Di **era mobile**, maka kebutuhan ini dapat diterjemahkan sebagai pengembangan aplikasi berbasis mobile dengan karakteristik 'mobile' yang sifatnya **mudah dipindah-pindahkan** dan **tetap terkoneksi** internet



4. Mudah diubah (Modifiable)

- Kebutuhan dari pengguna relatif mudah berubah, sehingga struktur dari spesifikasi tadi harus dibuat **pengelompokan** atau **strukturisasi** yang rapi
 - Perlu pengelompokan struktur yang **logis**
 - **Tetapi**, pengelompokan ini dapat menyebabkan terjadinya **konflik** antar **spesifikasi**, atau urutan skala prioritas dalam kelompok yang berbeda



5. *Terurut (Ranked)*

- Spesifikasi dapat **diurutkan** sesuai dengan **prioritas**, atau **kepentingan** atau Stabilitas dari suatu kebutuhan
 - Urutan ini dapat dibentuk dalam **struktur dokumen**
- Makin **tinggi kompleksitas** masalah, maka makin **sulit pengurutan** spesifikasi ini,
 - Tetapi hal ini juga menunjukkan akan pentingnya pengurutan spesifikasi ini dilakukan dengan benar
 - **Pengurutan** spesifikasi kebutuhan ini dapat mempengaruhi **pengembangan**.
 - Kesalahan **urutan pekerjaan** pengembangan spesifikasi kebutuhan dapat mengakibatkan pengembangan yang **tidak efisien**



5. *Terurut (Ranked) (2)*

- Contoh:
 - Pembangunan sistem perpustakaan melibatkan beberapa SKPL, contoh:
 - “Perangkat Lunak memiliki fungsi peminjaman buku oleh anggota” (SKPL01)
 - “Perangkat Lunak memiliki fungsi pengembalian buku oleh anggota” (SKPL02)
 - “Perangkat lunak memiliki fungsi layanan pendaftaran anggota baru” (SKPL03)
 - “Perangkat lunak memiliki fungsi layanan pendataan buku baru” (SKPL04)
 - Keempat SKPL tersebut, dapat diurutkan berdasarkan:
 - Fungsi utamanya: SKPL01, SKPL02, SKPL03, SKPL04
 - Susunan urutan langkah: SKPL03, SKPL04, SKPL01, SKPL02
 - Buku dan anggota baru dapat melakukan transaksi peminjaman setelah data buku/anggota sudah tersedia



6. Dapat diujikan (Testable)

- Spesifikasi kebutuhan harus dapat **diujikan**, atau artinya dapat **dibuktikan** atau dinilai secara **kuantitatif**
 - Kebutuhan bahwa “Sistem harus mudah digunakan” sifatnya sangat subyektif, jadi kebutuhan semacam ini **tidak dapat diujikan**
 - Pada **SKPL** kita harus **mendefinisikan kembali** apa itu ‘sistem yang mudah digunakan’ bagi pengguna, pada beberapa kasus dapat dituliskan dengan.
 - Contoh Spesifikasi Kebutuhan yang **baik**:
 - “Akan dilakukan pelatihan 5 hari sebelum operasional” (dengan harapan pengguna akan mudah menggunakan sistem baru setelah dilakukan pelatihan)
 - “Sistem menggunakan sistem operasi Windows” (dengan harapan pengguna yang sudah terbiasa menggunakan Windows akan tidak sulit menjalankan aplikasi yang dikembangkan dengan Windows)
 - “Sistem memiliki fasilitas Bantuan/Panduan” (dengan harapan pengguna yang kesulitan akan dapat menggunakan fasilitas ini.



7. Dapat ditelusuri (Traceable)

- Setiap Spesifikasi kebutuhan yang ditulis pada dokumen harus diberikan **nomor identifikasi** yang **unik**
 - Agar mudah **ditelusuri**, juga mudah diucapkan
- Penulisan nomor identifikasi ini harus **konsisten** dan dengan struktur yang **logis**
 - Contoh:
 - SRS001: "PL memiliki fungsi pencatatan anggota baru"
 - SRS002: "PL memiliki fungsi peminjaman buku"
 - SRS003: "PL memiliki fungsi pengembalian buku"



8. Tidak Ambigu (Unambiguous)

- Spesifikasi kebutuhan ini harus **tidak ambigu**
 - Tidak ada interpretasi **ganda**
- Ketidakambiguan ini relatif paling **sulit** jika menggunakan **Bahasa sehari-hari** (Bahasa natural)
- Penggunaan **ungkapan** yang **tidak tegas** atau **struktur** yang **jelek** dapat menyebabkan ambiguitas atau salah mengerti/interpretasi bagi yang membaca.
 - Hati-hati menggunakan kata hubung “dan”, “atau” , “seperti”
 - Sistem dapat melakukan X dan Y
 - Apakah artinya: X dan Y harus dilakukan **bersamaan**?
 - Atau artinya ada **dua kebutuhan**:
 - Sistem dapat melakukan X
 - Sistem dapat melakukan Y



Lain-lain (1)

- Hindari kata-kata seperti “kecuali”, “tetapi” atau “jika diperlukan”
 - Contoh:
 - *Perangkat lunak memiliki fungsi pembayaran denda jika diperlukan*
- Satu spesifikasi harus ditulis **lengkap** tapi **atomik** (tidak mengandung spesifikasi lain)
 - Hati-hati menuliskan spesifikasi yang intinya tersebar di beberapa spesifikasi
- Hindari suatu spesifikasi dengan kata-kata semacam buzzword
 - Contoh: system mampu melayani **Big Data** (apa ini?)



Lain-lain (2)

- Setiap kebutuhan harus ditulis sebagai:
 - **Subjek – Predikat (- objek)**
 - Subjek: Nama identifikasi dari sistem
 - Predikat: melakukan suatu aksi, hasil yang diinginkan
- Hindari penulisan **singkatan**, apalagi yang dapat menimbulkan ambiguitas
 - Contoh:
 - “PL melakukan penyimpanan data, pencetakan, dan lain-lain”
 - “PL melakukan proses seperti di word”
- Hindari penggunaan kata jumlah yang **tidak terukur** atau tidak jelas
 - Contoh: “kira-kira”, “mendekati jumlah “, “efek minimal”, “mungkin”, “sebaiknya”
 - “PL memiliki fungsi penghitungan yang kira-kira tidak lebih dari 100ribu data”



Lain-lain (3)

- Hindari kata-kata yang **mengambang/tidak perlu**
- Hindari istilah yang **berbeda** untuk **mengacu** pada 'benda' atau sesuatu yang **sama**
- Hindari **spekulasi harapan sistem** yang tidak perlu
 - Jangan menuliskan yang mungkin tidak akan tercapai
- Hindari penulisan **kebutuhan** yang **belum jelas**
- Gunakan kalimat '**positif**' dan bukan 'negative'
 - Contoh: "PL dapat melakukan <sesuatu>"
 - Contoh salah: "PL tidak dapat melakukan <sesuatu>"



Lain-lain (4)

- Rules of thumbs, dalam Bahasa Indonesia gunakan kalimat
 - <**Perangkat Lunak/Sistem>Nama**> <kata kerja> <obyek>
 - Perhatikan **konsistensi** penggunaan “**nama**” pada **Subyek**
 - Contoh:
 - PL memiliki layanan pendaftaran anggota baru
 - PL memiliki layanan pembelian barang
 - PL memberikan fungsi pembayaran barang yang sudah dipesan
 - PL melakukan permintaan Informasi login/password untuk setiap transaksi pembelian dan penjualan
 - Contoh lain:
 - Sistem Ecommerce memiliki layanan pendaftaran anggota baru
 - Sistem Ecommerce memiliki layanan pembelian barang



