Pendidikan dan Pelatihan Profesi Guru (PLPG) TIK
Gelombang 14

Rekayasa Perangkat Lunak

JURUSAN PENDIDIKAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK UNIVERSITAS NEGERI YOGYAKARTA

2 0 0 8

Pendahuluan

Yang dimaksud dengan perangkat lunak adalah :

1. Perintah (program komputer) yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang diharapkan.
2. Struktur data yang memungkinkan program memanipulasi informasi secara proporsional.
3. Dokumen yang menggambarkan operasi dan kegunaan program.

Pemodelan dalam suatu rekayasa perangkat lunak merupakan suatu hal yang dilakukan di tahapan awal. Dalam rekayasa perangkat lunak sebenarnya masih memungkinkan tanpa melakukan suatu pemodelan. Hal itu tidak dapat lagi dilakukan dalam suatu industri perangkat lunak karena dengan pemodelan maka akan lebih mudah untuk memahami sistem baik pengembang perangkat lunak itu sendiri maupun pelanggan. Dengan demikian pengembang akan lebih cepat dalam melakukan desain dan mengkontruksi program untuk perangkat lunak tersebut berdasarkan model yang sudah ada dan telah disepakati bersama. Pemodelan ini juga sering disebut dengan metodologi pengembangan sistem.

Proses

Di dalam suatu industri dikenal berbagai macam proses, demikian juga halnya dengan industri perangkat lunak. Perusahaan yang berbeda seringkali menggunakan proses yang berbeda untuk menghasilkan produk yang sama. Tetapi produk yang berbeda mungkin dihasilkan oleh sebuah perusahaan dengan menggunakan proses yang sama. Jika proses yang digunakan salah maka akan mengurangi kualitas dari produk yang dikembangkan.

Seperti produk, proses juga memiliki atribut dan karakteristik seperti :

- *Understandability*, yaitu sejauh mana proses secara eksplisit ditentukan dan bagaimana kemudahan definisi proses itu dimengerti.

- *Visibility*, apakah aktivitas-aktivitas proses mencapai titik akhir dalam hasil yang jelas sehingga kemajuan dari proses tersebut dapat terlihat nyata/jelas.
- *Supportability*, yaitu sejauh mana aktivitas proses dapat didukung oleh CASE.
- *Acceptability*, apakah proses yang telah ditentukan oleh insinyur dapat diterima dan digunakan dan mampu bertanggung jawab selama pembuatan produk perangkat lunak.
- *Reliability*, apakah proses didesain sedikian rupa sehingga kesalahan proses dapat dihindari sebelum terjadi kesalahan pada produk.
- *Robustness*, dapatkah proses terus berjalan walaupun terjadi masalah yang tak diduga.
- *Maintainability*, dapatkah proses berkembang untuk mengikuti kebutuhan atau perbaikan.
- *Rapidity*, bagaimana kecepatan proses pengiriman sistem dapat secara lengkap memenuhi spesifikasi.

Model

Model proses perangkat lunak masih menjadi objek penelitian. Ada banyak model umum atau paradigma yang digunakan untuk mengembangan perangkat lunak, antara lain:

- Pendekatan Waterfall

  Berisi rangkaian aktivitas proses yaitu spesifikasi kebutuhan, implementasi desain perangkat lunak, uji coba dst. Setelah setiap langkah didefinisikan, langkah tersebut di *sign off* dan pengembangan dilanjutkan pada langkah berikutnya.

- Pengembangan secara evolusioner

  Sistem awal dengan cepat dikembangkan dari pelanggan untuk memproduksi sistem yang memenuhi kebutuhan pelanggan tersebut kemudian sistem disampaikan. Sistem itu mungkin diimplementasikan kembali dengan pendekatan yang lebih terstruktur untuk menghasilkan sistem yang kuat dan maintable.

- Transformasi formal

  Pendekatan ini berdasarkan pembuatan spesifikasi sistem formal secara matematik dan transformasi spesifikasi dengan menggunakan metode matematik atau dengan suatu program. Transformasi ini adalah *correctness preserving*, ini berarti bahwa kita dapat yakin program yang dikembangkan sesuai dengan spesifikasi.

- Penggabungan sistem dengan menggunakan komponen-komponen yang dapat digunakan kembali (*reusable*).

  Teknik ini menganggap bagian-bagian dari sistem sudah ada. Proses pengembangan sistem lebih berfokus pada penggabungan bagian-bagian daripada pengembangan tiap bagian yang sudah ada tersebut.

- Berorientasi objek

  Teknik ini merupakan teknik terbaru yang sekarang banyak digunakan dan terus dikembangkan. Teknik ini memandang segala sesuatu sebagai objek sehingga dengan mudah pengembang memahami sistem yang akan dikembangkannya.

Secara umum metodologi ini meliputi serangkaian tugas yang luas seperti analisis kebutuhan, desain, konstruksi program, pengujian, dan pemeliharaan.

Definisi Analisis Kebutuhan Sistem

Menurut Yogiyanto (1995) analisis sistem adalah penguraian dari suatu sistem informasi yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan. Sedangkan menurut Kristanto (2003) analisis sistem adalah suatu proses mengumpulkan dan menginterpretasikan kenyataan-kenyataan yang ada, mendiagnosa persoalan dan menggunakan keduanya untuk memperbaiki sistem.

Analis Kebutuhan Sistem

Menurut Yogiyanto (1995) analis sistem (analis informasi) adalah orang yang menganalis sistem (mempelajari masalah-masalahan yang timbul dan menentukan kebutuhan pemakai sistem) untuk mengidentifikasikan pemecahan permasalahan tersebut. Sedangkan menurut Kristanto (2003) analis sistem adalah orang yang mempunyai kemampuan untuk menganalisis sebuah sistem, memilih alternatif pemecahan masalah dan menyelesaikan masalah tersebut dengan menggunakan komputer.

