iskusi.4

Lakukan: Kirim balasan: 1

Jatuh tempo: Minggu, 2 November 2025, 23:59

menampilkan balasan dalam bentuk bertingkat

Setelan

Diskusi.4
Rabu, 28 Mei 2025, 10:28

Sebuah klinik bernama SehatMed ingin mengembangkan sistem manajemen klinik untuk membantu pencatatan pasien, jadwal dokter, dan transaksi pembayaran. Sistem ini akan digunakan oleh staf administrasi, dokter, dan pasien.

Sebagai analis sistem, Anda diminta untuk merancang Data Flow Diagram (DFD) untuk sistem ini. Namun, tim pengembang masih memiliki pertanyaan:

- Sampai level berapa DFD harus diturunkan agar sistem dapat dipahami dengan baik?
- Bagaimana menentukan bahwa DFD sudah cukup detail untuk melanjutkan ke tahap desain pemrograman?
- Jika DFD terlalu sederhana, sistem bisa jadi kurang jelas. Tetapi jika terlalu rinci, bisa jadi membuang waktu dan membuat analisis lebih kompleks dari yang diperlukan.

Pertanyaan Diskusi:

1. Sampai kapan analisis DFD dilakukan? Bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu diturunkan lebih lanjut?
2. Apa dampak dari DFD yang terlalu sederhana atau terlalu kompleks dalam pengembangan sistem berbasis pemrograman terstruktur?
3. Bagaimana pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD?

Selamat berdiskusi!

Catatan: Saya sarankan, teman-teman mahasiswa menjawab langsung pada tempat yang disediakan, TIDAK mengupload jawaban berupa file, termasuk TIDAK mengupload jawaban dengan GAMBAR ya. Terima kasih!

Tautan permanen Balas

Re: Diskusi.4

oleh 054416248 PANDU WIJAYA - Senin, 27 Oktober 2025, 11:26

1. Sampai kapan analisis DFD dilakukan? Bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu diturunkan lebih lanjut?

Analisis DFD dilakukan secara bertahap (berjenjang), mulai dari Diagram Konteks → Level 0 → Level 1 → Level 2, dan seterusnya hingga proses sudah cukup rinci untuk dijadikan dasar desain program.

#Kapan berhenti menurunkan level DFD?

DFD tidak perlu diturunkan lebih lanjut jika:
1. Setiap proses sudah dapat dijelaskan dalam bentuk satu prosedur atau modul program tunggal.
2. Aliran data antar proses sudah jelas: sumber, tujuan, serta bentuk datanya (misal: formulir pasien, jadwal dokter, bukti pembayaran).
3. Tidak ada proses yang masih terlalu umum seperti “Mengelola Data Pasien” tanpa dijelaskan lebih lanjut (misalnya bisa dipecah jadi Menambah Pasien, Mengedit Pasien, Menghapus Pasien).
4. Tim pengembang dan pemangku kepentingan sudah memiliki pemahaman yang sama terhadap cara kerja sistem dari DFD yang ada.

• Intinya: turunkan DFD sampai setiap proses bisa langsung diterjemahkan menjadi modul, fungsi, atau prosedur dalam pemrograman.

2. Dampak dari DFD yang terlalu sederhana atau terlalu kompleks

• Kondisi DFD terlalu sederhana
Dampak positif : Cepat dibuat dan mudah dipahami di awal
Dampak negatif :
-Detail sistem tidak tergambar dengan jelas
-Menyulitkan tahap desain dan implementasi
-Risiko salah tafsir antar pengembang tinggi

• Kondisi DFD terlalu kompleks / terlalu detail
Dampak positif : Menjelaskan sistem dengan sangat lengkap
Dampak Negatif :
-Membutuhkan waktu lama untuk analisis
-Sulit dipahami oleh non-teknis (misal: manajemen)
-Membuat dokumentasi berlebihan dan rawan inkonsistensi

• Dengan kata lain, DFD terlalu sederhana = risiko miskomunikasi, sedangkan DFD terlalu kompleks = risiko inefisiensi dan kebingungan.

3. Pendekatan terbaik untuk menjaga keseimbangan antara kejelasan dan efisiensi

Beberapa pendekatan profesional yang umum digunakan:
1. Gunakan prinsip “Top-Down Refinement”
Mulai dari level tertinggi (gambaran umum), lalu turunkan hanya bagian yang masih belum jelas atau terlalu besar.
2. Fokus pada proses bisnis utama (core processes)
Seperti pendaftaran pasien, penjadwalan dokter, dan pembayaran transaksi. Jangan langsung mendetailkan hal-hal teknis seperti validasi input — itu bisa dilakukan di tahap desain program.
3. Konsultasikan setiap level dengan pengguna atau pengembang
Setelah membuat Level 0 dan Level 1, mintalah umpan balik. Jika pengguna sudah bisa memahami sistem dengan

baik, berarti DFD cukup.
4. Gunakan batasan “satu proses = satu modul”
Jika satu proses di DFD sudah dapat diimplementasikan sebagai satu fungsi/modul di kode program, maka tidak perlu dipecah lagi.

• Kesimpulan

-Kapan berhenti menurunkan DFD? Ketika setiap proses sudah bisa langsung diimplementasikan ke modul program
-DFD terlalu sederhana memiliki dampak kurang jelas, berisiko salah tafsir
-DFD terlalu kompleks memiliki dampak membuang waktu, membingungkan
-Pendekatan terbaik gunakan top-down refinement dan libatkan pengguna tiap level

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:06

Jawaban Anda menarik karena mampu menonjolkan keseimbangan antara detail dan efisiensi.
tambahkan referensi agar jawaban lebih ilmiah,referensi utamakan dari Universitas terbuka

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh RISKI SEPTIADI MURTI WAHYU 052218499 - Senin, 27 Oktober 2025, 13:50

1. Sampai kapan analisis DFD dilakukan dan bagaimana mengetahui bahwa DFD sudah cukup detail?

Analisis DFD dilakukan sampai sistem bisa dipahami dengan jelas oleh analis, desainer, dan programmer.DFD dibuat secara bertahap, mulai dari tahap yang sederhana hingga lebih rinci. DFD dikatakan sudah cukup detail jika:
1) Setiap proses bisa dijelaskan dengan satu cara kerja atau program tunggal saat sistem dibuat.

2) Tidak ada proses lagi yang harus dibagi lagi agar bisa dipahami oleh programmer.

3) Aliran data dan tempat penyimpanan data sudah mewakili semua kebutuhan sistem secara lengkap dan tidak membingungkan.

4) Setiap entitas eksternal, proses, dan tempat penyimpanan data sudah jelas fungsi dan hubungannya satu sama lain.

Oleh karena itu, proses analisis DFD berhenti ketika semua proses di tahap terakhir bisa langsung diterjemahkan menjadi modul program.

2. Dampak DFD yang Terlalu Sederhana atau Terlalu Kompleks

Kondisi DFD dan dampaknya terhadap pengembangan sistem
Terlalu Sederhana:
1) Tidak semua informasi sistem terlihat dengan jelas.
2) Programmer kesulitan memahami proses secara mendalam.
3) Bisa menyebabkan kesalahpahaman antara analis dan pengembang.
4) Risiko kesalahan meningkat saat proses implementasi karena kebutuhan tidak jelas.

Terlalu Kompleks:
1) Memakan waktu lebih lama untuk menganalisis.
2) Mengganggu tim pengembang karena banyak detail yang tidak penting.
3) Mengurangi efisiensi dalam proses pengembangan.

4) Sulit dilakukan validasi dengan pengguna karena terlalu teknis.

Dalam konteks rekayasa perangkat lunak terstruktur, menyeimbangkan tingkat kompleksitas DFD sangat penting agar setiap modul tetap terukur dan mudah dipelihara.

3. Cara terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam DFD

Beberapa cara yang direkomendasikan adalah sebagai berikut:
1) Gunakan pendekatan top-down decomposition.
Mulailah dari diagram konteks dan hentikan proses ketika setiap langkah sudah jelas secara logis dan bisa dikembangkan menjadi bagian program.

2) Lakukan validasi di setiap tingkatan dengan pengguna.
Setiap kali DFD dibuat lebih rinci, mintalah konfirmasi dari pihak yang menggunakan sistem, seperti staf administrasi, dokter, atau pasien, untuk memastikan bahwa DFD mencerminkan kebutuhan bisnis secara tepat.

3) Fokus pada fungsi inti dan data penting.
Jangan menggambarkan hal-hal yang tidak berdampak langsung terhadap alur data utama.

4) Pertahankan konsistensi dalam penamaan dan notasi.
Pastikan bahwa simbol, nama proses, arus data, dan data penyimpanan tetap konsisten di setiap tingkatan agar mudah dipahami.

5) Terapkan aturan “satu proses satu program.”
Jika satu proses dalam DFD bisa diubah menjadi satu bagian program, maka dekomposisi tersebut sudah cukup.

Kesimpulan:
Analisis DFD dilakukan hingga mencapai tingkat yang bisa langsung diterjemahkan ke dalam desain program. Jika DFD terlalu sederhana, sistem menjadi tidak jelas, sedangkan jika terlalu rumit, proses analisis jadi tidak efisien. Pendekatan terbaik adalah melakukan dekomposisi secara bertahap dan melakukan validasi di setiap tingkatan agar tetap seimbang antara kejelasan dan efisiensi.

Sumber:
1. Universitas Terbuka. (2021). BMP STSI4202 – Rekayasa Perangkat Lunak. Modul 3: Analisis Sistem dan Pemodelan Proses.,
2. Pressman, R. S. (2010). Software Engineering: A Practitioner’s Approach. McGraw-Hill.,
3. Kendall, K. E., & Kendall, J. E. (2014). Systems Analysis and Design. Pearson Education.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:06

Penjabaran Anda tentang batas level DFD sudah logis. Semoga selalu diberi kesehatan dan semangat untuk belajar!