Lain-lain (5)

- *Requirement dalam Bahasa Inggris:*
 - **Shall** digunakan di mana **persyaratan** (*requirement*) sedang **dinyatakan**
 - **Will** harus digunakan untuk mewakili **pernyataan fakta**
 - **Should** mewakili **tujuan** yang ingin **dicapai**



Kebutuhan Pengguna vs. SKPL

Kebutuhan Pengguna	SKPL
Software harus 'user friendly' (bagaimana mengukurnya?)	PL memiliki rancangan yang menggunakan menu, dialog box,, dropdown menu.
Semua tampilan di layar harus terlihat di monitor dengan cepat (seberapa cepat?)	PL dapat menampilkan hasil dalam kurang dari 2 detik saat pengguna mengakses suatu fungsi
Jika software hang, maka sistem akan dapat di restart secepatnya	PL dapat di-restart dalam waktu kurang dari 30 menit jika terjadi hang
Upgrade terhadap sistem dapat dilakukan tanpa sama sekali mengganggu sistem produksi yang berjalan 24 jam (harapan yang tidak realistis)	PL tidak menyebabkan sistem produksi berhenti lebih dari 2 hari jika terjadi upgrade terhadap sistem.
Kalau user gagal login tiga kali, maka program javascript harus di jalankan untuk mengunci user dari sistem (javascript program adalah aspek implementasi)	PL akan mengunci user yang gagal login 3 kali.



Kesalahan Umum

- Jangan membuat **asumsi** yang 'Jelek'
- Jangan menuliskan 'HOW', tapi seharusnya 'WHAT'
 - **Jangan** menuliskan **penjelasan operasinya**
 - **Jangan** terlalu **rinci** menuliskan kebutuhannya, cenderung **bias** menjadi penjelasan HOW-nya.
- Contoh:
 - Kalau user gagal login tiga kali, maka program javascript harus di jalankan untuk mengunci user dari system
 - javascript program adalah aspek implementasi



Kesalahan Umum

- Jangan menggunakan **istilah** yang **salah**
- Jangan menggunakan **struktur** kalimat yang **tidak tepat**
 - Grammar yang jelek
- Jangan ada **kebutuhan** yang '**hilang**'
 - Mungkin **terlupakan** saat wawancara, atau **tidak tercatat** dalam notulen hasil dari suatu rapat/ pertemuan



Jangan membuat Asumsi yang Jelek'

- Biasanya terjadi karena analis **tidak punya informasi** yang cukup
 - Informasi dapat diperjelas dengan menanyakan lebih lanjut
 - Walaupun sering pengguna/customer mempercayai pendapat kita, tetapi mereka biasanya lebih tahu apakah asumsi yang kita gunakan sudah cukup atau berlebihan atau masih kurang
- Asumsi ini sering terkait dengan:
 - Anggaran/Biaya: Anggaran/biaya yang terlalu sedikit menyebabkan suatu solusi disederhanakan, sehingga asumsi kadang diperlukan, tentunya harus di konfirmasi dulu ke calon pengguna/customer
- Asumsi ini harus jelas untuk pengguna/customer, juga jelas untuk semua anggota tim pengembang
 - Selalu di review dengan kebutuhan yang terkait



Jangan menuliskan 'HOW', tapi seharusnya 'WHAT'

- Kenapa?
 - Untuk menghindari pemaksaan suatu solusi 'Desain/Perancangan'
 - Bila suatu kebutuhan sudah berisi solusi rancangan, maka ada kemungkinan solusi ini tidak/kurang memperhatikan kebutuhan yang lain
 - Solusi harus perancangan harus memperhatikan semua kebutuhan lain
 - Solusi 'bagaimana' pada suatu kebutuhan menyebabkan rancangan menjadi tidak fleksibel
 - Sering terjadi pengguna yang tidak pengalaman menyatakan suatu solusi rancangan padahal yang kita butuhkan adalah 'APA' yang dia butuhkan.
 - Ada kemungkinan solusi rancangannya hanya bersifat sementara atau hanya memandang dari sudut pandang teknologi tertentu
- Jadi yang ditulis adalah 'Requirements'/Kebutuhan bukan 'Operasi'
 - Sulit memverifikasi suatu kebutuhan yang dinyatakan dalam suatu 'operasi'



Jangan menggunakan istilah yang salah (1)

- Dalam bahasa Inggris dibedakan istilah:
 - Shall: sesuatu yang harus dapat di verifikasi, biasanya requirements menggunakan 'shall'
 - Will: menyatakan suatu fakta
 - Should: menyatakan suatu obyektif/tujuan
 - Kata 'are', 'is', 'was', 'must' tidak digunakan pada requirements
 - Kata ini dapat digunakan sebagai penjelasan suatu kebutuhan
- Dalam Bahasa Indonesia, disarankan menggunakan:
 - Kata 'mampu' atau 'dapat' atau 'akan' untuk menggantikan 'shall'
 - Kata 'memiliki' untuk menggantikan 'Will'
 - Kata 'mampu' atau 'dapat' atau 'akan' untuk menggantikan 'should'



Jangan menggunakan istilah yang salah (2)

- **Hindari** penggunaan kata:
 - Support (mendukung)
 - But not limited to (tapi tidak dibatasi pada...)
 - Etc (dan lain-lain)
 - And/or (dan/atau)
- Mendukung..
 - Salah:
 - Sistem akan mendukung pengguna dalam memperbaiki data lama
 - Benar
 - Sistem akan menyediakan layar untuk memperbaiki data pengguna yang lama
- Istilah lain tidak jelas berapa batasnya atau tidak jelas apa jumlahnya, atau juta tidak jelas apakah 'dan' atau 'atau'



Struktur Kalimat (1)

- Pernyataan kebutuhan harus **mudah dibaca** dan **dimengerti**.
- Dimulai dengan APA yang dapat/mampu dilakukan oleh sistem/perangkat lunak
 - Bukan 'HOW'
- Bentuk umum (panduan)
 - <sistem> <akan | dapat | mampu> memberikan/melakukan <suatu aksi>
 - <sistem> dapat <kata kerja>
- Nama <sistem> dapat diganti dengan <perangkat lunak> atau suatu nama sistem atau nama perangkat lunak
 - Contoh:
 - PL dapat menyimpan data perubahan
 - Sistem PL mampu menyimpan data perubahan
 - Aplikasi mampu menyimpan data perubahan
 - Sistem Perpustakaan mampu menyimpan data perubahan



Struktur Kalimat (2)

- Penulisan dengan penggunaan daftar (*list*) sebaiknya dihindari, Karena setiap item kebutuhan harus diverifikasi, kecuali memang dalam daftar tersebut mungkin dapat diverifikasi dengan satu cara saja
- Contoh
 - Sistem dapat melakukan:
 - Mencatat peminjaman buku
 - Mencatat anggota perpustakaan baru
 - Sebaiknya ditulis sebagai:
 - Sistem dapat mencatat peminjaman buku
 - Sistem dapat mencatat anggota perpustakaan baru



Struktur Kalimat: Subjek

- Hindari penggunaan subjek yang salah
 - Contoh salah
 - Database akan melakukan penyimpanan data perubahan
 - Perhatikan bahwa walaupun memang Database yang faktanya akan melakukan proses penyimpanan, tetapi gunakan Nama Sistem/ perangkat lunak.
- Contoh:
 - Sistem akan melakukan penyimpanan data perubahan



Kalimat yang terlalu panjang

- Kebutuhan harus dituliskan dengan kalimat yang sifatnya '**atomik**' atau tidak terlalu panjang atau terlalu kompleks
 - Mungkin saja sebenarnya kebutuhan yang **panjang** tadi seharusnya ditulis sebagai **lebih dari satu** kebutuhan
- Contoh salah:
 - Sistem perpustakaan dapat memberikan fasilitas peminjaman dan pengembalian buku khususnya untuk anggota yang sudah mendaftar
- Contoh benar:
 - Sistem perpustakaan dapat memberikan layanan peminjaman buku untuk anggota terdaftar
 - Sistem perpustakaan dapat memberikan layanan pengembalian buku untuk anggota terdaftar.