Peranan Analis Kebutuhan Sistem

Analis sistem secara sistematis menilai bagaimana fungsi bisnis dengan cara mengamati proses input dan pengolahan data serta proses output informasi untuk membantu peningkatan proses organisasional. Dengan demikian, analis sistem mempunyai tiga peranan penting, yaitu :

1. Sebagai konsultan
2. Sebagai ahli pendukung
3. Sebagai agen perubahan

Adapun tugas-tugas yang dilakukan oleh seorang analis sistem adalah sebagai berikut :

1. mengumpulkan dan menganalisis semua dokumen, file, formulir yang digunakan pada sistem yang telah berjalan.
2. menyusun laporan dari sistem yang telah berjalan dan mengevaluasi kekurangan-kekurangan pada sistem tersebut dan melaporankan semua kekurangan tersebut kepada pemakai sistem.
3. merancang perbaikan pada sistem tersebut dan menyusun sistem baru.
4. menganalisis dan menyusun perkiraan biaya yang diperlukan untuk sistem yang baru dan memberikan argumen tentang keuntungan yang dapat diperoleh dari pemakian sistem yang baru tersebut.
5. mengawasi semua kegiatan terutama yang berkaitan dengan sistem yang baru tersebut.

Waterfall

Model ini menawarkan cara pembuatan perangkat lunak secara lebih nyata. Langkah-langkah yang penting dalam model ini adalah :

- Penentuan dan analisis spesifikasi

  Jasa, kendala dan tujuan dihasilkan dari konsultasi dengan pengguna sistem. Kemudian semuanya itu dibuat dalam bentuk yang dapat dimengerti oleh user dan staf pengembang.

- Desain sistem dan perangkat lunak

  Proses desain sistem membagi kebutuhan-kebutuhan menjadi sistem perangkat lunak atau perangkat keras. Proses tersebut menghasilkan sebuah arsitektur sistem keseluhan. Desain perangkat lunak termasuk menghasilkan fungsi sistem perangkat lunak dalam bentuk yang mungkin ditransformasi ke dalam satu atau lebih program yang dapat dijalankan.

- Implementasi dan ujicoba unit

  Selama tahap ini desain perangkat lunak disadari sebagai sebuah program lengkap atau unit program. Uji unit termasuk pengujian bahwa setiap unit sesuai spesifikasi.

- Integrasi dan ujicoba sistem

  Unit program diintegrasikan dan diuji menjadi sistem yang lengkap untuk menyakinkan bahwa persyaratan perangkat lunak telah dipenuhi. Setelah ujicoba, sistem disampaikan ke pelanggan

- Operasi dan pemeliharaan

  Normalnya, ini adalah phase yang terpanjang. Sistem dipasang dan digunakan. Pemeliharaan termasuk pembetulan kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru ditemukan.

Dalam prakteknya, setiap langkah sering tumpang tindih dan saling memberi informasi satu sama lain. Proses perangkat lunak tidak linier dan sederhana tapi mengandung urutan iterasi dari aktivitas pengembangan. Selama di langkah terakhir, perangkat lunak telah digunakan. Kesalahan dan kelalaian dalam menentukan kebutuhan perangkat lunak original dapat diatasi.

Sayangnya, model yang banyak mengandung iterasi sehingga membuat sulit bagi pihak manajemen untuk memeriksa seluruh rencana dan laporan. Maka dari itu, setelah sedikit iterasi, biasanya bagian yang telah dikembangkan akan dihentikan dan dilanjutkan dengan langkah pengembangan selanjutnya. Masalah-masalah selama resolusi selanjutnya, dibiarkan atau diprogram. Pemberhentian yang prematur dari persyaratan akan berarti bahwa sistem tidak akan sesuai dengan keinginan user. Mungkin juga sistem terstruktur secara jelek yang sebenarnya merupakan masalah desain akan dibiarkan karena terkalahkan oleh trik implementasi.

Masalah pendekatan waterfall adalah ketidakluwesan pembagian project ke dalam langkah yang nyata/jelas. Sistem yang disampaikan kadang-kadang tidak dapat digunakan sesuai keinginan pelanggan. Namun demikian model waterfall mencerminkan kepraktisan engineering. Konsekuensinya, model proses perangkat lunak yang berdasarkan pada pendekatan ini digunakan dalam pengembangan sistem perangkat lunak dan hardware yang luas.

Pengembangan secara Evolusioner

Model ini berdasarkan pada ide pengembangan pada implementasi awal yang akan menghasilkan komentar pemakai sehingga dapat dilakukan perbaikan melalui banyak versi sampai sistem yang mencukupi dapat dikembangan. Selain memiliki aktivitas-aktivitas yang terpisah model ini memberikan feedback dengan cepat dan serentak.

Terdapat 2 tipe pada model ini, yaitu :

1. Pemprograman evolusioner

   Dimana tujuan proses adalah bekerjasama dengan pelanggan untuk menghasilkan kebutuhan-kebutuhan dan menyampaikan sistem akhir kepada pemakai atau pelanggan. Pengembangan dimulai dengan bagian-bagian sistem yang dimengerti. Sistem dikembangkan melalui penambahan features sesuai yang diusulkan oleh pelanggan.

2. Pemodelan

   Dimana tujuan pengembangan evolusioner pada tipe ini adalah mengetahui kebutuhan-kebutuhan pelanggan dan mengembangkan difinisi kebutuhan yang lebih baik untuk sistem. Model/contoh difikuskan pada penelitian bagian-bagian kebutuhan pelanggan yang kurang dimengerti.

Pemprograman evolusioner penting saat sulit untuk membuat spesifikasi sistem secara rinci. Beberapa orang mungkin setuju bahwa semua sistem masuk dalam tipe ini. Namun, pemprograman evolusioner banyak digunakan dalam pengembangan sistem *AI* (*artificial intelligence*) yang berusaha untuk menyamai kemampuan manusia. Kita tidak mungkin membuat spesifikasi yang rinci untuk perangkat lunak yang menyamai manusia karena kita tidak mengerti bagaimana manusia menjalankan tugas-tugas mereka.

Pendekatan evolusioner biasanya lebih efektif daripada pendekatan waterfall untuk hal pengembangan perangkat lunak yang harus dengan segera dapat memenuhi kebutuhan pelanggan. Namun, dari segi teknik dan manajemen, model ini memiliki masalah mendasar yaitu:

- Proses tidak visibel.

  Manager-manager membutuhkan "deliverables" yang teratur untuk mengukur kemajuan. Jika sistem dikembangkan dengan cepat akan terjadi

pemborosan pada pembuatan dokumen yang menggambarkan setiap versi sistem.

- Sistem-sistem biasanya kurang terstruktur

  Kecenderungan perubahan yang terus menerus akan mengurangi stuktrur dari perangkat lunak. Evolusi perangkat lunak terlihat sulit dan mahal.