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh I NYOMAN ARYA YUDIHARTA 052254338 - Senin, 27 Oktober 2025, 14:22

Perancangan dan Kedalaman Analisis Data Flow Diagram (DFD) pada Sistem Klinik SehatMed
1. Batasan Kedalaman (Level) DFD
Analisis DFD dilakukan secara bertahap dengan metode dekomposisi, yaitu memecah proses besar menjadi subproses yang lebih kecil hingga sistem dapat dipahami dengan baik. Berdasarkan prinsip pada Modul 4 MSIM4302, DFD biasanya disusun mulai dari:

- Level 0 (Diagram Konteks): menggambarkan hubungan sistem dengan entitas luar.
- Level 1: menampilkan proses-proses utama yang ada di dalam sistem.
- Level 2 dan seterusnya: menggambarkan rincian proses tertentu yang masih dianggap kompleks.

Analisis DFD dapat dihentikan pada level tertentu apabila setiap proses sudah memiliki input, output, dan penyimpanan data yang jelas, serta dapat diterjemahkan ke dalam desain pemrograman tanpa kehilangan makna sistem. Dengan kata lain, jika semua alur data sudah dapat dipahami baik oleh pengembang maupun pengguna, maka dekomposisi tidak perlu dilanjutkan lagi.

2. Kriteria DFD yang Cukup Detail

Suatu DFD dinilai cukup detail apabila memenuhi kriteria berikut:

a. Setiap proses memiliki deskripsi fungsi yang lengkap dan tidak ambigu.
b. Semua data flow memiliki arah dan nama yang jelas (menggunakan kata benda).
c. Tidak terdapat proses yang “menggantung” tanpa input atau output.
d. Semua data store telah merepresentasikan kebutuhan penyimpanan dalam sistem.
e. DFD sudah dapat dijadikan acuan dalam tahap desain pemrograman.

Dalam konteks SehatMed, jika setiap proses seperti pendaftaran pasien, pengelolaan jadwal dokter, dan transaksi pembayaran sudah dapat dijelaskan secara logis dengan aliran data yang lengkap, maka DFD dapat dianggap selesai sampai level 1 atau 2 saja.

3. Dampak DFD yang Terlalu Sederhana atau Terlalu Kompleks

- DFD Terlalu Sederhana

Jika DFD dibuat terlalu ringkas, informasi penting bisa terlewat. Misalnya, dalam sistem SehatMed, hubungan antara pendaftaran pasien dan transaksi pembayaran bisa tidak terlihat jelas, menyebabkan kesalahan pada tahap implementasi karena kebutuhan data tidak tergambar secara utuh.

- DFD Terlalu Kompleks

Sebaliknya, DFD yang terlalu rinci dapat memperlambat proses analisis, membingungkan tim, dan menyulitkan dokumentasi. Diagram yang berisi terlalu banyak level atau simbol akan membuat komunikasi dengan pengguna non-teknis menjadi tidak efektif.

Oleh karena itu, kedalaman DFD harus disesuaikan dengan tujuan analisis dan kebutuhan pemahaman sistem.

4. Pendekatan Menjaga Keseimbangan Kejelasan dan Efisiensi

Pendekatan terbaik dalam merancang DFD adalah dengan metode top-down:

a. Mulai dari diagram konteks (Level 0) untuk menggambarkan hubungan antara sistem dan entitas luar.
b. Turunkan menjadi DFD Level 1 untuk menggambarkan proses utama.
c. Hanya proses yang masih kompleks yang diuraikan lebih lanjut ke Level 2 atau Level 3.
d. Hentikan dekomposisi ketika rincian sudah cukup untuk membangun modul pemrograman.

Pendekatan ini membantu menjaga keseimbangan antara kejelasan sistem (clarity) dan efisiensi analisis (efficiency).

5. Contoh Hierarki DFD Sistem Klinik SehatMed

a. DFD Level 0 (Diagram Konteks)

Entitas luar: Pasien, Dokter, Staf Administrasi.
Sistem utama: Sistem Manajemen Klinik SehatMed.
Aliran data utama:
Pasien → Data pendaftaran, pembayaran.
Dokter → Jadwal dan hasil pemeriksaan.
Staf Administrasi → Pengelolaan data dan laporan.

b. DFD Level 1

1.0 Pendaftaran Pasien → menyimpan data pasien baru.
2.0 Pengelolaan Jadwal Dokter → menyusun jadwal dan konsultasi.
3.0 Transaksi Pembayaran → mencatat dan mencetak nota pembayaran.

c. DFD Level 2 (contoh subproses dari 3.0 Transaksi Pembayaran)

3.1 Input Data Layanan dan Biaya
3.2 Verifikasi Pembayaran
3.3 Cetak Nota dan Update Data Pembayaran

Analisis DFD pada sistem SehatMed dilakukan sampai tingkat di mana setiap proses sudah menggambarkan alur data, entitas, dan penyimpanan informasi secara jelas tanpa perlu penjabaran tambahan. DFD yang terlalu sederhana berisiko menimbulkan kesalahpahaman, sedangkan yang terlalu kompleks menghambat efisiensi kerja. Oleh karena

itu, perancangan DFD sebaiknya mengikuti pendekatan bertingkat (top-down) dengan fokus pada kejelasan, konsistensi, dan keseimbangan antara kebutuhan analisis dan kepraktisan implementasi.

Sumber Referensi

Sukamto, R. A. (2021). Rekayasa perangkat lunak. Universitas Terbuka.
Jogiyanto, H. M. (2008). Analisis dan Desain Sistem Informasi: Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis. Yogyakarta: Andi Offset.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:07

Jawaban Anda menunjukkan pemahaman mendalam. Saya bangga melihat perkembangan Anda. Terus belajar dengan konsisten!

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh HAMDI 053837115 - Senin, 27 Oktober 2025, 14:42

1. Sampai kapan analisis DFD dilakukan? Bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu diturunkan lebih lanjut?

Analisis DFD dilakukan sampai setiap proses sudah dapat dijelaskan secara logis dan mudah diterjemahkan ke dalam desain program.
DFD tidak perlu diturunkan lebih lanjut jika:

Setiap proses sudah menggambarkan satu fungsi tunggal yang jelas.

Aliran data antar proses, penyimpanan data, dan entitas eksternal sudah lengkap dan tidak ambigu.

Tim pengembang dapat memahami proses tanpa perlu menanyakan detail tambahan.

Dengan kata lain, DFD dianggap cukup detail jika setiap proses di level terendah bisa langsung diterjemahkan menjadi modul program.

2. Apa dampak dari DFD yang terlalu sederhana atau terlalu kompleks dalam pengembangan sistem berbasis pemrograman terstruktur?

DFD terlalu sederhana:
Akan membuat sistem sulit dipahami secara menyeluruh karena detail penting mungkin terlewat. Akibatnya, desain program menjadi tidak akurat, dan kebutuhan pengguna bisa tidak terpenuhi.

DFD terlalu kompleks:
Membuat proses analisis menjadi lambat dan membingungkan karena terlalu banyak rincian yang tidak diperlukan. Hal ini dapat menyebabkan pemborosan waktu dan biaya dalam tahap desain serta implementasi.

Idealnya, DFD harus seimbang — cukup detail untuk menjelaskan sistem, tetapi tidak berlebihan hingga memperumit pemahaman.

3. Bagaimana pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD?

Pendekatan terbaik adalah dengan menggunakan prinsip “top-down refinement” atau perincian bertahap:

Mulai dari DFD level 0 (konteks diagram) yang menggambarkan sistem secara umum.

Turunkan ke level 1 dan seterusnya hanya untuk proses yang masih terlalu kompleks.

Hentikan perincian ketika setiap proses sudah cukup jelas untuk diimplementasikan.

Lakukan review berkala dengan pengguna dan tim pengembang agar DFD tetap relevan, tidak terlalu rumit, dan sesuai kebutuhan.

Dengan pendekatan ini, kejelasan sistem dan efisiensi analisis dapat dijaga secara optimal.

Sumber Referensi:

Kendall, Kenneth E., & Kendall, Julie E. (2019). Systems Analysis and Design (10th Edition). Pearson.

Shelly, Gary B., & Rosenblatt, Harry J. (2011). Systems Analysis and Design (9th Edition). Course Technology, Cengage Learning.

Pressman, Roger S. (2010). Software Engineering: A Practitioner's Approach (7th Edition). McGraw-Hill Education.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:08

Sangat baik! Anda sudah memahami bahwa DFD yang terlalu kompleks bisa menghambat efisiensi. Tetap semangat ya!
referensi utamakan dari Universitas terbuka

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh 054453816 ACH. RIYANTO - Senin, 27 Oktober 2025, 15:39

1. Batas akhir analisis DFD dan indikator detail yang cukup

Analisis DFD pada dasarnya berhenti ketika diagram tersebut sudah cukup jelas untuk dipahami oleh seluruh pemangku kepentingan—analis, desainer sistem, dan terutama programer. Proses ini dilakukan secara bertingkat (*top-down*), dimulai dari gambaran umum (Diagram Konteks) lalu dipecah menjadi lebih rinci.

DFD dianggap cukup detail dan tidak perlu diturunkan lebih lanjut apabila telah memenuhi kriteria berikut:

- Setiap proses pada level terendah sudah cukup spesifik untuk diimplementasikan sebagai satu fungsi atau modul program tunggal.
- Programer dapat memahami logika proses tersebut tanpa perlu memecahnya lebih jauh untuk membuat kode.
- Seluruh kebutuhan aliran data (*data flow*) dan penyimpanan data (*data store*) yang esensial telah terwakili secara lengkap dan tidak membingungkan.
- Fungsi serta relasi antar entitas eksternal (seperti Pasien, Dokter, Staf Administrasi), proses, dan data store sudah terdefinisi dengan jelas.

2. Dampak tingkat detail DFD pada pengembangan

Tingkat kerincian DFD memiliki dampak langsung pada pengembangan sistem terstruktur di Klinik SehatMed:

- DFD Terlalu Sederhana:
  - Gambaran sistem menjadi tidak utuh dan ambigu.
  - Programer akan kesulitan memahami logika bisnis secara mendalam (misalnya, bagaimana alur pendaftaran pasien terhubung ke pembayaran).

- Potensi kesalahpahaman antara analis dan tim pengembang meningkat.
- Risiko *bug* atau kegagalan implementasi lebih tinggi karena kebutuhan sistem tidak terdokumentasi dengan baik.