Hindari Kata yang Ambigu

- Contoh:
 - Minimisasi, maksimisasi, cepat, user-friendly (mudah digunakan), cukup mudah, secukupnya
- “Seminimum/semaksimal mungkin” harus jelas seberapa minimum atau maksimal



Kebutuhan atau objektif

- Kadang kita sulit mendefinisikan apa yang dibutuhkan
 - Maka yang harus kita tulis adalah obyektifnya atau tujuannya
 - Jadi pernyataannya akan ditulis sebagai obyektif/tujuan dan bukan kebutuhan
- Struktur kalimat:
 - <sistem> <diharapkan | sebaiknya> <dapat | mampu> <melakukan suatu aksi>
 - Dalam Bahasa Inggris digunakan istilah 'Should' dan bukan 'shall'



Kalimat Kebutuhan bersifat Imperative (keharusan)

- Menyatakan sesuatu yang 'harus' dibuat atau 'harus' ada pada perangkat lunak, walaupun demikian kata 'harus' perlu dihindari, kata 'harus' sebaiknya digunakan untuk menyatakan suatu performansi/kualitas
- Contoh:
 - PL mampu melakukan kalkulasi gaji pegawai, kalkulasi harus menghitung gaji seluruh karyawan



Panduan Umum penulisan SKPL

- Mengikuti **aturan tata tulis** (grammar)
- Dokumen harus bebas dari **salah ketik** (kurang huruf), salah penulisan ataupun penggunaan tanda baca yang salah
- Kebutuhan ini perlu dituliskan mengikuti **template** penulisan yang sama dalam satu organisasi
- Kebutuhan perlu dituliskan pada **bagian** yang sudah **disiapkan** (dari template).



Struktur Kalimat (yang disarankan)

- [Lokalisasi] [Aktor/Owner] [aksi] [target | owned] [constraint]
- Contoh:
 - Jika sistem terkena virus, sistem dapat beroperasi kembali kurang dari 2 hari.
- [Lokalisasi] → Jika sistem kena virus
- [Aktor/Owner] → Sistem
- [Aksi] → beroperasi
- [Target/Owned] →



Pemberian Contoh

- Suatu contoh dapat digunakan untuk melengkapi pernyataan kebutuhan
- Contoh yang sama dapat digunakan jika diperlukan agar memperjelas masalah
- Harus jelas dituliskan '**Contoh**' jangan tertukar dengan 'spesifikasi kebutuhan'
 - Contohnya adalah



Hindari ... (1)

- Hindari penggunaan kata “**harus**”
 - “Sistem harus melakukan pencetakan laporan”
 - Penulisan cukup dengan pernyataan:
 - “Sistem akan melakukan pencetakan laporan”
 - “Sistem memiliki fungsi pencetakan laporan”
 - Penggunaan kata harus memiliki sifat ‘wajib’ dilakukan, padahal pernyataan yang sederhana sudah menunjukkan sesuatu yang perlu/akan dilakukan oleh pengembang



Hindari ... (2)

- Hindari kata '**sebaiknya**'
 - “sistem sebaiknya bisa mengkalkulasi data total penerimaan harian”
 - Seharusnya ditulis:
 - “Sistem akan melakukan kalkulasi data total penerimaan harian”
 - “Sistem memiliki fungsi kalkulasi data total penerimaan harian”
 - Penambahan kata “sebaiknya” membuat tidak jelas, apakah **bisa dilakukan** ataukah mungkin **tidak perlu dilakukan**.



Hindari ... (3)

- Hindari penggunaan kata-kata yang mungkin berarti ganda atau tidak tegas.
 - Contoh: “**mendukung**”
 - “Sistem mendukung pencetakan laporan bulanan”
 - Kata mendukung, tidak secara langsung atau tidak tegas menyatakan apa yang akan ditunjukkan pada SKPL
 - Contoh yang benar: “Sistem akan mencetak laporan bulanan”



Hindari ... (4)

- Hindari kalimat yang memiliki arti tidak pasti. Misalnya kata **“mungkin”**
 - “Sistem dapat melakukan X, tapi tidak dibatasi hanya X, mungkin sistem dapat juga melakukan Y”
 - Sebaiknya: “Sistem akan melakukan X dan Y”
 - Kata mungkin juga perlu dihindari, karena tidak jelas status pernyataannya.
- Kata sambung “atau” juga bisa membingungkan, karena “atau” bisa berarti salah satu atau dua-duanya.
 - User: “sistem perlu melakukan operasi A atau operasi B”
 - Kalau memang dua-dua perlu dilakukan, maka harus dituliskan: “Sistem akan melakukan operasi A dan B”



Hindari ... (5)

- Contoh yang tidak pasti, adalah kalimat dengan kata: “...
dan lain-lain”
 - Sistem akan mampu menyimpan laporan harian, bulanan, mingguan dan lain-lain”
 - Seharusnya: “Sistem akan menyimpan laporan harian, bulanan, mingguan dan tahunan”
- Perhatikan bahwa kita harus menuliskan dengan jelas dan lengkap apa saja yang diperlukan. Kata “dan-lain-lain” membuat interpretasi menjadi tidak jelas



Penulisan Referensi

- Semua bagian yang merujuk ke suatu informasi lain, harus jelas dituliskan acuannya
 - Acuan dapat dituliskan sebagai kode
 - Contoh:
 - “Sistem dapat memiliki fungsi khusus yang didefinisikan di [DEF100]”
- Referensi juga harus jelas
 - Contoh:
 - [DEF100] “Data Management Company”, by Ali Budi, 2017.
- Acuan terhadap suatu bagian dari dokumen dapat dituliskan sebagai contoh berikut:
 - “Sistem menggunakan fungsi yang dijelaskan di bab 3, paragraph ke 4”



Penggunaan Tabel dan Gambar

- Tiap tabel dan gambar diberikan nomor yang unik
- Daftar judul tabel dan gambar dicantumkan pada daftar isi
 - Agar mudah dicari
 - Berikan penjelasan untuk setiap tabel/gambar

Program Editor untuk dokumen memiliki fungsi 'referensi' yang dapat digunakan secara otomatis

Contoh: 'Reference' di MS Word



- Penulisan Spesifikasi Kebutuhan Perangkat Lunak saat ini:
 - Belum ada panduan standard untuk Bahasa Indonesia
 - Aturan umumnya:

<NamaPerangkatLunak atau Sistem>
<mampu | dapat | akan>
<predikat>
<obyek>



Pengembangan SKPL

- Cari **fungsi** yang akan ada pada **perangkat lunak**
 - Tentunya juga jelas kenapa suatu fungsi itu diperlukan.
- Cari **proses-proses** apa saja yang akan ada nanti di aplikasi perangkat lunak ini
 - Pada bagian apa suatu proses akan digunakan, bagaimana dan apa efeknya.
- Jika proses dalam software ini nantinya berbeda dengan kenyataannya, maka kita harus berkonsultasi dengan user.
Contoh:
 - Perubahan dalam kebijaksanaan baru organisasi kadang mengubah kebiasaan yang ada.
 - Pergantian staf dalam perusahaan