- Ketrampilan khusus jarang dimiliki

  Tidak jelas batasan ketrampilan yang normal dalam rekayasa perangkat lunak yang mungkin dapat digunakan secara efektif dalam model pengembangan ini. Kebanyakan sistem yang dikembangkan melalui cara ini telah diimplementasikan oleh kelompok kecil yang memiliki ketrampilan yang tinggi dan motivasi yang kuat.

Untuk memecahkan masalah-masalah tersebut, kadang-kadang tujuan dari pengembangan evolusioner adalah mengembangkan contoh sistem. Contoh ini digunakan untuk mengerti dan mevalidasikan spesifikasi sistem. Disinilah pengembangan evolusioner merupakan bagian dari beberapa proses yang lebih luas (seperti model waterfall). Karena masalah-masalah tersebut, sistem dengan skala besar biasanya tidak dikembangkan melalui cara ini. Pengembangan evolusioner lebih tepat untuk pengembangan sistem yang relatif kecil.

Masalah-masalah mengenai perubahan sistem yang ada dihindari dengan mengimplementasi ulang sistem keseluruhan kapanpun perubahan yang signifikan diperlukan. Jika pemodelan digunakan, tidak terlalu mahal.

Spiral Boehm

Model proses nyata waterfall yang berorientasi dokumen telah diambil sebagai standar umum oleh banyak agen pemerintah dan pembuat perangkat lunak. Jadi, tidak mudah melupakan model tersebut walaupun masih terdapat

masalah-masalah yang ditimbulkan dalam model tersebut. Kita membutuhkan sebuah proses yang lebih baik untuk manajemen yang dapat menggunakan semua model umum seperti yang telah kita bicarakan sebelumnya. Model perbaikan tersebut juga harus memenuhi kebutuhan-kebutuhan pembuat perangkat lunak. Pendekatan alternatif diusulkan oleh Boehm (1988). Boehm mengusulkan sebuah model yang secara eksplisit menjelaskan bahwa resiko yang disadari mungkin membentuk dasar model proses umum.

Model Boehm bebrbentuk spiral. Setiap loop mewakili sebuah tahap dari proses perangkat lunak. Tidak ada tahap yang tetap dalam model ini. Manajemen harus memutuskan bagaimana membentuk proyek kedalam tahap-tahap. Perusahaan biasanya bekerja dengan beberapa model umum dengan tahap tambahan untuk proyek khusus atau ketika masala-masalah ditemukan selama pembuatan proyek.

Setiap loop dibagi dalam 4 sektor, yaitu :

1. Pembuatan tujuan

   Tujuan, hambatan dalam proses ataupun produk serta resiko-resiko proyek ditentukan. Rencana rinci manajemen juga ditulis lengkap. Pembuatan strategi-strategi alternatif direncanakan sesuai dengan resiko yang ada.

2. Perkiraan dan pengurangan resiko

   Untuk setiap resiko yang telah diidentifikasi, akan dibuat analisis rincinya. Kemudian diambil langkah-langkah untuk mengurangi resiko. contohnya, jika ada resiko bahwa persyaratan-persyaratan tidak tepat maka sebuah model contoh mungkin dapat dikembangkan.

3. Pengembangan dan validasi

   Setelah evaluasi resiko, sebuah model pengembangan untuk sistem dipilih. Misalnya, jika resiko interface pengguna yang dominan maka model pengembangan yang tepat mungkin pengembangan evolusioner dengan menggunakan model contoh (prototipe).

Jika resiko keselamatan yang diutamakan, model pengembangan yang sesuai adalah transformasi formal dan seterusnya. Model waterfall mungkin tepat digunakan jika resiko yang diutamakan adalah integrasi sistem.

4. Perencanaan

   Jika diputuskan untuk melanjutkan pada loop spiral berikutnya maka proyek dibicarakan kembali dan rencana dibuat untuk tahap selanjutnya.

Tidak perlu untuk menggunakan satu model tunggal pada setiap loop spiral bahkan dalam keseluruhan sisten perangkat lunak. Model spiral encompasses model lainnya. Pemodelan digunakan pada salah satu psiral untuk memecahkan masalah kebutuhan. Kemudian dapat diikuti oleh model konvensional, waterfall. Transformasi formal digunakan untuk mengembangkan bagian-bagian sistem yang memiliki persyaratan keselamatan yang tinggi dan pendekatan reuse digunakan untuk pengimplementasian bagian-bagian lain dari sistem data manajemen.

Pada implementasinya, model spiral ini juga banyak digunakan, tetapi biasanya dikombinasikan dengan model yang lain. Pemodelan waterfall, yang sangat bagus dalam menentukan millestones dan pemodelan spiral, yang sangat bagus dengan menggunakan prototyping, merupakan kombinasi yang sering dipakai di dalam kontrak-kontrak untuk perangkat lunak dewasa ini.

Manajemen Resiko

Perbedaan yang mendasar antara model spiral dengan model lainnya adalah bahwa model spiral dengan eksplisit menyadari resiko-resiko yang ada. Resiko adalah konsep yang sulit didefinisikan secara tepat. Secara informal resiko adalah sesuatu yang sederhana yang dapat menyebabkan kesalahan. Contohnya, jika bertujuan menggunakan pemprograman bahasa baru (new programming

language), resiko yang mungkin adalah alat pengumpul yang digunakan tidak reliabel dan tidak menghasilkan code objek yang efesien.

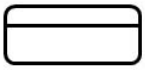
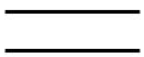
Resiko adalah sebagai hasil ketidakcukupan informasi. Resiko tersebut dapat dipecahkan dengan pengenalan beberapa kegiatan yang dapat menutupi informasi yang kurang menyakinkan. Dalam contoh diatas, resiko mungkin dapat diatasi dengan survey pasar untuk menemukan alat pengumpul mana yang dapat digunakan dan bagaimana kebaikan alat tersebut. Jika sistem ternyata tidak sesuai maka keputusan untuk menggunakan bahasa baru harus diubah.

Siklus spiral dimulai dengan penguraian tujuan-tujuan seperti performance, kegunaan, dan seterusnya. Cara alternatif dalam pencapaian tujuan dan hambatan dipergunakan dengan sebaik-baiknya kemudian diperhitungkan. Setiap alternatif diperhitungan bertentangan dengan tujuan. Ini biasanya menghasilkan identifikasi sumber resiko proyek. Langkah selanjutnya adalah mengevaluasi resiko-resiko ini dengan aktivitas seperti analisis yang lebih detail, pembuatan model/contoh, simulasi dan seterusnya. Untuk menggunakan model spiral, Boehm menyarankan sebuah bentuk umum yang dipenuhi dalam setiap daerah spiral. Bentuk ini mungkin dilengkapi pada sebuah level abtrak atau perkiraan rinci yang imbang dari pengembangan produk.