- DFD Terlalu Kompleks (Terlalu Rinci):
  - Proses analisis memakan waktu yang tidak efisien karena terlalu banyak detail minor yang digambarkan.
  - Tim pengembang terdistraksi oleh detail-detail kecil yang mungkin tidak relevan untuk level pemrograman (misalnya, *error handling* spesifik).
  - Proses validasi dengan *end-user* (Staf Administrasi atau Dokter) menjadi sulit karena diagram terlalu teknis dan rumit bagi mereka.
  - Mengurangi efisiensi pengembangan secara keseluruhan.

Dalam pemrograman terstruktur, keseimbangan detail DFD sangat penting untuk memastikan modul-modul program tetap terkelola, terukur, dan mudah dipelihara.

3. Pendekatan terbaik untuk keseimbangan DFD

Untuk menjaga keseimbangan antara kejelasan (clarity) dan efisiensi (efficiency) dalam perancangan DFD, berikut adalah pendekatan yang direkomendasikan:

- Gunakan dekomposisi top-down (Bertahap)
  Mulailah dari Diagram Konteks (Level 0) yang menunjukkan sistem secara keseluruhan, lalu turunkan ke Level 1 (proses utama), Level 2 (sub-proses), dan seterusnya. Hentikan pemecahan proses ketika setiap proses sudah cukup logis untuk dikembangkan menjadi satu unit program.
- Validasi bertahap dengan pengguna
  Setiap kali DFD diturunkan (misalnya dari Level 1 ke Level 2), lakukan konfirmasi dengan pemangku kepentingan di SehatMed (Staf, Dokter, Pasien jika perlu) untuk memastikan alur kerja bisnis tercermin secara akurat.
- Fokus pada fungsi inti dan data penting
  Prioritaskan penggambaran alur data dan proses utama (pendaftaran, jadwal, rekam medis, pembayaran). Hindari menggambarkan proses-proses trivial atau detail teknis implementasi yang bukan merupakan alur bisnis utama.
- Jaga konsistensi notasi
  Gunakan penamaan yang standar dan jelas untuk proses, aliran data, dan data store di semua level DFD agar mudah dilacak dan dipahami oleh seluruh tim.
- Terapkan prinsip "Satu Proses, Satu Modul": Jadikan ini sebagai aturan praktis. Jika sebuah proses dalam DFD (misalnya "Proses Hitung Biaya Konsultasi") sudah dapat diimplementasikan sebagai satu fungsi atau modul program, dekomposisi untuk proses tersebut sudah cukup.

Referensi:

Universitas Terbuka. (2021). BMP STSI4202 – Rekayasa Perangkat Lunak

Modul materi inisiasi 4

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:09

Sangat baik! Anda sudah memahami bahwa DFD yang terlalu kompleks bisa menghambat efisiensi. Tetap semangat ya!

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh VEYSTI DICNORYA BERNANTI 052338378 - Senin, 27 Oktober 2025, 15:52

Selamat Siang
Izin menjawab diskusi

1. Analisis DFD dilakukan hingga setiap proses dalam sistem dapat dijelaskan secara jelas dan logis, tanpa menimbulkan ambiguitas bagi tahap desain dan implementasi.
Secara umum, analisis DFD berhenti ketika:
- Setiap proses dapat dijelaskan dalam satu kalimat yang jelas dan tidak ambigu.
Artinya, setiap proses sudah memiliki input, output, dan transformasi data yang terdefinisi dengan baik.
- Setiap data flow dan data store sudah teridentifikasi secara lengkap.
Semua aliran data utama dalam sistem (misalnya data pasien, jadwal dokter, transaksi pembayaran) sudah digambarkan.
- Tidak ada proses yang terlalu kompleks.
Jika sebuah proses tidak bisa dijelaskan dengan satu kalimat atau terlalu luas, berarti masih perlu diturunkan ke level yang lebih rendah.
- Pengembang dan pengguna memahami sistem yang sama.
Analisis berhenti ketika pengguna non-teknis (staf klinik, dokter, dll.) dapat memahami bagaimana data mengalir dalam sistem.

2. Apabila DFD terlalu sederhana makan dampaknya adalah Tidak semua proses dan aliran data tergambar dengan jelas, Pengembang kebingungan menentukan logika program, Sistem yang dihasilkan tidak sesuai kebutuhan pengguna
Misalnya hanya ada satu proses “Manajemen Klinik” tanpa rincian cara input data pasien, dokter, dan pembayaran.
Sedangkan jika DFD terlalu kompleks dampaknya membutuhkan waktu lama untuk analisis, Sulit dipahami oleh tim non-teknis dan Berpotensi redundansi proses atau aliran data.
Misalnya setiap langkah kecil (seperti “klik tombol simpan”) dijadikan satu proses dalam DFD.

3. Agar DFD efisien namun tetap jelas, berikut pendekatan yang disarankan :
- Gunakan Prinsip “Top-Down Decomposition” Secara Bertahap.
Mulailah dari gambaran besar (context diagram), lalu turunkan hanya bagian yang masih kurang jelas, Jangan langsung membuat DFD Level 3 tanpa memastikan Level 1 dan 2 stabil.
- Validasi Tiap Level Bersama Pengguna (User Validation)
Setelah setiap level selesai, lakukan sesi verifikasi dengan staf klinik atau dokter untuk memastikan alur data benar dan mudah dipahami.
- Batasi Jumlah Proses di Setiap Level (5–7 Proses per Diagram)
Menurut Yourdon & DeMarco (1980), manusia sulit memahami lebih dari 7 elemen sekaligus.
Jika terlalu banyak proses, pecah ke diagram baru agar tetap mudah dibaca.
- Gunakan Konsistensi Antar Level
Pastikan nama aliran data dan data store tetap sama antar level agar tidak membingungkan.
- Gunakan Alat Dokumentasi Terstandar
Misalnya dengan notasi Gane & Sarson atau Yourdon, dan simpan setiap versi diagram agar perubahan bisa dilacak.

Sumber Referensi :
Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner’s Approach (9th ed.). McGraw-Hill Education.
BMP Rekayasa Perangkat Lunak Modul 4 Universitas Terbuka 2025

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:21

Penjelasan Anda runtut dan mudah dipahami. Terus tingkatkan kemampuan analisis sistemnya

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh 054518659 DHAFFA NIZHAR ADINDHA PUTRA - Senin, 27 Oktober 2025, 20:54

1. Selama tahap analisis sistem dan awal tahap perancangan, analisis Data Flow Diagram (DFD) digunakan untuk memetakan aliran data dari input hingga output melalui berbagai proses yang terjadi di dalam sistem. Analisis DFD dilakukan secara bertahap, dimulai dengan diagram konteks (level 0) yang menggambarkan sistem secara keseluruhan, kemudian diturunkan menjadi diagram DFD level 1, level 2, dan seterusnya hingga setiap proses di dalam sistem terdefinisi dengan menggunakan diagram DFD level 1.
Apabila setiap proses sudah menggambarkan aktivitas yang spesifik dan dapat diimplementasikan langsung dalam bentuk modul program, analisis DFD tidak perlu diturunkan lebih jauh. Analisis DFD juga dianggap cukup atau selesai ketika setiap proses memiliki input, output, dan hubungan dengan gudang data atau entitas eksternal yang sesuai. Selain itu, DFD dianggap cukup detail dan siap digunakan untuk tahap perancangan sistem berikutnya jika seluruh aliran data mudah dipahami oleh perancang, analis, dan pengguna.

2. Dampak DFD yang terlalu sederhana
-Jika konteks atau proses penting hilang atau tidak dimodelkan, ada kemungkinan bahwa interaksi entitas luar atau arus data penting akan terlupakan, yang dapat menyebabkan kekurangan dalam spesifikasi sistem.
-Programer mungkin menghadapi tugas besar yang tidak terstruktur, tidak efisien, atau salah arti karena modul pemrograman mungkin tidak memiliki panduan yang cukup.
-Stakeholder yang tidak teknis mungkin memahami gambaran yang luas, tetapi teknisi tidak memiliki detail desain.

Dampak DFD yang terlalu kompleks, yang berarti terlalu banyak level dan detail
-Bisa menjadi sulit untuk dibaca, dipahami, dan dikelola, terutama bagi pengguna, stakeholder bisnis, dan tim pengembangan yang juga membutuhkan penjelasan yang mudah dipahami.
-Proses desain dan implementasi DFD dapat terhambat oleh jumlah waktu yang dihabiskan untuk pembuatan dan pemeliharaan DFD.
-Risiko "over-engineering" adalah ketika proses yang sebenarnya sederhana dipecah terlalu banyak. Ini dapat menyebabkan modularisasi yang berlebihan, kompleksitas yang tinggi, dan mungkin mengakibatkan pemborosan waktu atau kode yang sulit dipelihara.
-Jika diagram terlalu rinci dan memiliki banyak level, koordinasi tim menjadi lebih sulit. Perubahan kecil memerlukan update banyak diagram, yang meningkatkan risiko inkonsistensi.
Jadi, keseimbangan diperlukan agar DFD cukup detail untuk mendukung implementasi tetapi tidak menjadi beban pengembangan.

3. Untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi perancangan DFD, metode terbaik adalah dengan membuat diagram secara bertahap dan proporsional sesuai dengan kebutuhan analisis. Proses ini dimulai dengan diagram DFD level tinggi atau konteks yang menggambarkan sistem secara keseluruhan, dan kemudian dekomposisi dilakukan hanya pada bagian yang terlalu luas atau tidak jelas. Setiap proses harus dijelaskan dengan detail yang cukup untuk membantu perancangan modul program, tanpa menambah detail yang tidak berguna. Selain itu, setiap diagram harus mudah dibaca dan dipahami dengan menandai proses, aliran data, dan penyimpanan data.Untuk memastikan bahwa DFD yang dibuat sudah sesuai dengan kebutuhan sistem dan tingkat detailnya tidak berlebihan, kolaborasi antara perancang, analis, dan pengguna juga penting. Dengan demikian, kejelasan sistem dapat terjaga sementara efisiensi waktu dan usaha perancangan tetap terjaga.