Analisis Kebutuhan

- Melakukan pengkajian ulang (**review**) terhadap semua dokumen yang relevan
- Mengembangkan konsep awal
- Konsultansi dengan pengguna
 - Mendapatkan kejelasan
 - Negosiasi jika diperlukan, misalnya karena
 - Ada kebutuhan yang mungkin di luar anggaran
 - Butuh waktu yang lebih lama karena mungkin alasan teknis/non teknis
 - Ada kebutuhan di luar batasan yang pernah di bicarakan sebelumnya.
- Konsolidasi semua data masukan ke sistem
- Konsolidasi semua hasil keluaran dari sistem



Studi Kasus

Mesin Jual Otomatis (Vending Machine)



IF2250 Penulisan SKPL



KNOWLEDGE & SOFTWARE ENGINEERING



Kebutuhan Pengguna untuk Vending Machine (VM)

Suatu VM menjual barang secara otomatis. VM akan menampilkan barang dan harganya. Seorang pembeli akan memasukkan sejumlah koin dan kemudian memilih barang yang akan di beli.

Transaksi pembelian hanya akan terjadi jika pengguna telah memasukkan koin yang benar (100, 200, 400, 500 dan 1000). Artinya jika benda lain dimasukkan selain koin yang benar, maka benda itu akan otomatis dikeluarkan.

Sistem yang dikembangkan harus berjalan di sistem Linux, dan software harus menyesuaikan dengan kebutuhan perangkat keras VM.



KNOWLEDGE & SOFTWARE ENGINEERING

Kebutuhan Sistem untuk VM (1)

- Mesin memiliki mekanisme untuk memeriksa apakah objek yang dimasukkan adalah koin yang benar dengan melakukan validasi terhadap ukuran, berat, ketebalan dan juga sisi logamnya.
- Mesin menerima koin 100, 200, 500 dan 1000 rupiah. Selain itu akan dianggap sebagai objek yang tidak valid.
- Mesin dapat melakukan proses perhitungan pembayaran dan juga pemilihan produk, setelah koin yang benar di deteksi.
- Mesin akan menerima masukan jenis barang yang akan dibeli oleh pembeli
- Mesin akan memeriksa apakah suatu barang yang dipilih tersedia, jika tidak ada maka koin akan dikembalikan secara otomatis, dan akan ada pemberitahuan ke pembeli.



Kebutuhan Sistem untuk VM (2)

- Jika pembeli tidak melakukan pemilihan barang (ingin dibatalkan), maka koin akan dikembalikan oleh mesin
- Mesin akan mengeluarkan barang jika memang masih tersedia dan jumlahnya koinnya sudah cukup
- Mesin akan memberikan uang kembali jika uang yang dimasukkan lebih dari harga barang.
- Tombol pemilihan barang di non-aktifkan jika suatu barang sudah dikeluarkan dan selama jumlah koin yang cukup sudah dimasukkan
- Mesin bisa menerima pergantian jenis barang oleh pemilik mesin. Dengan demikian harga dari produk yang baru harus dapat diperbaiki.



Spesifikasi Kebutuhan PL untuk VM (1)

- Sistem VM menjual barang secara otomatis
- Sistem VM menampilkan barang dan harganya
- Pembeli memasukkan koin
- Pembeli memilih barang
- Sistem VM berjalan di system operasi LINUX
- Sistem VM memeriksa validasi koin dengan ukuran, berat, ketebalan, sisi logam
- Sistem VM hanya menerima koin 100, 200, 500, dan 1000
- Sistem VM akan menolak masukan koin yang tidak sah
- Sistem VM otomatis akan mengeluarkan koin yang tidak sah
- Sistem VM akan mengembalikan koin jika pembeli batal melakukan pemilihan barang
- Sistem VM melakukan kalkulasi penghitungan pembayaran
- Sistem VM dapat memberikan kembalian jika pembayaran berlebih atau jika barang ternyata tidak tersedia
- Pemilik dapat mengganti jenis barang, VM akan mengupdate harga produk



Spesifikasi Kebutuhan PL untuk VM (2)

- ~~Sistem VM menjual barang secara otomatis~~
- Sistem VM menampilkan barang dan harganya
- ~~Pembeli memasukkan koin~~
- ~~Pembeli memilih barang~~
- Sistem VM berjalan di system operasi LINUX
- ~~Sistem VM memeriksa validasi koin dengan ukuran, berat, ketebalan, sisi logam~~
- ~~Sistem VM hanya menerima koin 100, 200, 500, dan 1000~~
- ~~Sistem VM akan menolak masukan koin yang tidak sah~~
- ~~Sistem VM otomatis akan mengeluarkan koin yang tidak sah~~
- Sistem VM akan mengembalikan koin jika pembeli batal melakukan pemilihan barang
- Sistem VM melakukan kalkulasi penghitungan pembayaran
- Sistem VM dapat memberikan kembalian jika pembayaran berlebih atau jika barang ternyata tidak tersedia
- Pemilik dapat mengganti jenis barang, VM akan mengupdate harga produk



Spesifikasi Kebutuhan PL untuk VM (3)

- ~~Sistem VM menjual barang secara otomatis~~
- Sistem VM menampilkan barang dan harganya **F**
- ~~Pembeli memasukkan koin~~
- ~~Pembeli memilih barang~~
- Sistem VM berjalan di system operasi LINUX **NF**
- ~~Sistem VM memeriksa validasi koin dengan ukuran, berat, ketebalan, sisi logam~~
- ~~Sistem VM hanya menerima koin 100, 200, 500, dan 1000~~
- ~~Sistem VM akan menolak masukan koin yang tidak sah~~
- ~~Sistem VM otomatis akan mengeluarkan koin yang tidak sah~~
- Sistem VM akan mengembalikan koin jika pembeli batal melakukan pemilihan barang **F**
- Sistem VM melakukan kalkulasi penghitungan pembayaran **F**
- Sistem VM dapat memberikan kembalian jika pembayaran berlebih atau jika barang ternyata tidak tersedia **F**
- Pemilik dapat mengganti jenis barang, VM akan mengupdate harga produk **F**



Spesifikasi Kebutuhan PL

- Kebutuhan Fungsional
 - Sistem VM dapat menerima masukan koin
 - Sistem VM dapat menerima masukan jenis barang
 - Sistem VM menampilkan barang dan harganya
 - Sistem VM melakukan kalkulasi penghitungan pembayaran
 - Sistem VM dapat memberikan kembalian jika pembayaran berlebih atau jika barang ternyata tidak tersedia
 - Sistem VM dapat menerima update barang dan harga dari pemilik mesin
- Kebutuhan Non Fungsional
 - Sistem VM berjalan di system operasi LINUX



Spesifikasi Kebutuhan PL (Berikan Kode)

- Kebutuhan Fungsional
 - VM01: Sistem VM dapat menerima masukan koin
 - VM02: Sistem VM dapat menerima masukan jenis barang
 - VM03: Sistem VM menampilkan barang dan harganya
 - VM04: Sistem VM melakukan kalkulasi penghitungan pembayaran
 - VM05: Sistem VM dapat memberikan kembalian jika pembayaran berlebih atau jika barang ternyata tidak tersedia
 - VM06: Sistem VM dapat menerima update barang dan harga dari pemilik mesin
- Kebutuhan Non Fungsional
 - VM07: Sistem VM berjalan di system operasi LINUX