Analisis Kebutuhan

Untuk menganalisis kebutuhan sistem, tool yang biasa dipakai adalah DFD (Data Flow Diagram). Jadi DFD ini sangat penting bagi seorang analis sistem. Penggunaan DFD sebagai Modeling Tool dipopulerkan Oleh Demacro & Yordan (1979) dan Gane & Sarson (1979) dengan menggunakan pendekatan Metoda Analisis Sistem Terstruktur.

DFD menggambarkan arus data dari suatu sistem informasi, baik sistem lama maupun sistem baru secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut berada.

Demacro & Yordan	Keterangan	Gane & Sarson
	External Entity	
	Process	
	Data Flow	
	Data Store	

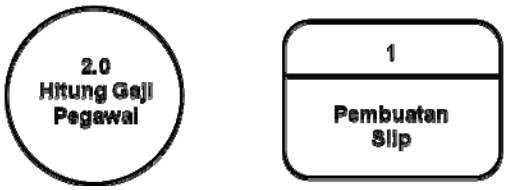
Simbol dalam DFD

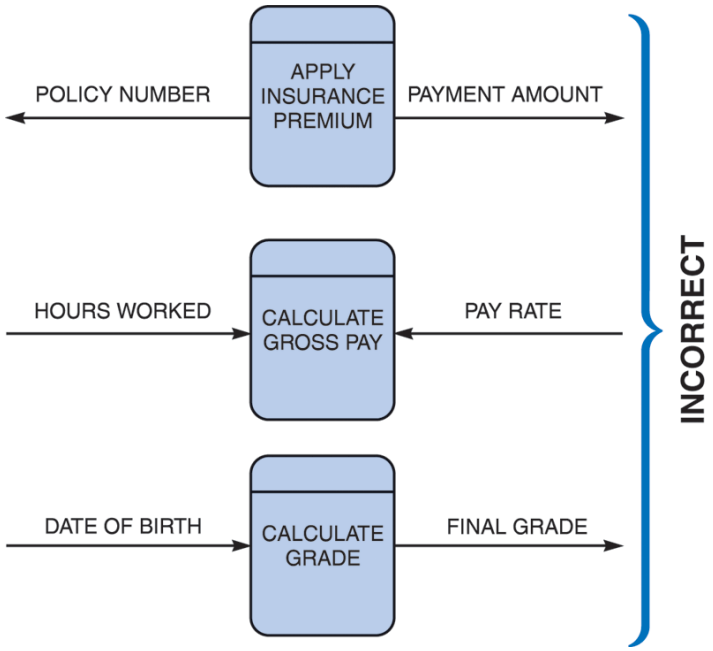
External Entity

External Entity adalah merupakan entitas diluar sistem yang berinteraksi dengan sistem yang akan dikembangkan. External Entity ini dapat berupa orang, organisasi, maupun sistem yang lain. Contoh dari External Entity ini adalah Pelanggan, Mahasiswa, Yayasan, dan lain sebagainya.

Process

Merupakan kegiatan atau pekerjaan yang dilakukan oleh orang atau mesin komputer, dimana aliran data masuk kemudian ditranformasikan keluar (aliran data keluar).





Data Flow

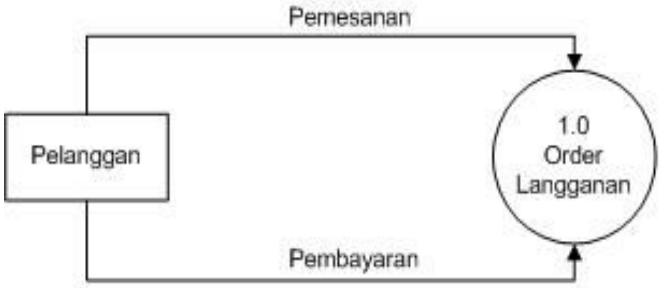
Data Flow merupakan aliran data yang masuk maupun keluar dari suatu Process dan disimbolkan dengan anak panah. Aliran data dapat berbentuk sebagai berikut :

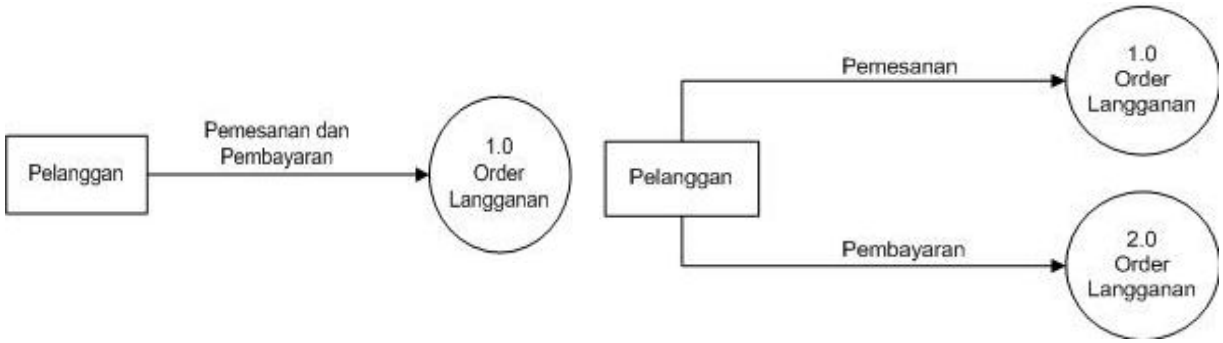
a. Formulir atau dokumen yang digunakan perusahaan
b. Laporan tercetak yang dihasilkan sistem
c. Output dilayar komputer
d. Masukan untuk komputer
e. Komunikasi ucapan
f. Surat atau memo
g. Data yang dibaca atau direkam di file
h. Suatu isian yang dicatat pada buku agenda
i. Transmisi data dari suatu komputer ke komputer lain

Data ada tiga kemungkinan, antara lain :