Sumber referensi: Modul 4 MSIM4303 Rekaysa perangkat lunak

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:22

Jawaban yang sangat reflektif. Anda sudah memahami konsep “cukup detail” dalam DFD dengan baik.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh ULIKTA MUTAWAFINA 051517222 - Senin, 27 Oktober 2025, 21:31

1. Sampai kapan analisis DFD dilakukan? Bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu diturunkan lebih lanjut?

Analisis DFD dilakukan sampai setiap proses dalam sistem telah diuraikan secara logis dan dapat dipahami dengan jelas oleh pengembang maupun pengguna. Berdasarkan Modul STSI4301 Sistem Penunjang Keputusan (Universitas Terbuka, 2022), penurunan DFD dimulai dari diagram konteks, DFD level 0, level 1, hingga level 2 atau lebih, tergantung pada kompleksitas sistem.

Proses analisis dapat dihentikan ketika setiap proses telah mencapai tingkat elementer, yaitu proses terkecil yang sudah dapat langsung diterjemahkan menjadi modul program atau prosedur yang jelas. Pada tahap ini, setiap aliran data memiliki sumber dan tujuan yang pasti, serta tidak ada proses yang ambigu atau belum terdefinisi. Dengan kata lain, DFD sudah cukup detail ketika setiap fungsi sistem dapat diimplementasikan tanpa menimbulkan interpretasi ganda bagi pengembang.

2. Apa dampak dari DFD yang terlalu sederhana atau terlalu kompleks dalam pengembangan sistem berbasis pemrograman terstruktur?

DFD yang terlalu sederhana akan menyebabkan pemahaman sistem menjadi kurang jelas. Hubungan antarproses dan aliran data tidak tergambarkan secara rinci, sehingga pengembang kesulitan menerjemahkannya ke dalam program. Akibatnya, sistem yang dibangun bisa menyimpang dari kebutuhan sebenarnya.

Sebaliknya, DFD yang terlalu kompleks dapat memperlambat proses analisis karena berisi terlalu banyak detail yang tidak relevan. Dokumentasi menjadi berlebihan, sulit dipahami, dan membingungkan tim pengembang. Selain itu, waktu dan biaya analisis akan meningkat karena beban pekerjaan yang tidak efisien.

Menurut Kendall & Kendall (2016), keseimbangan antara kompleksitas dan kejelasan sangat penting agar DFD berfungsi optimal sebagai alat komunikasi dan perancangan sistem.

3. Bagaimana pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD?

Pendekatan terbaik adalah dengan menerapkan prinsip dekomposisi bertahap (top-down analysis). Proses penjabaran dilakukan dari umum ke khusus, dan hanya pada bagian sistem yang kompleks atau memerlukan rincian tambahan. Proses yang sudah jelas tidak perlu diturunkan lagi.

Selain itu, dilakukan validasi dan verifikasi DFD secara berkala bersama pengguna (user) untuk memastikan bahwa setiap proses sudah sesuai kebutuhan sistem nyata. Penggunaan data dictionary dan model pendukung lainnya dapat membantu memperjelas aliran data tanpa harus menambah level DFD yang berlebihan.

Dengan pendekatan tersebut, DFD akan tetap efisien, mudah dipahami, dan mampu menggambarkan sistem secara menyeluruh tanpa kehilangan detail penting untuk tahap perancangan pemrograman.

Sumber Referensi:
- Modul STSI4301 – Sistem Penunjang Keputusan. Tangerang Selatan: Universitas Terbuka.
- Kendall, K. E., & Kendall, J. E. (2016). Systems Analysis and Design (9th ed.). Pearson Education.
- Pressman, R. S. (2015). Software Engineering: A Practitioner's Approach. McGraw-Hill.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:22

Bagus sekali, Anda menunjukkan pemahaman konseptual dan aplikatif dalam menjelaskan DFD.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh 052471106 MUHAMAD FUJI SUBEKTI - Selasa, 28 Oktober 2025, 00:41

Nama: Muhamad Fuji Subekti
NIM: 052471106
Ijin menanggapi

1. Analaisis DFD sebaiknya dilakukan sampai semua proses utama dan interaksi antar aktor dalam sistem dapat dipahami dengan jelas. DFD biasanya diturunkan hingga mencapai level 2 atau level 3, tergantung pada kompleksitas sistem. DFD dianggap cukup detail jika:
- Semua aliran data diidentifikasi dengan baik.
- Proses-proses utama dapat dipecah menjadi sub-proses yang jelas.
- Interaksi antara aktor dan sistem dapat dipahami tanpa ambiguitas.
Jika penambahan detail tidak memberikan nilai tambah atau memperjelas sistem, maka DFD sudah cukup. Jumplah proses dalam satu DFD sebaiknya antara 3 hingga 9, tidak termasuk diagram konteks yang hanya memiliki satu proses.

2. a. Terlalu Sederhana:
- Dapat mengakitabkan kehilangan informasi penting yang dibutuhkan untuk pengembangan.
- Menyebabkan kebingungan di kalangan pengembang dan pemangku kepentingan, karena proses dan aliran data yang tidak jelas.
b. Terlalu Kompleks:
- Membuat pemahaman sistem menjadi sulit, bahkan bagi pengembang yang terlibat.
- Meningkatkan waktu dan usaha yang diperlukan untuk mengembangkan sistem, mengakibatkan inefisiensi.
- Dapat menyebabkan kesalahan dalam implementasi karena berbagai detail yang berlebihan.

3. Pendekatan terbaik meliputi:
- Iterasi: Gunakan pendekatan iteratif untuk merancakan dan mendiskusikan DFD dengan tim dan pemangku kepentingan. Umpan balik dari mereka dapat membantu menentukan tingkat detail yang tepat.
- Fokus pada Proses Utama: Identifikasi dan fokus pada proses-proses yang paling kritis bagi sistem. Jangan terjebak dalam detail-detail kecil yang tidak berpengaruh signifikan.
- Definisikan Tujuan yang Jelas: PAstikan setial level DFD memiliki tujuan yang jelas dan memberikan informasi yang relevan untuk pengembangan.
- Gunakan Standar: Adopsi standar yang jelas dalam pembuatan DFD, seperti notasi yang konsisten, untuk membantu memahami diagram secara keseluruhan.
- Balancing: PAstikan konsistensi antara level DFD yang berbeda, dimana input dan putput pada level yang lebih tinggi harus sesuai dengan level yang lebih rendah.

Dengan pendekatan ini, DFD dapat menjadi alat yang efektif untuk merancang sistem yang jelas dan efisien.
Referensi Tambahan:
- DFD Lels: DFD memiliki beberapa level, dimulai dari Diagram Konteks (Level 0) yang memberikan gambaran umum sistem hingga Level 1, LEvel 2, dan seterusnya yang memberikan detail lebih lanjut. Semakin rendah levelnya, semakin detail diagramnya.
- Komponen DFD: DFD terdiri dari entitas eksternal, proses, penyimpanan data, dan aliran data.
- Tujuan DFD: DFD digunakan untuk menganalisis sitem, mengidentifikasi masalah, meningkatkan proses, dan mendorong kolaborasi.
- Tools untuk Membuat DFD: Terdapat berbagai tools yang dapat digunakan untuk membuat DFD, seperti Microsoft Visio, Lucidchart, dan Lainnya.

Sumber Referensi.
1. What is a Data Flow Diagram - Lucidchart
2. What Is a Data Flow Diagram (DFD)? - IBM
3. What is DFD(Data Flow Diagram)? - GeeksforGeeks
4. Data-flow diagram - Wikipedia

5. Dataflow Analysis with DFD Overlays Minimizes Missing and Misunderstood Requirements
6. What Is a DFD? Data Flow Diagrams Explained - Atlassian
7. DFD Guide: Step by Step Approach For Business Analysts - Adaptive US
8. Mastering Data Flow Diagram Balancing: A Comprehensive Guide
9. A Comprehensive Guide to Data Flow Diagrams (DFDs) | by JIN | Tech x Humanity | Medium
10. Data Flow Diagram - Balancing - Tutorials Point
11. Levels in Data Flow Diagrams (DFD) - GeeksforGeeks
12. Microsoft Visio - Download

Tautan permanen Tampilkan induk Balas

Hide sidebars
Course dashboard

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:25

jawaban sudah bagus, tapi penulisan sumber referensi belum tepat , pelajari cara penulisan referensi , referensi utamakan dari Universitas terbuka

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh SUPRIYADI 050127551 - Selasa, 28 Oktober 2025, 09:03

Ijin Menjawab Diskusi 4
1. Sampai kapan analisis DFD dilakukan? Bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu diturunkan lebih lanjut?
Analisis DFD dilakukan hingga tingkat detail di mana setiap proses sudah dapat dijelaskan dalam bentuk logika program atau modul terpisah, tanpa kehilangan makna fungsional dari sistem.
Kriteria bahwa DFD sudah cukup detail:
1. Setiap proses pada DFD level terakhir hanya melibatkan satu jenis transformasi data. Artinya, proses tersebut dapat langsung diterjemahkan menjadi modul program atau fungsi dalam kode.
2. Tidak ada proses yang masih terlalu umum atau ambigu. Misalnya, proses “Kelola Data Pasien” harus diturunkan menjadi “Tambah Data Pasien”, “Ubah Data Pasien”, dan “Hapus Data Pasien”.
3. Semua aliran data sudah jelas sumber dan tujuannya. Setiap data flow memiliki asal (entitas/proses) dan tujuan yang terdefinisi.
4. Tidak ada proses yang perlu dipecah lagi untuk memahami logika kerjanya.
Dengan kata lain, analisis DFD berhenti saat setiap proses sudah bisa diterjemahkan langsung ke desain pemrograman (pseudocode atau struktur modul).

2. Dampak dari DFD yang terlalu sederhana atau terlalu kompleks dalam pengembangan sistem berbasis pemrograman terstruktur.
Jika DFD terlalu sederhana:
• Kehilangan detail penting, sehingga kebutuhan sistem tidak sepenuhnya terwakili.
• Menyulitkan pengembang karena logika program tidak jelas.
• Meningkatkan risiko kesalahan implementasi, karena deskripsi proses terlalu umum.
Contoh: Jika hanya ada proses “Kelola Pembayaran”, pengembang tidak tahu apakah sistem mendukung “Input Transaksi”, “Cetak Kwitansi”, atau “Rekap Laporan Keuangan”.
Jika DFD terlalu kompleks:
• Menghabiskan waktu dan tenaga analisis untuk detail yang tidak relevan.
• Sulit dipahami oleh stakeholder non-teknis (misalnya dokter atau staf administrasi).
• Meningkatkan risiko inkonsistensi antar-level DFD, karena terlalu banyak dekomposisi.
Kesimpulan: Baik kesederhanaan maupun kompleksitas berlebihan dapat mengganggu efektivitas komunikasi dan akurasi desain sistem.

3. Pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD.
Pendekatan Terbaik:
1. Gunakan prinsip top-down decomposition (dekomposisi bertahap). Mulai dari Context Diagram (gambaran umum),

kemudian turunkan menjadi Level 0, Level 1, dan seterusnya hingga tiap proses cukup jelas.
2. Gunakan aturan “one function per process”. Setiap proses hanya mewakili satu fungsi utama yang logis dan terukur.
3. Libatkan stakeholder (user & developer) secara aktif. Pastikan setiap level DFD diverifikasi bersama pengguna agar sesuai kebutuhan riil.
4. Gunakan dokumentasi pendukung seperti Kamus Data (Data Dictionary). Ini membantu menghindari kebutuhan untuk terlalu banyak level DFD karena definisi data sudah tertulis jelas.
5. Berhenti di level di mana DFD sudah bisa diturunkan ke rancangan modul atau pseudocode.
Intinya: Keseimbangan tercapai ketika DFD:
• Cukup rinci untuk pengembang, dan
• Cukup ringkas untuk dipahami oleh pemakai non-teknis.

Referensi:
• Kendall, K. E., & Kendall, J. E. (2014). Systems Analysis and Design (9th ed.). Pearson Education.
• Shelly, G. B., & Rosenblatt, H. J. (2012). Systems Analysis and Design. Cengage Learning.
• Whitten, J. L., & Bentley, L. D. (2007). Systems Analysis and Design Methods (7th ed.). McGraw-Hill.
• Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2012). Systems Analysis and Design in a Changing World. Cengage Learning.
• Dennis, A., Wixom, B. H., & Tegarden, D. (2015). Systems Analysis and Design: An Object-Oriented Approach with UML. Wiley.
• Pressman, R. S. (2010). Software Engineering: A Practitioner’s Approach (7th ed.). McGraw-Hill.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:26

Jawaban Anda padat dan mencerminkan pemahaman yang matang terhadap DFD., referensi utamakan dari Universitas terbuka

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh 054276582 MUHAMMAD RIFKI ARSAH - Selasa, 28 Oktober 2025, 19:10

1.-DfD dilakukan Hingga Tingkat Detail yang Spesifik: Analisis DFD berlanjut dari diagram konteks (level 0) ke level-level yang lebih rendah (level 1, 2, dst.) melalui proses dekomposisi, di mana setiap proses dipecah menjadi sub-proses yang lebih detail.

-Menentukan bahwa DFD sudah cukup detail yaitu dengan kriteria sebagai berikut:
- Proses Atomik: Setiap proses pada level terakhir cukup sederhana dan fokus pada satu fungsi tertentu.
- Tidak Ada Nilai Tambah pada Dekomposisi Lanjut: Penurunan detail lebih lanjut tidak memberikan pemahaman baru yang signifikan bagi pengembang atau pemangku kepentingan.
- Detail Terlalu Rumit: DFD menjadi terlalu rumit dengan terlalu banyak proses atau aliran data, sehingga sulit dibaca dan dipahami.
- Kesiapan untuk Tahap Berikutnya: DFD sudah cukup detail untuk digunakan sebagai dasar desain detail pada tahap selanjutnya dalam pengembangan sistem.

2.-DFD Terlalu Sederhana:
- Kehilangan Detail Penting: Tidak menggambarkan aliran data dan proses secara memadai, sehingga detail fungsional penting terlewatkan.
- Kesalahan Analisis: Dapat menyebabkan kesalahpahaman dalam analisis kebutuhan dan desain awal.
- Implementasi yang Salah: Berisiko menghasilkan sistem yang tidak sesuai dengan kebutuhan nyata.

-DFD Terlalu Kompleks:
- Sulit Dipahami: Diagram menjadi "berantakan" dan sulit dibaca, mengurangi kejelasan.
- Penurunan Efisiensi: Membutuhkan lebih banyak waktu untuk menganalisis dan memahami diagram, memperlambat

proses desain dan implementasi.
- Pemborosan Sumber Daya: Menyebabkan pengembangan yang berlebihan atau fitur yang tidak perlu karena kompleksitas yang tidak dibutuhkan.

3.pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD yaitu:
- Dekomposisi Bertahap: Mulai dari DFD level tinggi (konteks) dan pecah setiap proses menjadi level yang lebih rendah secara bertahap, hanya jika perlu, hingga mencapai tingkat detail yang fungsional.
- Gunakan Aturan Penomoran yang Konsisten: Pastikan setiap proses dan alur data dinomori secara sistematis untuk memudahkan pelacakan antar level.
- Validasi dengan Pemangku Kepentingan: Lakukan tinjauan rutin dengan pengguna dan analis untuk memastikan DFD akurat, jelas, dan relevan.
- Fokus pada Informasi Penting: Pastikan DFD menyoroti input, output, proses, dan penyimpanan data secara jelas, tanpa detail implementasi teknis yang berlebihan.
- Gunakan Alat Bantu Pemodelan yang Tepat: Manfaatkan perangkat lunak pemodelan DFD untuk memudahkan pembuatan, modifikasi, dan visualisasi diagram secara efisien.

Sumber:
https://www.sekawanmedia.co.id/blog/dfd-adalah/

https://www.collegesidekick.com/study-docs/14448571

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:27

> Anda sudah berada di jalur yang benar dalam memahami tahapan analisis sistem. referensi utamakan dari Universitas terbuka

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh RIANDI SOLIHIN 051676705 - Selasa, 28 Oktober 2025, 21:25

Izin menjawab diskusi tutor

Analisis DFD dilakukan sampai tahap di mana setiap proses dalam sistem sudah tergambarkan dengan jelas dan tidak menimbulkan kebingungan bagi pengembang. Dalam modul Rekayasa Perangkat Lunak (STSI4202) dijelaskan bahwa DFD diturunkan dari diagram konteks (Level 0) ke Level 1, Level 2, dan seterusnya sampai seluruh proses kompleks dapat diuraikan menjadi proses yang lebih sederhana dan mudah dipahami. Analisis dianggap cukup jika setiap aliran data, penyimpanan data, dan entitas eksternal sudah terdefinisi dengan jelas dan siap diterjemahkan ke dalam desain pemrograman.
DFD yang terlalu sederhana bisa menyebabkan sistem menjadi kurang jelas, kebutuhan pengguna tidak tergambar dengan baik, dan programmer berpotensi salah dalam membuat logika program. Sebaliknya, DFD yang terlalu kompleks akan menyulitkan tim dalam membaca, memperpanjang waktu analisis, serta membuat dokumentasi menjadi tidak efisien. Hal ini bisa menimbulkan pemborosan waktu dan sumber daya dalam tahap pengembangan perangkat lunak.
Untuk menjaga keseimbangan, pendekatan terbaik adalah dengan menggunakan metode dekomposisi bertahap (top-down decomposition) — memecah proses besar menjadi subproses hanya sampai tingkat yang benar-benar dibutuhkan untuk pemahaman sistem. Selain itu, melibatkan pengguna (user) dalam setiap tahap analisis sangat penting agar DFD yang dihasilkan sesuai kebutuhan nyata dan tetap efisien dalam pengembangan sistem berbasis pemrograman terstruktur.

Referensi:

Universitas Terbuka. (2020). Rekayasa Perangkat Lunak (STSI4202). Tangerang Selatan: Universitas Terbuka.
Pressman, R. S. (2010). Software Engineering: A Practitioner's Approach. McGraw-Hill.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:28

Anda sudah memahami esensi dari batas detail DFD, ini pencapaian yang baik!

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh WENDI BUDIARSO 053647749 - Selasa, 28 Oktober 2025, 21:58

Nama : Wendi Budiarso
NIM : 053647749

1. Analisis DFD harus dilakukan sampai semua kebutuhan sistem telah diidentifikasi dan dipetakan dengan baik. DFD yang terlalu sederhana dapat menyebabkan hilangnya informasi penting, sedangkan DFD yang terlalu rinci dapat mempersulit pemahaman dan implementasi. Oleh karena itu, gunakan pendekatan bertahap, dan selalu libatkan pemangku kepentingan dalam proses untuk mendapatkan perspektif yang lebih luas.
Untuk menentukan bahwa Data Flow Diagram (DFD) sudah cukup detail dan tidak perlu diturunkan lebih lanjut dapat dilakukan dengan mempertimbangkan beberapa kriteria berikut:

a. Pastikan semua proses yang relevan telah diidentifikasi dan digambarkan. Setiap fungsi utama dalam sistem harus memiliki representasi dalam DFD.

b. Periksa apakah aliran data antara entitas, proses, dan penyimpanan data jelas dan mudah dipahami. Jika aliran data sudah terdefinisi dengan baik, DFD mungkin sudah cukup detail.

c. Dapatkan umpan balik dari pemangku kepentingan. Jika mereka merasa bahwa DFD sudah mencakup semua aspek yang penting dan tidak ada tambahan yang diperlukan, itu adalah indikator yang baik bahwa DFD sudah memadai.

d. Pertimbangkan kompleksitas sistem. Jika DFD sudah menggambarkan interaksi yang kompleks tanpa menambah kebingungan, maka detil lebih lanjut mungkin tidak diperlukan.

e. Sesuaikan dengan tujuan pembuatan DFD. Jika DFD digunakan untuk analisis awal, maka detail yang lebih rendah mungkin cukup. Namun, jika digunakan untuk pengembangan lebih lanjut, detail yang lebih tinggi mungkin diperlukan.

f. Ikuti standar dan praktik terbaik dalam pembuatan DFD. Jika DFD memenuhi standar tersebut dan tidak ada kebutuhan tambahan, maka sudah cukup detail.