*Cara pengkodean tidak ada standard,
yang penting kode digunakan konsisten
dan mudah digunakan*

Dokumentasi Spesifikasi Kebutuhan Perangkat Lunak (SKPL)



KNOWLEDGE & SOFTWARE ENGINEERING

IF2250 Penulisan SKPL

Pendahuluan

- Hasil analisa kebutuhan itu nanti harus didokumentasikan dalam dokumen yang sudah standard
 - Dengan standard, maka komunikasi akan lebih lancar
- Dokumen kebutuhan ini sering disebut sebagai **SRS** (Software Requirement Specification) atau **SKPL** (Spesifikasi Kebutuhan Perangkat Lunak)



Dokumen Kebutuhan (Requirements Document)

- Dokumen kebutuhan ini berisi pernyataan resmi tentang apa yang harus dikerjakan oleh pengembang
- Dokumen ini sebaiknya mengikutsertakan definisi dari kebutuhan pengguna (user requirements) dan juga spesifikasi dari kebutuhan sistem (system requirements).
- Dokumen ini adalah BUKAN dokumen perancangan
 - Dokumen ini harus berisi 'APA' yang dilakukan oleh sistem dan seminimal mungkin tentang 'BAGAIMANA' mengerjakannya.



Siapa pengguna dalam dokumen kebutuhan?

Peran	Apa guna dokumen?
Pengguna Sistem	Pengguna sistem memberikan spesifikasi kebutuhan dan membaca dokumen untuk memeriksa apakah sudah sesuai dengan keinginannya. Keinginan mereka bisa merubah isi dokumen kebutuhan ini
Manajer	Dokumen ini kadang digunakan untuk bahan mengajukan tender. Pada situasi lain, manajer menggunakan dokumen ini untuk melakukan perencanaan pengembangan.
System Engineers	Dokumen digunakan untuk mengerti apa yang harus dikembangkan.
System Test Engineers	Dokumen digunakan untuk membuat validasi dari setiap kebutuhan
System Maintenance Engineers	Dokumen digunakan untuk membantu mengerti sistem dan keterhubungan antar bagian dalam sistem.

** Software Engineering 7th ed, Ian Sommerville*



KNOWLEDGE & SOFTWARE ENGINEERING

IF2250 Penulisan SKPL

Struktur Umum Dokumen Kebutuhan

- Preface
- Introduction
- Glossary
- User requirements definition
- System architecture
- System requirements specification
- System models
- System evolution
- Appendices
- Index



Struktur Dokumen Kebutuhan (1)

Chapter	Deskripsi
Preface	Berisi apa yang akan dibaca dari dokumen. Termasuk juga berisi penjelasan sejarah versi dari dokumen, termasuk kenapa versi terakhir dikembangkan.
Introduction	Berisi penjelasan mengapa sistem ini dibutuhkan. Bagian ini secara umum menjelaskan fungsi utama dari sistem dan menjelaskan keterhubungannya dengan sistem lain. Bagian ini juga menjelaskan keberadaan sistem ini dalam konteks bisnis perusahaan secara umum ataupun tujuan strategis mengapa perusahaan memerlukan sistem ini.
Glossary	Bagaimana ini menjelaskan istilah teknis dalam dokumen. Bagian ini mungkin dibaca oleh orang awam, jadi jangan diasumsikan pembaca adalah orang yang memang sudah ahli atau berpengalaman.
User requirements definition	Layanan sistem bagi user dijelaskan di bagian ini. Kebutuhan NF juga di nyatakan di bagian ini. Deskripsi ini dapat menggunakan bahasa natural, diagram atau notasi lain yang dapat dimengerti pengguna. Standard produk dan proses dapat dispesifikasikan juga.
System architecture	Bagian ini berisi pandangan arsitektur sistem secara umum. Caranya dengan menunjukkan fungsi-fungsi yang tersebar ke modul sistem lain. Bagian komponen arsitektur yang di gunakan lagi dapat di tekankan lagi



Struktur Dokumen Kebutuhan (2)

Chapter	Description
System requirements specification	Bagian ini menjelaskan kebutuhan fungsional dan NF secara lebih rinci. Bahkan interface ke sistem lain juga dapat dijelaskan dibagian ini.
System models	Bagian ini mungkin menggunakan model grafik untuk menunjukkan keterhubungan antar komponen sistem dan juga sistem dengan lingkungannya. Contohnya: Model Objek, Model data-flow atau model Data Semantik.
System evolution	Bagian ini menjelaskan asumsi dasar dari sistem yang akan dikembangkan, dan antisipasi perubahan karena evolusi perangkat keras, dan perubahan kebutuhan pengguna. Bagian ini berguna bagi perancangan sistem agar keputusan perancangannya tidak dibatasi oleh perubahan sistem di masa depan.
Appendices	Bagian ini berisi informasi rinci/spesifik yang terkait dengan aplikasi yang sedang dikembangkan. Misalnya deskripsi hardware dan basisdata.
Index	Index dari dokumen.



Dokumen SKPL memiliki standard yang berbeda-beda

- IEEE SRS std 830-1998
- BS 6719:1986
- ESA Space Agency Standards
- US DoD Std 7935A
- NASA standard
- Canadian Standard (Z242.15.4-1979)
- dll



Struktur SRS – MIL-STD-498

SOFTWARE REQUIREMENT SPECIFICATION- DHPSC-81433

1. Scope
2. Reference Documents
3. Requirements
 - 3.1 Required states and modes
 - 3.2 CSCI capability requirements
 - 3.3 CSCI external interface requirements
 - 3.4 CSCI internal interface requirements
 - 3.5 CSCI internal data requirements
 - 3.6 Adaptation requirements
 - 3.7 Safety requirements
 - 3.8 Security & privacy requirements
 - 3.9 CSCI environment requirements
 - 3.10 Computer resource requirements
 - 3.11 Software quality factors
 - 3.12 Design and Implementation constraints
 - 3.13 Personnel-related requirements
 - 3.14 Training-related requirements
 - 3.15 Logistics-related requirements
 - 3.16 Other requirements
 - 3.17 Packaging requirements
 - 3.18 Precedence and criticality of requirements
4. Qualification Provisions
5. Requirements Traceability
6. Notes
- A. Appendixes

Dengan struktur seperti ini dokumen lengkap biasanya membutuhkan minimal 10 halaman

MIL-STD- Military Standard, digunakan oleh Department of Defense Amerika



IEEE ***Std 830-1998***

A.1 Template of SRS Section 3 organized by mode: Version 1

71

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 Mode 1

3.2.1.1 Functional requirement 1.1

.

.

.

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Mode 2

.

.

.

3.2.*m* Mode *m*

3.2.*m*.1 Functional requirement *m*.1

.

.

.