1. Packet of Data

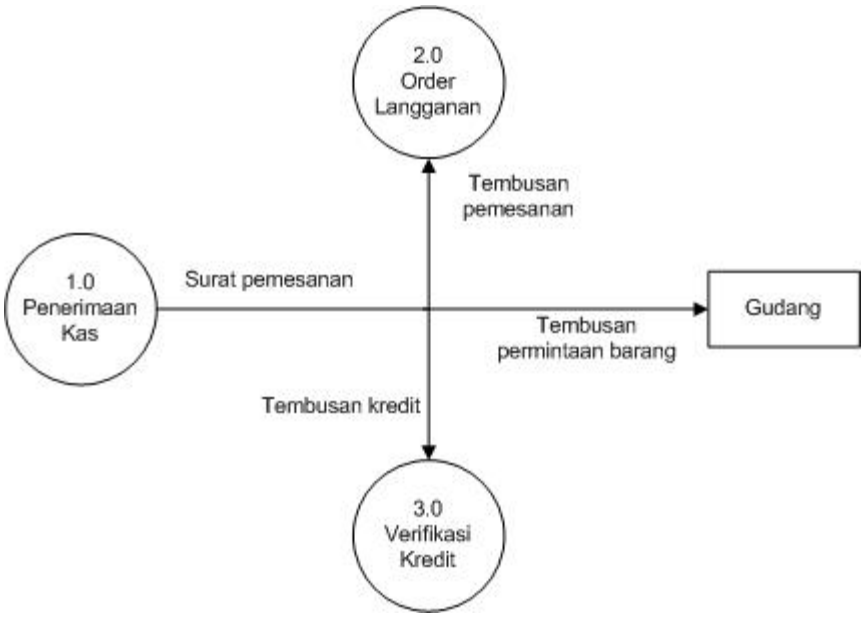
   Bila dua data mengalir dari suatu sumber yang sama ke tujuan yang sama, maka harus dianggap sebagai suatu aliran data yang tunggal.





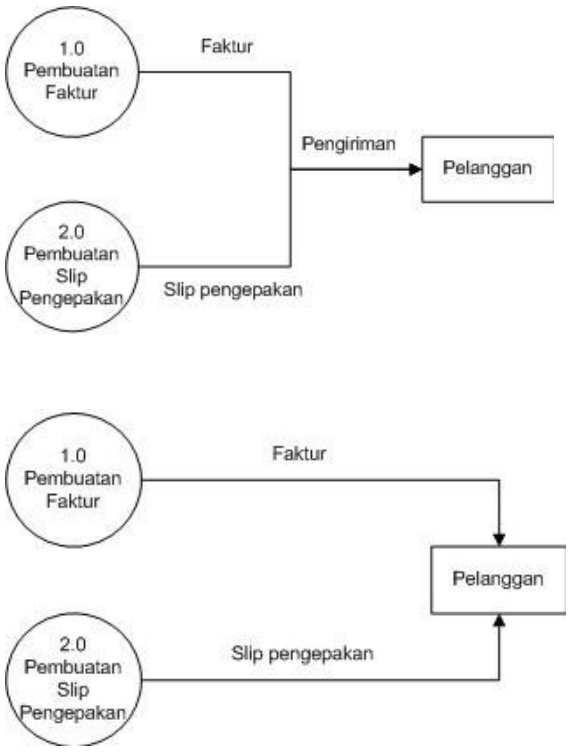
2. Diverging Data Flow

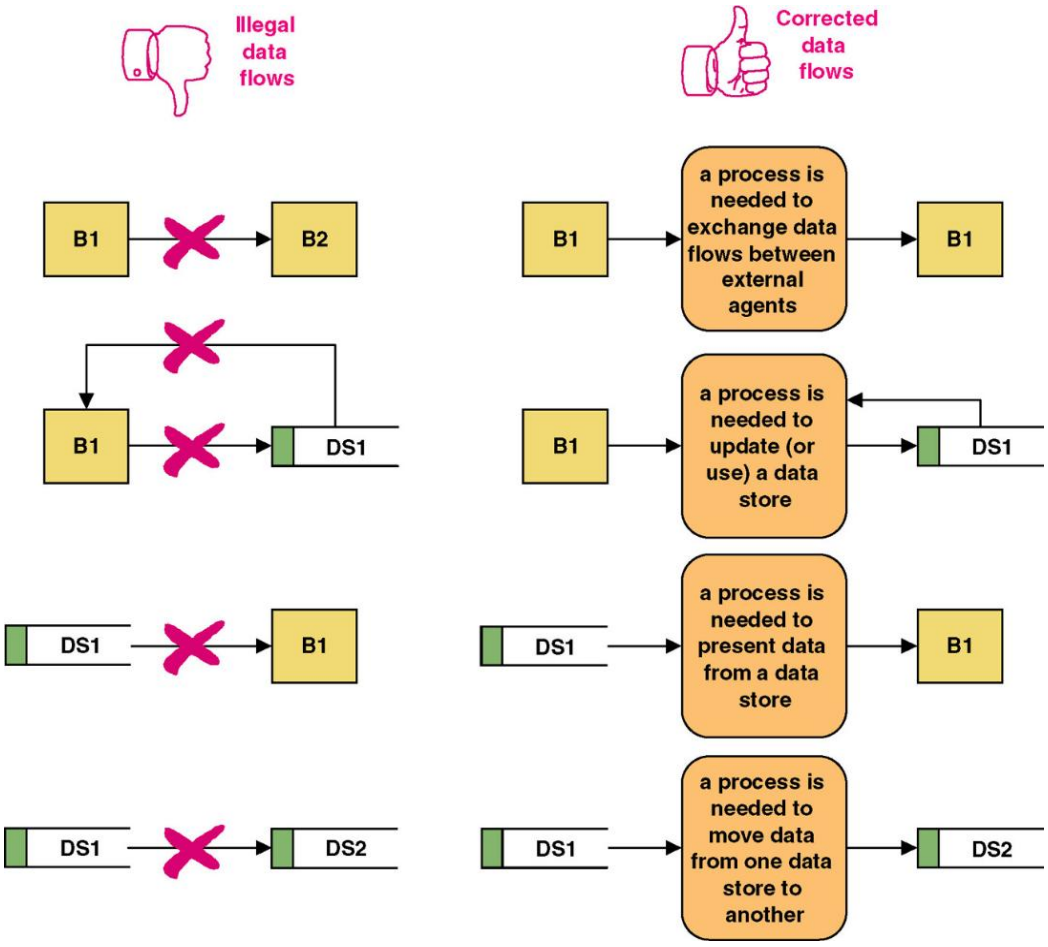
   Bila dari suatu sumber mengalir data yang menyebar ke tujuan yang berbeda, menunjukan bahwa aliran data tersebut merupakan tembusan dari aliran data.