2. Dampak dari Data Flow Diagram (DFD)

DFD Terlalu Sederhana :

- DFD yang terlalu sederhana mungkin tidak mencakup semua proses dan aliran data yang penting, sehingga mengakibatkan pemahaman yang tidak lengkap tentang sistem.
- Proses yang tidak teridentifikasi dapat menyebabkan kesalahan dalam pengembangan, karena pengembang mungkin tidak menyadari semua fungsi yang perlu diimplementasikan.
- Dengan kurangnya detail, akan sulit untuk merancang kasus pengujian yang komprehensif, sehingga mengurangi kualitas sistem.
- Pemangku kepentingan mungkin merasa bahwa sistem tidak memenuhi kebutuhan mereka jika DFD tidak

menggambarkan semua aspek yang penting.

DFD Terlalu Kompleks :

- DFD yang terlalu kompleks dapat membingungkan pengembang dan pemangku kepentingan, membuat sulit untuk memahami aliran data dan proses yang terjadi.
- Kompleksitas yang tinggi dapat memperpanjang waktu pengembangan dan meningkatkan biaya, karena lebih banyak waktu diperlukan untuk menganalisis dan memahami sistem.
- Pengembang mungkin berusaha menangani setiap detail, yang dapat menyebabkan over-engineering dan membuat sistem menjadi lebih rumit dari yang diperlukan.
- Sistem yang kompleks lebih sulit untuk dipelihara dan diubah di masa depan, karena setiap perubahan kecil dapat mempengaruhi banyak bagian dari sistem.

3. Pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD:

- Tentukan tujuan spesifik DFD (analisis, perancangan, dokumentasi).
- Buat DFD Level 0 untuk gambaran umum sistem.
- Rincikan ke dalam level bertahap sesuai kebutuhan.
-Identifikasi dan fokus pada proses yang utama dan krusial.
- Gunakan simbol dan notasi yang konsisten.
- Libatkan atau kolaborasikan dengan pemangku kepentingan untuk umpan balik.
- Uji DFD untuk memastikan pemahaman aliran data dan proses.
- Dokumentasikan dan cari cara untuk menyederhanakan bagian rumit.
- Lakukan revisi berdasarkan umpan balik secara iteratif.
- Gunakan alat pemodelan untuk visualisasi dan konsistensi.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh AJAY SUPRIADI 04001670 - Rabu, 29 Oktober 2025, 10:29

Jawaban Anda menegaskan pemahaman mendalam terhadap pentingnya DFD dalam pengembangan sistem. tambahkan referensi agar jawaban lebih ilmiah referensi utamakan dari Universitas terbuka

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh HASAN RAPI 053782613 - Rabu, 29 Oktober 2025, 20:35

Assalamualaikum, izin menanggapi diskusinya bapak/ibu tutor.

1. Sampai kapan analisis DFD dilakukan ?

Analisis DFD dilakukan sampai semua proses utama dalam sistem sudah bisa dipahami dengan jelas, dan setiap data antar proses dapat dijelaskan tanpa menimbulkan kebingungan. Biasanya DFD diturunkan samapi level 2 atau level 3, tergantung pada kompleksitas sistem. kalau dilevel tersebut setiap proses ternyata sudah bisa diterjemahkan langsung kedalam modul program,berarti analisis sudah cukup dan bisa lanjut ketahap selanjutnya (desain).

2. Bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu diturunkan lebih lanjut ?

DFD dianggap sudah cukup detail jika:

- setiap proses menggambarkan satu fungsi logis yangbisa diimplementasikan dalam kode program
- semua entitas aliran data dan penyimpanan data sudah jelas asal dan tujuannya ,
- tidak ada aliran data yang menggantung (tidak tahu titik awal dan tujuan)
- tim pengembang, analis dan pengguna sudah dimiliki pemahaman yang sama terhadap alur sistem.

kalau proses sudah bisa diterjemahkan menjadi pseucode atau rancangan modul, maka tidak perlu diturunkan lagi ke level lebih dalam.

3.Dampak DFD yang terlalu sederhana atau terlalu kompleks :

Terlalu sederhana:

- Alur sistem jadi tidak jelas
- Hubungan antar proses tidak terlihat detail
- programer bisa salah memahami kebutuhan sistem

Terlalu kompleks :

- Membuat analisis memakan waktu yang lama
- sulit difahami oleh tim non teknis (misalnya staf administrasi)
- membungungkan karena terlalu banyak detail keci yang belum tentu penting di tahap awal.

4. Pendekatan terbaik untuk menjaga keseimbangan antara kejelasan dan efisiensi :

Pendekatan yang paling efektif adalah top-down approach, yaitu mulai dari gambaran umum (DFD level 0/ konteks diagram), lalu tutunkan sedikit demi sedikit sampai tiap proses sudah jelas fungsinya.

selain itu :

- Fokus pada fungsi utama dulu (pencatatan pasien,jadwal dokter dan pembayaran)
- hanya turunkan level jika prosesnya masih terlalu luas atau belum bisa diimplementasikan langsung
- lakukan review bersama tim agar semua pihak sepakat bahwa detailnya sudah cukup.

sumber referensi :

- Modul **BMP MSIM4303 MODUL ; 4** Rekayasa perangkat lunak untuk pemrograman terstruktur
- **Kendall, K. E., & Kendall, J. E. (2011).***Systems Analysis and Design* (8th Edition). Pearson Education. konsep DFD, levelisasi, serta bagaimana menentukan batas detail analisis.
- **Shelly, G. B., & Rosenblatt, H. J. (2012).***Systems Analysis and Design* (9th Edition). Cengage Learning. metode *top-down approach* dan keseimbangan antara detail dan efisiensi dalam DFD.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Sabtu, 1 November 2025, 07:52

Penjabaran Anda tentang batas level DFD sudah logis. Semoga selalu diberi kesehatan dan semangat untuk belajar!

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh MARTO 055035643 - Kamis, 30 Oktober 2025, 21:13

Assalamu'alaikum

Mohon izin untuk memberikan tugas diskusi sesi ini

Data Flow Diagram(DFD) adalah diagram yang menggambarkan aliran data dari sebuah proses atau sistem informasi. Pada DFD, terdapat informasi terkait input dan output dari setiap proses tersebut. DFD juga memiliki berbagai fungsi, seperti menyampaikan rancangan sistem, menggambarkan sistem, dan perancangan model.

1. Sampai kapan analisis DFD dilakukan? Bagaimana menentukan bahwa DFD sudah cukup detil dan tidak perlu diturunkan lebih lanjut?
Analisis DFD dilakukan hingga level detail yang diperlukan oleh proyek, yang biasanya ditandai ketika semua proses dapat dijelaskan secara memadai dan tidak ada lagi "proses kosong" atau "proses yang terlalu rumit" yang perlu dipecah lebih lanjut. Modul-modul pada DFD level 1 dapat dipecah/breakdown menjadi DFD level 2. Modul mana saja yang harus dipecah/breakdown lebih detail tergantung pada tingkat kedetaikan modul tersebut. Apabila modul tersebut sudah cukup detail dan rinci maka modul tersebut sudah tidak perlu untuk dipecah/breakdown lagi.
2. Apa dampak dari DFD yang terlalu sederhana atau terlalu kompleks dalam pengembangan sistem berbasis pemrograman terstruktur?
Dampak DFD yang terlalu sederhana adalah spesifikasi program kurang jelas, data store mungkin tidak dipetakan secara akurat, kesulitan dalam pengujian.
Dampak DFD yang terlalu kompleks/rumit adalah pemborosan waktu untuk analisis, kesulitan analisis, pemorgraman yang over, masalah sinkronisasi.
3. Bagaimana pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD?
Dengan mnenerapkan leveling (tahapan-tahapan perancangan) yang terkontrol/ketat, menentukan kapan analisis DFD sudah merasa cukup/berhenti (tidak terlalu sederhana atau terlalu kompleks).

Referensi : https://www.sekawanmedia.co.id/blog/dfd-adalah/
Referensi : BUKU MODUL MSIM4303 REKAYASA PERANGKAT LUNAK Rosa Ariani Sukamto

Terima kasih

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Sabtu, 1 November 2025, 07:53

Jawaban Anda menunjukkan pemahaman mendalam. Saya bangga melihat perkembangan Anda. Terus belajar dengan konsisten!

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh BUNGA NUR ROSEANA 050428449 - Kamis, 30 Oktober 2025, 22:39

Assalamu'alaikum warahmatullahi wabarakatuh
Ijin menanggapi diskusi 4 sesi 4 MSIM4303 Rekayasa Perangkat Lunak

1. Batasan Analisis DFD

Analisis DFD dilakukan sampai seluruh proses utama dalam sistem sudah dapat dijelaskan secara logis dan dipahami baik oleh analis, pengguna, maupun pengembang sistem. Proses analisis dapat dihentikan ketika:

1. Setiap proses dalam DFD sudah dapat diterjemahkan ke dalam logika pemrograman atau algoritma.
2. Tidak ada lagi proses yang perlu dipecah menjadi subproses untuk menjelaskan alur data.
3. Semua kebutuhan pengguna sudah terwakili, dan tidak ada data yang mengalir tanpa tujuan atau tanpa sumber yang jelas.

Dengan demikian, analisis DFD dianggap selesai ketika setiap proses dapat dipetakan langsung ke dalam modul program pada tahap desain sistem. Dalam praktiknya, DFD biasanya cukup dikembangkan sampai level 2 atau level 3, tergantung kompleksitas sistem yang dianalisis (Pressman, 2010).

2. Menentukan DFD yang Sudah Cukup Detail

DFD dikatakan cukup detail apabila memenuhi beberapa kriteria berikut:

Setiap proses memiliki input dan output yang jelas dan terdefinisi.
Tidak terdapat aliran data yang menggantung, yaitu data yang tidak memiliki asal atau tujuan.
Setiap proses menggambarkan satu fungsi logis tunggal.
Pengguna sistem dapat memahami alur kerja sistem hanya dengan membaca DFD tersebut tanpa penjelasan tambahan.
Hubungan antarproses dapat ditelusuri dengan mudah dan tidak membingungkan.

Dengan kata lain, jika setiap proses dalam DFD dapat dijelaskan dengan satu kalimat fungsional yang utuh (misalnya: “Menerima data pasien baru dan menyimpan ke basis data”), maka DFD sudah cukup detail dan tidak perlu diturunkan lagi. Pendekatan ini sejalan dengan prinsip analisis sistem terstruktur yang disarankan oleh Kendall & Kendall (2011).