3.2.*m*.*n* Functional requirement *m*.*n*

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.6 Other requirements

IF2250 Penulisan SKPL



- 3. Specific requirements
 - 3.1. Functional requirements
 - 3.1.1 Mode 1
 - 3.1.1.1 External interfaces
 - 3.1.1.1.1 User interfaces
 - 3.1.1.1.2 Hardware interfaces
 - 3.1.1.1.3 Software interfaces
 - 3.1.1.1.4 Communications interfaces
 - 3.1.1.2 Functional requirements
 - 3.1.1.2.1 Functional requirement 1
 - 3.1.1.2. n Functional requirement n
 - 3.1.1.3 Performance
 - 3.1.2 Mode 2
 - .
 - .
 - .
 - 3.1. m Mode m
 - 3.2 Design constraints
 - 3.3 Software system attributes
 - 3.4 Other requirements

IEEE

Std 830-1998

A.3 Template of SRS Section 3 organized by user class

73

- 3. Specific requirements
 - 3.1 External interface requirements
 - 3.1.1 User interfaces
 - 3.1.2 Hardware interfaces
 - 3.1.3 Software interfaces
 - 3.1.4 Communications interfaces
 - 3.2 Functional requirements
 - 3.2.1 User class 1
 - 3.2.1.1 Functional requirement 1.1
 - .
 - .
 - 3.2.1.*n* Functional requirement 1.*n*
 - 3.2.2 User class 2
 - .
 - .
 - .
 - 3.2.*m* User class *m*
 - 3.2.*m*.1 Functional requirement *m*.1
 - .
 - .
 - .
 - 3.2.*m*.*n* Functional requirement *m*.*n*
 - 3.3 Performance requirements
 - 3.4 Design constraints
 - 3.5 Software system attributes
 - 3.6 Other requirements

IF2250 Penulisan SKPL



KNOWLEDGE & SOFTWARE ENGINEERING

Dokumen SKPL

***(Tentang dokumen akan
dibantu penjelasannya
oleh asisten)***



Cover

GL01

SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK

<Nama Perangkat Lunak atau Nama Sistem>

untuk:

<Nama User atau Nama Perusahaan>


Dipersiapkan oleh:

<Nomor Grup & Anggota>

Program Studi Teknik Informatika

STEI - ITB

Jl. Ganesha 10, Bandung 40132

	Program Studi Teknik Informatika STEI – ITB	Nomor Dokumen		Halaman
		GL01-SKPL		<#>/<jml #
		Revisi	<nomor revisi>	Tgl: <isi tanggal>

Analisis kebutuhan P/L - Terstruktur



- Daftar Perubahan
 - Versi awal tidak memiliki revisi
 - Revisi pertama, diberikan penjelasan apa yang direvisi
- Dokumen perlu ditandatangani oleh:
 - Yang bertanggung jawab dalam penulisan
 - Pemeriksaan dokumen
 - Penyetujuan dokumen

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	
G	

INDEX TGL	-	A	B	C	D	E	F	G
Ditulis oleh								
Diperiksa oleh								
Disetujui oleh								



KNOWLEDGE & SOFTWARE ENGINEERING

Program Studi Teknik Informatika

SKPL-xx

Halaman 2/ dari 9 halaman

Template dokumen ini dan informasi yang dimilikinya adalah milik Program Studi Teknik Informatika-STEI-ITB dan bersifat rahasia. Dilarang me-reproduksi dokumen ini tanpa diketahui oleh Program Studi Teknik Informatika STEI ITB.

ihan P/L - Terstruktur

- Isikan dengan halaman yang berubah dibandingkan dengan revisi sebelumnya.
- Tuliskan juga nomor revisi

Daftar Halaman Perubahan

Halaman	Revisi	Halaman	Revisi



Daftar Isi

1. Pendahuluan	5
1.1 Tujuan Penulisan Dokumen.....	5
1.2 Lingkup Masalah	5
1.3 Definisi, Istilah dan Singkatan.....	5
1.4 Aturan Penomoran.....	5
1.5 Referensi.....	5
1.6 Deskripsi umum Dokumen (Ikhtisar)	5
2 Deskripsi Umum Perangkat Lunak.....	6
2.1 Deskripsi Umum Sistem.....	6
2.2 Karakteristik Pengguna.....	6
2.3 Batasan	6
2.4 Lingkungan Operasi	6
3 Deskripsi Kebutuhan	7
3.1 Kebutuhan Antarmuka Eksternal.....	7
3.1.1 Antarmuka pemakai.....	7
3.1.2 Antarmuka Perangkat Keras	7
3.1.3 Antarmuka Perangkat Lunak	7
3.1.4 Antarmuka Komunikasi.....	7
3.2 Kebutuhan Fungsional	7
3.2.1 Diagram Konteks	7
3.2.2 DFD Level 1	7
3.2.2.1 DFD Level 2 <??>	7
3.2.2.2 DFD Level 2 <??>	7
3.3 Kebutuhan Data	7
3.3.1 E-R diagram.....	8
3.4 Kebutuhan Non Fungsional	8
3.5 Batasan Perancangan	8
3.6 Keruntutan (traceability).....	8
3.6.1 Data Store vs E-R	8
3.7 Ringkasan Kebutuhan.....	9
3.7.1 Kebutuhan Fungsional	9
3.7.2 Kebutuhan Non Fungsional	9

1. Pendahuluan

1.1 Tujuan Penulisan Dokumen

Tuliskan dengan ringkas tujuan dokumen SKPL ini dibuat, dan digunakan oleh siapa.

1.2 Lingkup Masalah

Tuliskan dengan ringkas nama aplikasi dan deskripsinya. Maksimal 1 paragraf

1.3 Definisi, Istilah dan Singkatan

Semua definisi dan singkatan yang digunakan dalam dokumen ini dan penjelasannya

1.4 Aturan Penomoran

Tuliskan jika anda memakai aturan penomoran

1.5 Referensi

Dokumentasi PL yang dirujuk oleh dokumen ini.

Buku, Panduan, Dokumentasi lain yang dipakai dalam pengembangan PL ini.

1.6 Deskripsi umum Dokumen (Ikhtisar)

Tuliskan sistematika pembahasan dokumen SKPL ini.



KNOWLEDGE & SOFTWARE ENGINEERING

2 Deskripsi Umum Perangkat Lunak

2.1 Deskripsi Umum Sistem

Tuliskan overview P/L, dalam bentuk gambar dan narasi yang dapat memberikan gambaran tentang aplikasi dan konteksnya, yaitu hubungannya dengan dunia luar (gambar yang mirip dengan diagram konteks, tetapi dengan notasi yang lebih mudah dimengerti orang awam).

2.2 Karakteristik Pengguna

Minimal sebuah tabel dengan Kolom : Pengguna, Pekerjaan, Hak Akses. Kolom Hak Akses dihubungkan dengan Fungsi utama yang muncul pada Fungsi Produk

Kategori Pengguna	Tugas	Hak Akses ke aplikasi

2.3 Batasan

Batasan (jika ada), ketergantungan SW terhadap SW/HW/sistem lain (misalnya modul Konsolidasi baru dapat dijalankan ketika rekapitulasi data akuntansi dari Aplikasi AKUNT sudah dijalankan dan datanya dinyatakan OK oleh petugas

Batasan yang harus dipakai. Misalnya :

- harus memakai file data dari Sistem lain (sebutkan),
- harus memakai format data yang sama dengan sistem lain
- harus berfungsi multi platform (di Windows dan linux)

2.4 Lingkungan Operasi

Operating system, DBMS, ...

Aplikasi Client server ini akan berfungsi dengan spesifikasi :

Server : ???

Client : ????

OS :

DBMS :

Analisis Kebutuhan P/L - Terstruktur



Hanya diisi jika P/L memerlukan fasilitas khusus.

Diawali dengan membuat daftar kebutuhan fungsional P/L, lengkap dengan ID dan penjelasan jika perlu. Bisa dibuat dalam bentuk tabel.

[illegible]

Diisi untuk kebutuhan kuliah basisdata.