3. Convergen Data Flow

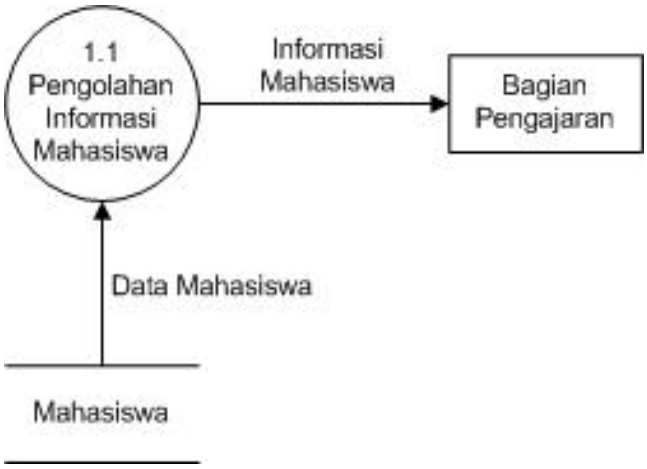
Data yang mengalir dari sumber yang berbeda menuju ke tujuan yang sama.





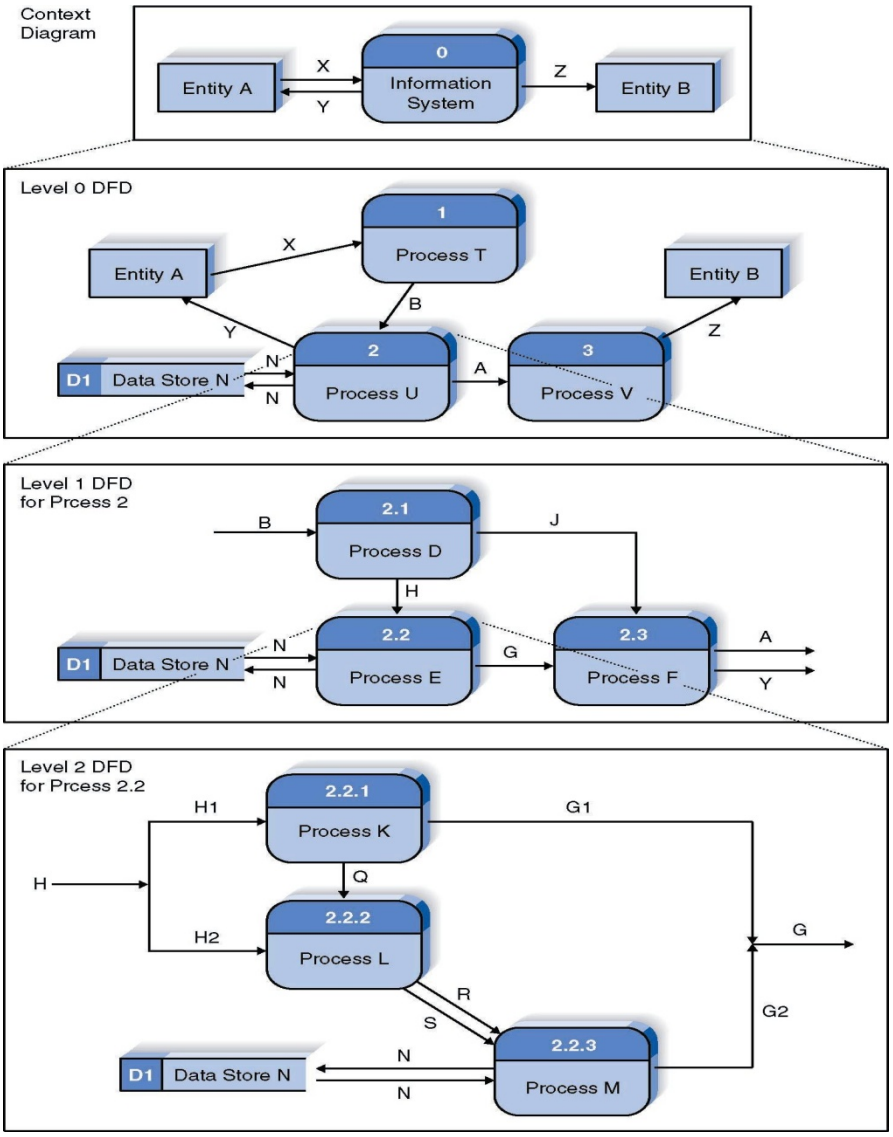
Data Store

Merupakan tempat penyimpanan data yang dapat berupa suatu file atau suatu sistem database dari suatu komputer, suatu arsip/dokumen, suatu agenda/buku.



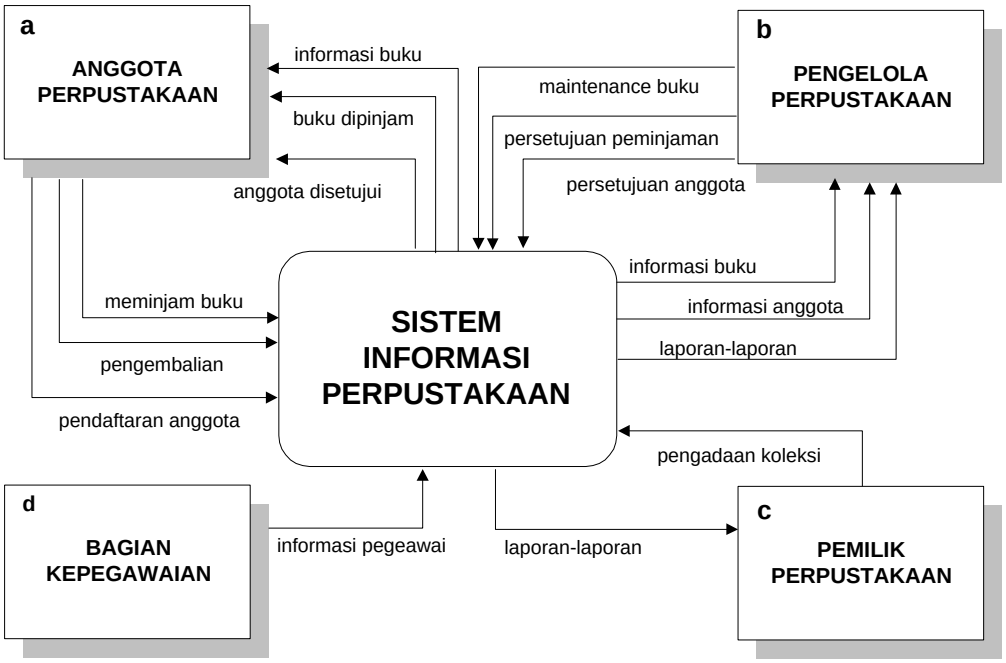
Process Decompotition

Merupakan aktifitas memecah suatu sistem menjadi komponen-komponen sub-sistem dimana Komponen-komponen sub-sistem tersebut memperlihatkan detail dari sistem diatasnya.