3. Dampak DFD yang Terlalu Sederhana atau Terlalu Kompleks

Keseimbangan dalam kedalaman analisis DFD sangat penting. DFD yang terlalu sederhana maupun terlalu kompleks dapat menimbulkan masalah dalam tahap selanjutnya.

a. Dampak DFD Terlalu Sederhana:

Hubungan antarproses menjadi tidak jelas.
Beberapa kebutuhan pengguna mungkin terlewat.
Programmer atau desainer sistem dapat menafsirkan proses dengan cara yang berbeda, sehingga menghasilkan implementasi yang salah.
Dokumentasi sistem menjadi tidak akurat karena banyak proses yang tidak teridentifikasi.

b. Dampak DFD Terlalu Kompleks:

Membuat proses analisis memakan waktu lebih lama dari yang diperlukan.
Menimbulkan kebingungan karena terlalu banyak simbol, proses, dan data flow.
Menyulitkan komunikasi antara analis sistem dengan pengguna non-teknis.
Dokumentasi sistem menjadi berat, sulit diperbarui, dan tidak efisien.

Menurut Whitten & Bentley (2007), tingkat detail DFD harus disesuaikan dengan tingkat pemahaman pengguna dan kebutuhan desain. DFD yang baik tidak hanya akurat secara teknis, tetapi juga mudah dipahami.

4. Pendekatan untuk Menjaga Keseimbangan dalam Perancangan DFD

Untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi analisis, beberapa pendekatan berikut dapat diterapkan:

1. Top-Down Refinement — mulai dari context diagram (gambaran umum sistem) kemudian menurunkannya secara bertahap ke level yang lebih detail (level 0, 1, 2, dan seterusnya) hanya pada bagian yang perlu diperjelas.
2. Libatkan pengguna dalam proses validasi DFD. Jika pengguna sudah memahami proses sistem melalui DFD yang ada, maka tidak perlu dilakukan penurunan level lebih lanjut.
3. Gunakan satuan fungsi logis. Satu proses sebaiknya mewakili satu fungsi utama yang dapat diimplementasikan sebagai modul dalam sistem.
4. Pastikan konsistensi penamaan dan aliran data agar mudah dipahami oleh seluruh tim pengembang.
5. Lakukan review berkala dengan tim analis dan pengembang untuk memastikan bahwa tingkat detail yang ada sudah cukup untuk diterjemahkan ke dalam rancangan pemrograman.

Pendekatan tersebut membantu menjaga keseimbangan antara kejelasan konseptual dan efisiensi analisis sistem, sebagaimana disarankan oleh Shelly, Cashman, dan Rosenblatt (2012).

Kesimpulan

Analisis DFD pada sistem manajemen klinik SehatMed harus dilakukan hingga setiap proses dapat dipahami secara logis dan dapat diterjemahkan menjadi modul program. DFD yang terlalu sederhana berisiko menyebabkan kesalahan implementasi, sedangkan DFD yang terlalu kompleks dapat memperlambat proses pengembangan. Oleh karena itu, perancangan DFD perlu menyeimbangkan antara kejelasan sistem dan efisiensi analisis melalui pendekatan top-down refinement dan validasi bersama pengguna.

Dengan DFD yang baik dan proporsional, sistem SehatMed dapat dirancang secara efisien, terstruktur, dan mudah dipahami oleh semua pihak yang terlibat — mulai dari analis, pengembang, hingga pengguna akhir.

Sumber Referensi

BMP MSIM4303 REKAYASA PERANGKAT LUNAK
Rosa Ariani Sukamto, S.T,. M.T.
Ejournal_
Kendall, K. E., & Kendall, J. E. (2011). Systems Analysis and Design (8th ed.). Pearson Education.
Pressman, R. S. (2010). Software Engineering: A Practitioner's Approach (7th ed.). McGraw-Hill.
Shelly, G. B., Cashman, T. J., & Rosenblatt, H. J. (2012). Systems Analysis and Design (9th ed.). Cengage Learning.
Whitten, J. L., & Bentley, L. D. (2007). Systems Analysis and Design Methods (7th ed.). McGraw-Hill/Irwin.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh AJAY SUPRIADI 04001670 - Sabtu, 1 November 2025, 07:54

Sangat baik! Anda sudah memahami bahwa DFD yang terlalu kompleks bisa menghambat efisiensi. Tetap semangat ya!

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh 053149469 TAUFIK HIDAYAT - Jumat, 31 Oktober 2025, 09:25

sealamat pagi

nama : Taufik Hidayat
nim : 053149469

1. DFD dilakukan sampai semua proses dalam sistem dapat dijelaskan dengan jelas dan dapat diterjemahkan menjadi modul atau program saat tahap desain dan implementasi.
Ciri-ciri DFD sudah cukup detail:
Setiap proses pada DFD level paling rendahsudah dapat diimplementasikan menjadi sebuah prosedur atau fungsi program tanpa perlu penjabaran lebih lanjut.
Setiap alur data dan data store sudah memiliki definisi yang jelas (input, output, asal, tujuan, dan format datanya).
Tidak ada proses "misterius" (yaitu proses yang tidak diketahui sumber datanya atau ke mana hasilnya pergi).
Pengguna dan tim pengembang sudah memahami alur sistem dengan baik tanpa kebingungan.
Kesimpulan: DFD cukup sampai level di mana setiap proses dapat diuraikan menjadi satu modul program yang siap untuk didesain dan dikodekan (biasanya sampai Level 2 atau Level 3 tergantung kompleksitas sistem).

2. Apa dampak dari DFD yang terlalu sederhana atau terlalu kompleks dalam pengembangan sistem berbasis pemrograman terstruktur?

Kondisi DFD Dampak dalam Pengembangan Sistem
Terlalu Sederhana - Informasi sistem tidak lengkap, banyak proses yang tidak teridentifikasi.
- Developer kesulitan memahami logika sistem secara detail.
- Resiko salah interpretasi tinggi saat implementasi.

- Dokumentasi analisis menjadi kurang berguna.
Terlalu Kompleks Membutuhkan waktu analisis yang lama dan membingungkan tim.
- Banyak level dan proses yang terlalu detail, membuat komunikasi sulit.
- Dokumentasi menjadi berat, padahal tidak semua detail dibutuhkan.
- Menurunkan efisiensi kerja karena terlalu fokus pada hal kecil.

Kesimpulan:
DFD harus cukup untuk menjelaskan fungsi bisnis dan aliran data, tetapi tidak perlu sampai menggambarkan logika algoritmik atau kode program.

3. Bagaimana pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD?
Beberapa pendekatan terbaik untuk menjaga keseimbangan:
1. Mulai dari konteks (level 0) untuk memahami batas sistem dan interaksi eksternal.
2. Turunkan DFD secara bertahap (top-down refinement) hanya pecah proses yang masih terlalu luas atau tidak jelas.
3.Gunakan prinsip “Need to Know”: uraikan hanya bagian yang perlu dipahami untuk implementasi, bukan setiap langkah kecil.
4.Diskusikan dengan pengguna dan pengembang pastikan semua pihak mengerti proses tanpa harus menelaah level yang terlalu rinci.
5.Gunakan kamus data (data dictionary) untuk melengkapi informasi detail tanpa harus membuat DFD menjadi terlalu ramai.

Inti Pendekatan:
DFD yang ideal adalah yang jelas, konsisten, dan cukup detail untuk diterjemahkan ke desain sistem, tanpa membebani proses analisis.
Kesimpulan Umum
DFD dikembangkan sampai semua proses bisa diimplementasikan dalam kode.
Jangan terlalu sederhana (bisa salah tafsir) dan jangan terlalu rinci (bisa boros waktu).
Keseimbangan dicapai dengan pendekatan bertahap, konsultasi dengan pengguna, dan dokumentasi pendukung seperti kamus data.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Sabtu, 1 November 2025, 07:56

Saya apresiasi keaktifan Anda dalam forum ini, tambahkan referensi agar jawaban lebih ilmiah
referensi utamakan dari Universitas terbuka
semoga terus semangat hingga akhir semester!

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AHMAD MINAN NAJIB 050480425 - Jumat, 31 Oktober 2025, 10:34

Assalamualaikum wr wb
Izin menjawab Diskusi 4
Perancangan Data Flow Diagram (DFD) untuk Sistem Manajemen Klinik SehatMed
Sebagai analis sistem, saya akan mulai dengan merancang DFD untuk sistem SehatMed. DFD adalah alat untuk menggambarkan aliran data dalam sistem, mulai dari level tinggi (overview) hingga level detail. Berdasarkan kebutuhan sistem (pencatatan pasien, jadwal dokter, transaksi pembayaran), saya rancang DFD hingga level 1 untuk memberikan gambaran yang jelas tanpa terlalu kompleks.
DFD Level 0 (Context Diagram)
Ini adalah diagram tingkat tertinggi, menunjukkan sistem sebagai satu proses utama yang berinteraksi dengan entitas eksternal.
- Entitas Eksternal:

- Staf Administrasi (menginput data pasien, jadwal, pembayaran).
- Dokter (mengakses jadwal dan catatan pasien).
- Pasien (mendaftar, melihat jadwal, membayar).
- Proses Utama: Sistem Manajemen Klinik SehatMed (proses 0.0).
- Aliran Data:
- Dari Staf Administrasi: Data pasien, jadwal dokter, transaksi pembayaran.
- Dari Dokter: Permintaan jadwal dan catatan pasien.
- Dari Pasien: Permintaan pendaftaran, pembayaran.
- Keluar: Laporan jadwal, konfirmasi pembayaran, catatan medis.
- Penyimpanan Data: Database Klinik (menyimpan data pasien, jadwal, transaksi).

Diagram sederhana: Entitas → Proses 0.0 → Penyimpanan Data → Entitas.

DFD Level 1 (Decomposition)

Memecah proses utama menjadi subproses utama.