Analisis Kebutuhan P/L - Terstruktur



3.3.1 E-R diagram

3.4 Kebutuhan Non Fungsional

Uraikan dengan ringkas kebutuhan non fungsional dalam tabel sebagai berikut. Isilah Kolom Kebutuhan dengan kalimat yang jelas dan kelak dapat dites untuk dipenuhi. ID adalah nomor kebutuhan yang harus ditelusuri pada saat test. Tuliskan N/A bila Not Applicable..

ID	Parameter	Kebutuhan
	Availability	
	Reliability	
	Ergonomy	
	Portability	
	Memory	
	Response time	
	Safety	N/A
	Security	
	Others 1: Bahasa komunikasi	Misalnya : semua tanya jawab harus dalam bahasa Indonesia
		Setiap layar harus mengandung logo PT Pos Indonesia

Catatan :

Availability : ketersediaan aplikasi, misalnya harus terus menerus beroperasi 7 hari perminggu, 24 jam per haritanpa gagal

Reliability : keandalan, misalnya tidak pernah boleh gagal(atau kegagalan yang ditolerir adalah ...%) sehingga harus dipikirkan fault tolerant architecture. Biasanya hanya perlu untuk Critical Application yang jika gagal akan berakibat fatal.

Ergonomy : kenyamanan pakai bagi pengguna

Portability : kemudahan untuk dibawa dan dioperasikan ke mesin/sistem operasi/platform yang lain

Memory : jika perhitungan kapasitas memori internal kritis (misalnya untuk SW yang harus dijadikan CHIPS dan ukurannya harus kecil

Response time : Batasan waktu yang harus dipenuhi. Sangat penting untuk aplikasi Real Time. Contoh:

"Aplikasi harus mampu menampilkan hasil dalam 4 detik", atau "ATM harus menarik kembali kartu yang tidak diambil dalam waktu 3 menit"

Safety: yang menyangkut keselamatan manusia, misalnya untuk SW yang dipakai pada sistem kontrol di pabrik

Security : aspek keamanan yang harus dipenuhi.

3.5 Batasan Perancangan

Sebutkan batasan perancangan jika ada. Contoh : harus memakai library yang ada, harus memakai sepotong kode yang sudah pernah dikembangkan, harus memperhatikan hal-hal tertentu

3.6 Kerunutan (traceability)

Diisi dengan tabel yang berisi traceability dari hasil analisis. Gunanya untuk menilai apakah hasil analisis "rumut" dan lojik. Untuk sementara, baru didefinisikan Data-store versus E-R.

3.6.1 Kebutuhan Fungsional vs Proses

Mapping kebutuhan fungsional dengan proses pada DFD

ID Kebutuhan Fungsional	Nomor Proses pada DFD

Analisis Kebutuhan P/L - Terstruktur



3.6.2 Data Store vs E-R

Mapping data store pada DFD dengan Entity - Relasi

Data Store	Entity	Relasi

3.7 Ringkasan Kebutuhan

Bab ini berisi ringkasan semua kebutuhan. Kebutuhan ini mencerminkan semua hal yang harus dipenuhi, dan nantinya akan menjadi arahan untuk tahapan testing, karena pada dasarnya, semua kebutuhan harus dapat dites supaya dapat dibuktikan dipenuhi. Dibagi menjadi dua bagian: fungsional dan non fungsional.

3.7.1 Kebutuhan Fungsional

ID	Deskripsi

3.7.2 Kebutuhan Non Fungsional

ID	Deskripsi

Analisis Kebutuhan P/L - Terstruktur



Analisis Kebutuhan dan Pemodelan



KNOWLEDGE & SOFTWARE ENGINEERING

IF2250 Penulisan SKPL

Pemodelan Kebutuhan

- Spesifikasi kebutuhan perlu diperjelas dengan suatu model atau diagram
- Pemodelan hingga saat ini menggunakan 2 pendekatan
 - Model dengan pendekatan Terstruktur (**Structured Approach**) – Tradisional/Konvensional
 - Diagram konteks – DFD
 - Diagram ER
 - Model dengan pendekatan Obyek (**Object Orientation**) – lebih baru, diagram UML (Unified Modeling Language)
 - Diagram Use case
 - Diagram Kelas/Objek
 - Diagram Sekuens, dll



Model Kebutuhan Apa Yang Dipilih?

- Selain pendekatan terstruktur dengan pemodelan Diagram Konteks, untuk pendekatan lain dapat juga dengan pendekatan berorientasi objek
 - Use case (UC) dapat digunakan untuk tujuan yang sama dengan Diagram Konteks (DK).
- Diagram Konteks dan Use case digunakan untuk menggambarkan interaksi sistem dengan entitas/aktor yang terkait
 - DK dapat menggambarkan hubungan antara entitas dengan sistem yang akan dikembangkan
 - DK memberikan deskripsi aliran data dari/ke suatu entitas dari/ke sistem
 - Berawal dari ide pendekatan **Input-Proses-Output**
 - UC menggambarkan suatu aktor dapat melakukan atau memiliki kemampuan apa saja terhadap sistem yang akan dikembangkan.
- Menggunakan DK harus dilengkapi dengan proses dan aliran data yang lebih rinci untuk melengkapi model kebutuhannya.
 - Model ini lebih dikenal dengan nama DFD (Data Flow Diagram)
 - Menggunakan pendekatan terstruktur (Structured Approach)
- Use Case hanya fokus pada kebutuhan yang ada pada sistem tanpa terlalu memikirkan data apa yang akan diperlukan oleh sistem.
 - Model ini banyak digunakan untuk pendekatan berorientasi objek (Object Oriented Approach)



Interaksi dengan Sistem

- Pada tahap awal pengembangan perangkat lunak kita harus mendefinisikan **siapa** atau **apa** saja yang terkait dengan sistem/perangkat lunak(software) yang akan dibuat.
- Pengembangan model diperlukan untuk memudahkan analisa dan juga komunikasi
 - Diagram konteks dapat digunakan untuk menggambarkan siapa saja yang berinteraksi dengan sistem
 - Use-case digunakan untuk menggambarkan keterlibatan 'aktor' dengan sistem yang akan dikembangkan.