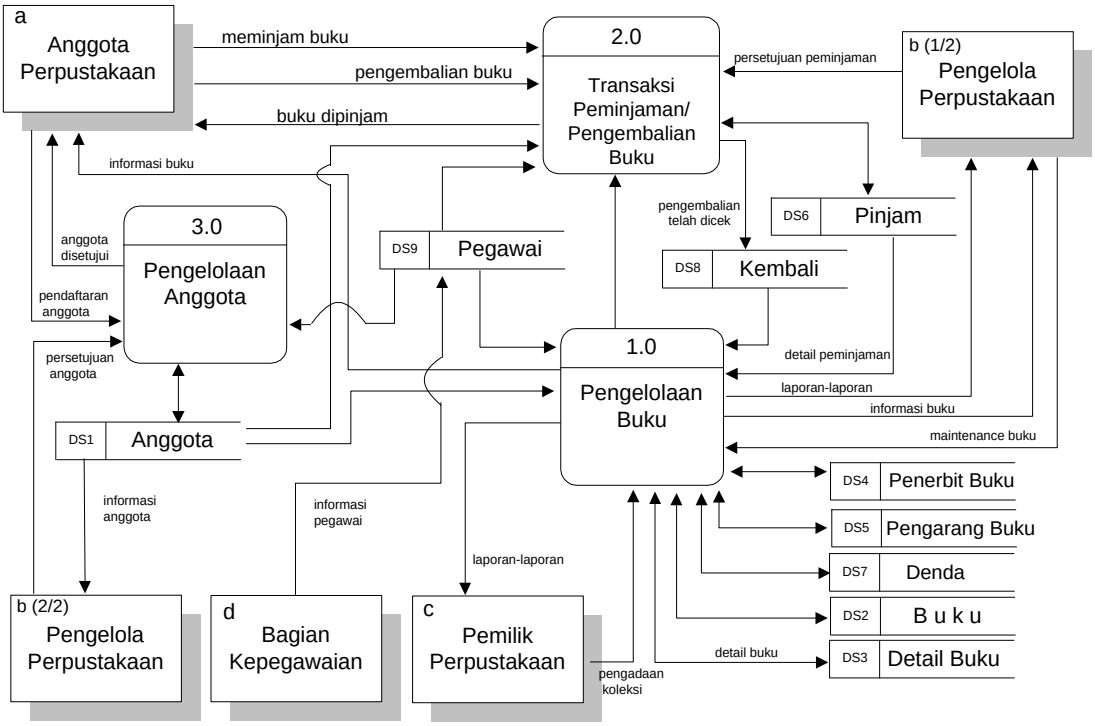
Pembuatan DFD – Context Diagram

Diagram atas atau yang paling awal dibuat dari sistem adalah Context Diagram (Diagram Konteks) yang memperlihatkan ruang lingkup (gambaran) dari sistem tersebut. Pada diagram konteks ini semua External Entity yang terlibat dengan sistem diidentifikasi. Selain itu juga mengidentifikasi aliran data baik yang masuk maupun keluar dari sistem serta arah aliran tersebut. Dalam diagram konteks hanya ada satu proses saja yaitu proses 0 dan belum memperlihatkan penyimpanan data.



Pembuatan DFD – Diagram Level 0

Diagram Level 0 (nol) merupakan penggambaran dari Context Diagram yang lebih rinci. Diagram pada level ini memperlihatkan proses-proses utama pembentuk proses 0 (komponen internal dari proses 0). Pada level ini sudah memperlihatkan Data Store yang digunakan. Hal yang paling penting yang harus diingat adalah keseimbangan aliran data antara Diagram Konteks dan Diagram Level 0 harus terus terjaga.



Pembuatan DFD – Diagram Level 1

Diagram Level 1 ini merupakan gambaran rinci dari setiap proses pada Diagram Level sebelumnya, yaitu Diagram Level 0. Pada diagram ini keseimbangan Data Store yang digunakan harus tetap dijaga. Selain itu juga sama dengan Diagram Level 0, keseimbangan aliran data antara Diagram Level 0 dan Diagram Level 1 juga harus tetap terjaga. Demikian seterusnya untuk Diagram Level selanjutnya.

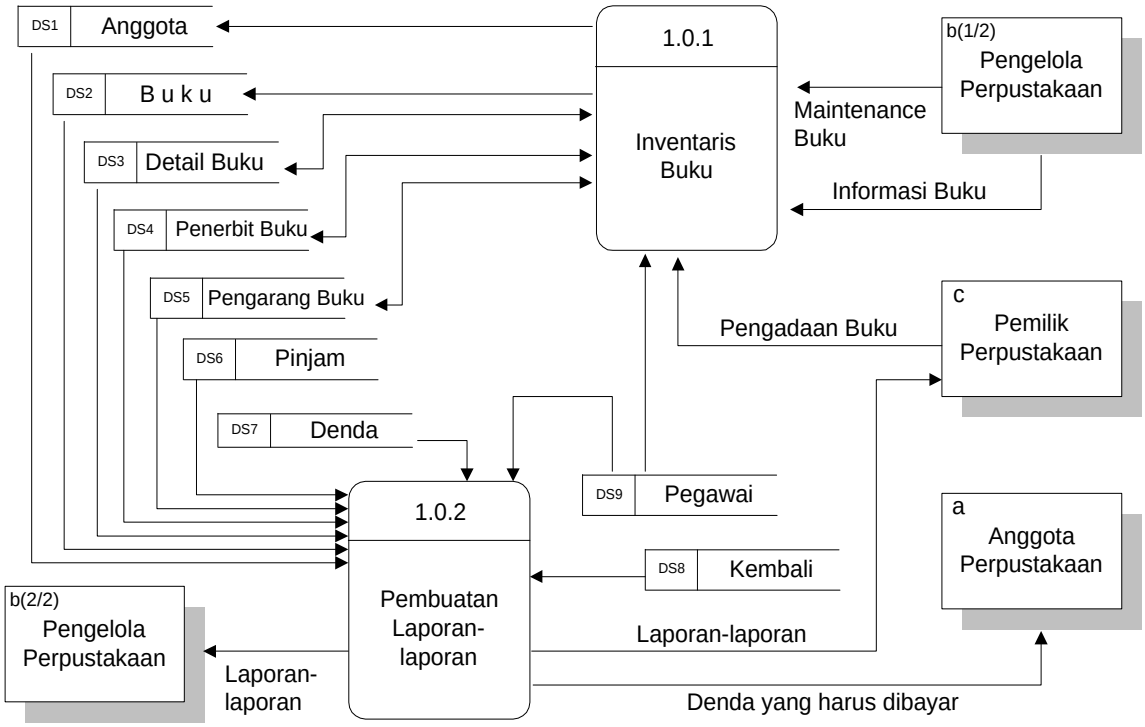


Diagram Level 1 proses 1.0 Pengelolaan Buku

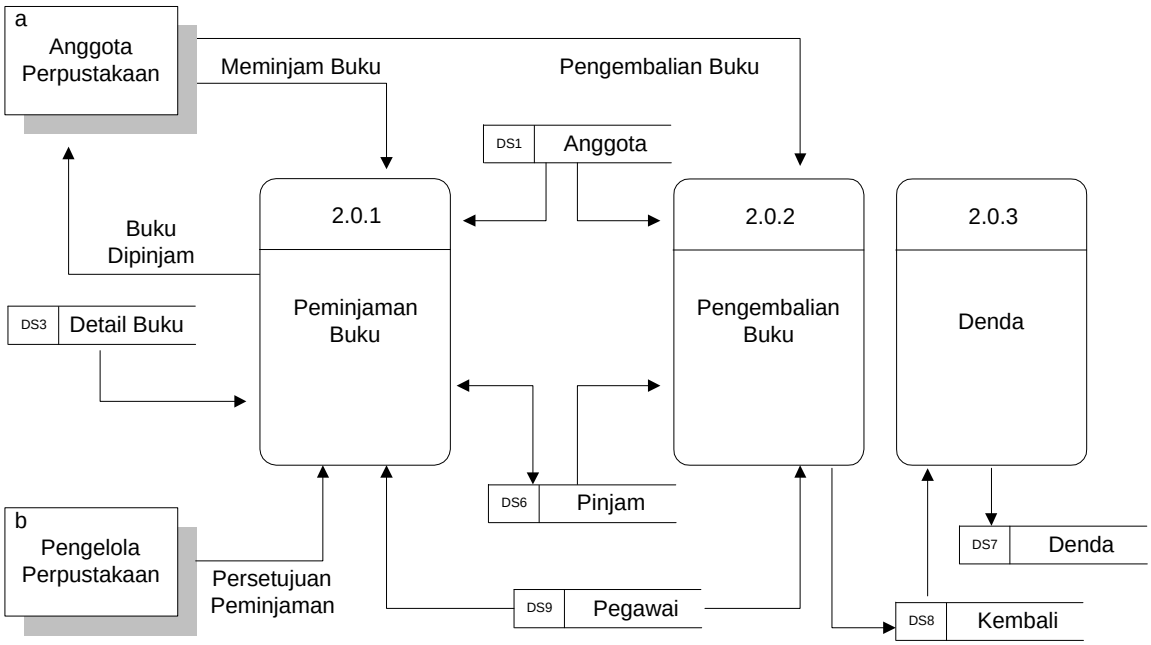


Diagram Level 1 proses 2.0 Transaksi Peminjaman / Pengembalian Buku

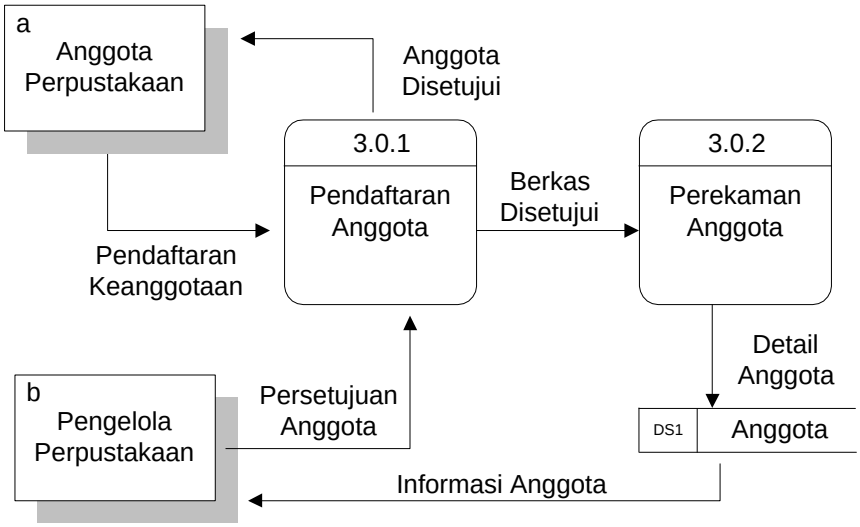


Diagram Level 1 proses 3.0 Pengelolaan Anggota

Nama Level	Nama Diagram	Nomor Proses
0	Konteks	0
1	Level Nol	1.0, 2.0, 3.0, …
2 2 2	Level 1	1.1, 1.2, 1.3, … 2.1, 2.2, 2.3, … 3.1, 3.2, 3.3, …
3 3 3	Level 2	1.1.1, 1.1.2, 1.1.3, … 1.2.1,1. 2.2, 1.2.3, … 1.3.1, 1.3.2, 1.3.3, …
dst		

Aturan Penamaan dalam DFD

Daftar Pustaka

1. Raymond McLeod, Jr., *"Management Information System A Study of Computer Based Information Systems"*, Prentice Hall, Inc, 1995.
2. Roger S. Pressman, *"Software Engineering, a Practitioner's Approach*" Fourth Edition, McGraw Hill, 1997.
3. Roger S. Pressman, *"Software Engineering, A Beginner's Guide"*, McGraw Hill, 1998.
4. Barbee Teasley Mynatt, *"Software Engineering with Student Project Guidance"*, Prentice Hall, Inc, 1990.