Diagram Konteks VM

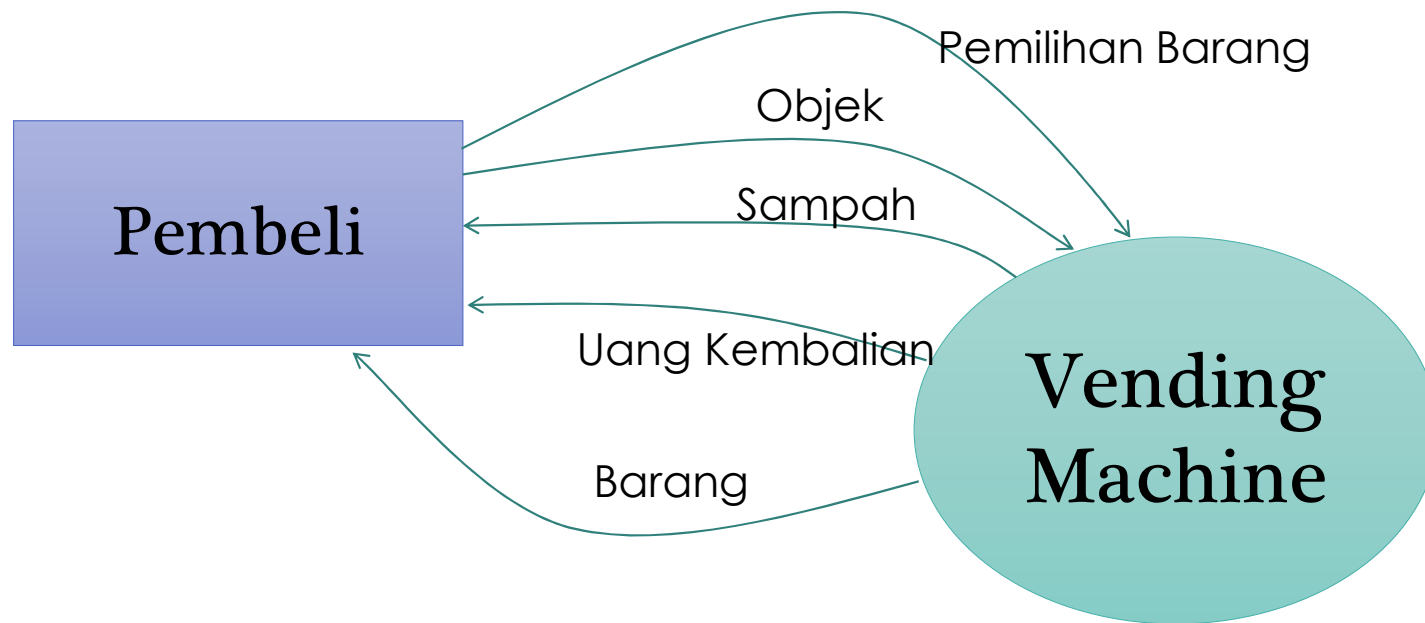
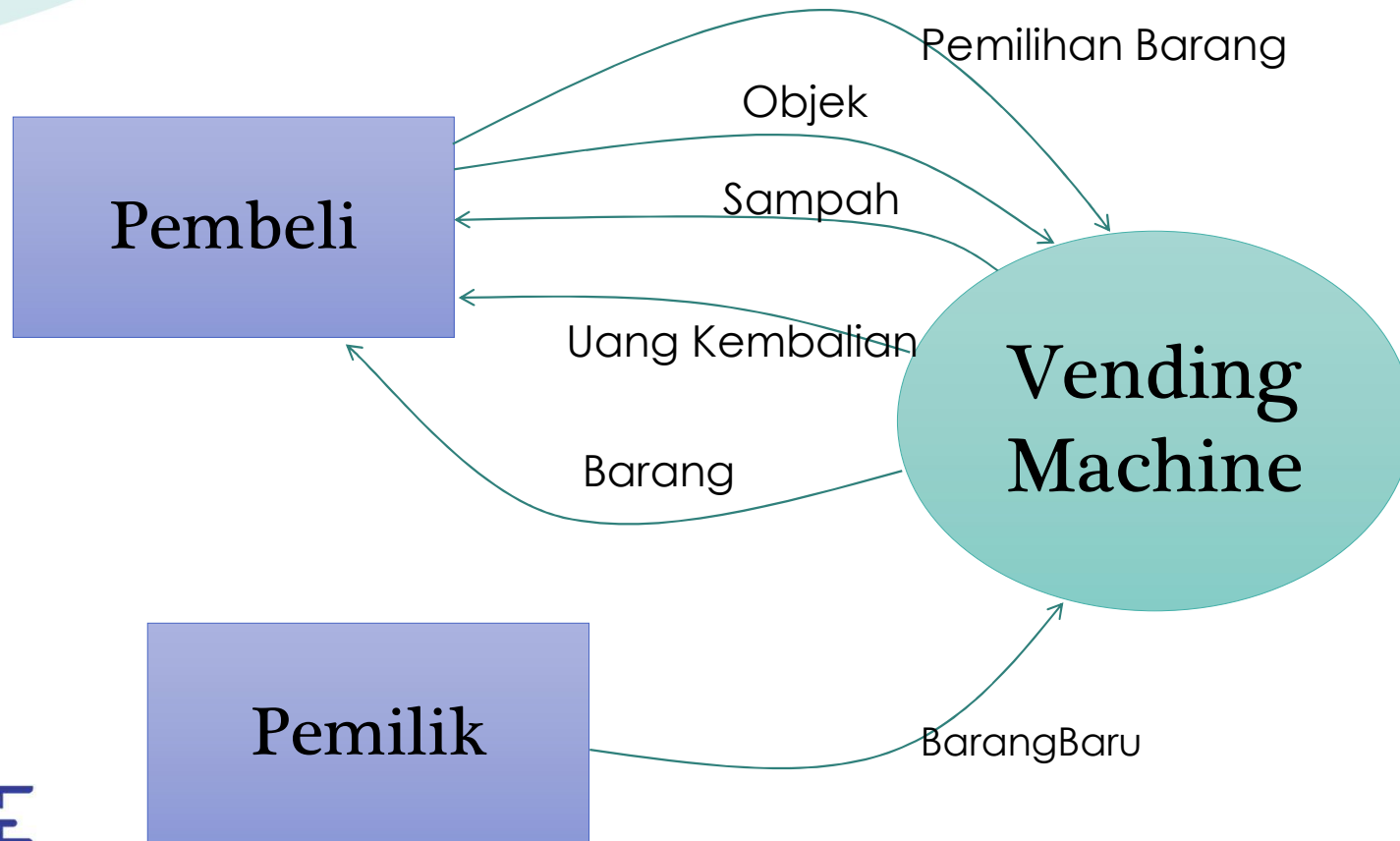


Diagram Konteks VM (Lengkap)



Catatan

- Diagram Konteks biasanya digunakan dalam pendekatan model struktural
- Use-Case biasanya digunakan dalam pendekatan pemodelan berorientasi objek
 - akan dijelaskan di bagian slide terakhir
- Bentuk diagram di atas mungkin bukan solusi tunggal, solusi lain bisa ditawarkan tergantung pemahaman kita, dan juga jenis/kategori persoalan yang akan dipecahkan
 - Jumlah aktor atau entitas luar yang terlibat mungkin bertambah
 - Contoh : Pemilik Sistem, User, Sistem Bank, Sensor, dan lain-lain
- Diagram ini digunakan sebagai
 - Pemodelan awal yang dapat digunakan sebagai alat komunikasi dengan user atau customer.
 - Menunjukkan ruang lingkup permasalahan



Checklist untuk validasi kebutuhan

- Apakah semua kebutuhan sudah dinyatakan dengan jelas? Hati-hati salah interpretasi
- Apakah sumber kebutuhan sudah teridentifikasi dengan jelas?
 - Sumber dapat berasal dari 'nama orang', nama dokumen,
 - Apakah SKPL sudah dicek lagi dengan sumber ?
- Apakah semua kebutuhan sudah jelas secara 'kuantitatif'
- Kebutuhan apa yang mungkin berhubungan atau terkait dengan kebutuhan lain
 - Harus jelas dinyatakan bagaimana hubungan atau kaitannya
 - Harus jelas urutan kebutuhannya
- Apakah suatu kebutuhan tidak sesuai dengan 'constraint' dari suatu domain
 - Misalnya: apakah meminjam 10 buku sesuai dengan aturan organisasi perpustakaan
- Apakah suatu kebutuhan dapat diselesaikan (sesuai dengan rencana waktu dan biaya)?
- Apakah kebutuhan non-fungsional telah terdefinisi dengan baik bagaimana merealisasikannya
 - Misalnya kebutuhan 'aplikasi' harus cepat, harus terdefinisi 'cepat' itu bagaimana?
 - Apakah 10 transaksi per detik? Atau penyimpanan suatu data tidak lebih dari 1 detik.