- Proses:
- 1.0: Kelola Data Pasien (input/edit data pasien dari staf atau pasien).
- 2.0: Kelola Jadwal Dokter (atur jadwal berdasarkan input staf atau dokter).
- 3.0: Kelola Transaksi Pembayaran (proses pembayaran dari pasien, validasi oleh staf).
- Aliran Data:
- Data pasien mengalir ke 1.0 dan 3.0.
- Jadwal mengalir ke 2.0 dan 1.0 (untuk penjadwalan pasien).
- Pembayaran mengalir ke 3.0.
- Penyimpanan Data: Tetap Database Klinik, dengan tabel seperti Pasien, Dokter, Jadwal, Transaksi.

Ini cukup untuk memahami aliran utama. Jika perlu lebih detail (misalnya, subproses dalam 1.0 seperti validasi data), bisa turun ke level 2.

Jawaban untuk Pertanyaan Diskusi

1. Sampai Kapan Analisis DFD Dilakukan? Bagaimana Menentukan Bahwa DFD Sudah Cukup Detail?

Analisis DFD dilakukan hingga level di mana semua proses utama, aliran data, penyimpanan, dan entitas teridentifikasi dengan jelas, tanpa kebutuhan lebih lanjut untuk pemahaman sistem. Biasanya berhenti di level 1 atau 2, tergantung kompleksitas sistem.

- Penentuan Kecukupan:
- Kriteria Teknis: DFD sudah cukup jika setiap proses dapat dipecah menjadi langkah-langkah primitif (tidak lagi dapat dibagi) yang mudah diimplementasikan dalam kode. Misalnya, jika proses "Kelola Data Pasien" di level 1 masih terlalu luas, turunkan ke level 2 hingga detail seperti "Validasi Input" dan "Simpan ke Database".
- Kriteria Praktis: Diskusikan dengan stakeholder (staf, dokter, pasien) apakah diagram memenuhi kebutuhan mereka. Jika semua aliran data jelas dan tidak ada ambiguitas, berhenti. Juga, evaluasi terhadap spesifikasi fungsional: jika DFD mencakup semua use case (misalnya, pendaftaran pasien, pembayaran), itu cukup.
- Indikator Berhenti: Saat proses tidak lagi mengandung subproses yang signifikan, atau ketika detail lebih cocok untuk pseudocode/diagram alur (flowchart) daripada DFD. Dalam pengembangan terstruktur, DFD biasanya berhenti sebelum desain modul, agar tidak overlap dengan coding.

Secara umum, analisis berhenti ketika DFD mendukung transisi ke desain pemrograman, seperti membuat struktur data atau algoritma.

2. Dampak dari DFD yang Terlalu Sederhana atau Terlalu Kompleks dalam Pengembangan Sistem Berbasis Pemrograman Terstruktur

- DFD Terlalu Sederhana:
- Dampak Negatif: Sistem menjadi kurang jelas, menyebabkan kesalahpahaman aliran data. Misalnya, jika DFD level 0 saja tanpa level 1, staf mungkin tidak paham bagaimana data pasien terhubung dengan jadwal, berujung pada bug dalam kode (seperti duplikasi data atau kehilangan transaksi). Ini memperlambat debugging dan meningkatkan risiko error saat implementasi.
- Konsekuensi: Pengembang mungkin harus menebak detail, menghasilkan kode yang tidak efisien atau tidak sesuai spesifikasi, meningkatkan biaya rework.
- DFD Terlalu Kompleks:
- Dampak Negatif: Membuang waktu analisis, membuat diagram sulit dipahami dan memelihara. Misalnya, jika turun ke level 5 untuk sistem sederhana seperti SehatMed, itu overkill pengembang bisa kelelahan sebelum coding, dan perubahan kecil memerlukan revisi besar. Dalam pemrograman terstruktur, ini bisa menyebabkan kode yang terlalu

modular tapi tidak fleksibel, atau konflik dengan prinsip KISS (Keep It Simple, Stupid).
- Konsekuensi: Proyek tertunda, biaya naik, dan risiko stakeholder bosan atau kehilangan minat. Dalam kasus ekstrem, bisa menyebabkan "analysis paralysis" di mana tim terjebak dalam detail tanpa kemajuan.
Secara keseluruhan, keduanya dapat mengurangi efisiensi pengembangan, meningkatkan risiko kegagalan proyek, dan mengurangi kualitas software akhir.
3. Pendekatan Terbaik untuk Menjaga Keseimbangan Antara Kejelasan Sistem dan Efisiensi dalam Perancangan DFD
- Pendekatan Iteratif: Mulai dengan DFD level 0 untuk overview cepat, lalu turunkan bertahap berdasarkan feedback. Gunakan prototyping: buat mockup DFD dan validasi dengan stakeholder sebelum mendetailkan.
- Prinsip Keseimbangan: Ikuti aturan "just enough detail"—turunkan hingga proses dapat dipetakan ke fungsi pemrograman (misalnya, satu proses DFD = satu modul kode). Hindari detail implementasi seperti algoritma spesifik; fokus pada aliran data.
- Teknik Praktis:
- Gunakan Tools: Software seperti Visio atau Draw.io untuk visualisasi cepat, memungkinkan revisi mudah.
- Kolaborasi: Libatkan tim kecil (analis, pengembang, stakeholder) untuk review. Tetapkan batas level (misalnya, maksimal level 2 untuk sistem kecil seperti ini).
- Metrik Efisiensi: Ukur waktu analisis vs. kompleksitas—jika level 1 sudah jelas, jangan turunkan. Integrasikan dengan metodologi seperti Agile: DFD sebagai artefak awal, lalu refine selama sprint.
- Tips Tambahan: Jika ragu, bandingkan dengan sistem serupa. Untuk SehatMed, level 1 cukup karena fitur utamanya terbatas; hindari over-engineering agar tetap efisien dan fokus pada value delivery.
Pendekatan ini memastikan DFD mendukung pengembangan tanpa membuang waktu, sambil menjaga kejelasan untuk semua pihak. Jika ada detail lebih lanjut atau revisi DFD,saya siap untuk berdiskusi lagi.
Terimakasih
Referensi
BMP MSIM4303 Modul 4
Pembelajaran sesi 4
Pressman, R.S. (2001). Software engineering: A practitioner's approach. fifth edition. New York: Mc Graw Hill.
Sommerville, I. (2016). Software engineering. 10th edition. New York: McGraw-Hill.
Sukamto. R.A. (2018). Rekayasa perangkat lunak terstruktur dan berorientasi objek Bandung: Penerbit Informatika.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Sabtu, 1 November 2025, 07:57

Jawaban yang sangat reflektif. Anda sudah memahami konsep “cukup detail” dalam DFD dengan baik.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh MUHAMAD SATRIA RIZKY 051464226 - Jumat, 31 Oktober 2025, 15:37

Izin menanggapi diskusi
Nama : Muhamad Satria Rizky
Nim : 051464226
1. Sampai kapan analisis DFD dilakukan?

Analisis DFD dilakukan hingga setiap proses dalam sistem sudah tergambar dengan jelas dan tidak membingungkan, artinya:
• Semua input, proses, dan output telah terdefinisi dengan baik.
• Setiap proses tidak bisa lagi dipecah menjadi sub-proses yang berarti (tidak menambah pemahaman baru).
• Pada tahap ini, DFD dianggap cukup detail untuk diteruskan ke tahap desain pemrograman.

Berdasarkan materi, DFD dikembangkan dari Level 0 (Context Diagram) → Level 1 → Level 2 → Level 3, dan seterusnya bila diperlukan.
DFD diturunkan hanya sampai prosesnya cukup rinci, sesuai kebutuhan sistem
2. Dampak DFD yang terlalu sederhana atau terlalu kompleks

• Jika terlalu sederhana:
• Aliran data dan proses menjadi kurang jelas.
• Hubungan antar bagian sistem sulit dipahami oleh tim pengembang.
• Potensi kesalahan interpretasi saat implementasi meningkat.
• Jika terlalu kompleks:
• Membuat analisis bertele-tele dan membingungkan.
• Menyita banyak waktu tanpa menambah nilai pemahaman.
• Risiko redundansi data dan proses berulang.

Kesimpulannya, DFD harus menggambarkan sistem secara jelas namun tetap efisien, tidak terlalu dangkal dan tidak terlalu detail.

3. Pendekatan terbaik untuk menjaga keseimbangan kejelasan dan efisiensi

Pendekatan terbaik adalah dengan:
• Menggunakan prinsip hierarki bertahap (top-down):
• Mulai dari DFD Level 0 untuk gambaran umum.
• Turunkan ke Level 1, 2, dst. hanya bila proses masih perlu dijelaskan.
• Melibatkan pengguna (user) dalam setiap tahap untuk memastikan sistem mudah dipahami.
• Gunakan Kamus Data (Data Dictionary) untuk menjelaskan setiap aliran data tanpa harus menambah level DFD yang berlebihan.
• Fokus pada fungsi utama sistem, bukan pada detail teknis pemrograman.

Kesimpulan Umum

Analisis DFD berhenti ketika diagram sudah cukup menjelaskan semua proses bisnis secara logis dan bisa diterjemahkan ke tahap desain program. Keseimbangan dapat dijaga dengan membuat DFD yang jelas, terstruktur, dan proporsional terhadap kompleksitas sistem.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh AJAY SUPRIADI 04001670 - Sabtu, 1 November 2025, 07:59

> Uraian Anda sangat relevan dengan konsep DFD dalam metodologi terstruktur. tambahkan referensi agar jawaban lebih ilmiah referensi utamakan dari Universitas terbuka

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh BUNGA NUR ROSEANA 050428449 - Sabtu, 1 November 2025, 00:17

Sampai Kapan Analisis DFD Dilakukan?

Analisis DFD dilakukan melalui proses dekomposisi atau penurunan level hingga mencapai level detail yang memadai. Secara umum, proses ini dimulai dari Diagram Konteks (Level 0), dilanjutkan ke Level 1, Level 2, dan seterusnya.

DFD dianggap cukup detail dan tidak perlu diturunkan lebih lanjut (*primitive level*) ketika:

- Setiap proses (lingkaran) pada level DFD tersebut merepresentasikan satu fungsi dasar atau unit kerja logis yang dapat diimplementasikan sebagai satu modul program atau subrutin.
- *Contoh:* Sebuah proses "Memvalidasi Data Pasien" bisa dianggap primitif jika langkah-langkah di dalamnya (seperti cek format nama, cek usia) sudah jelas dan akan dikodekan sebagai satu unit.
- Semua aliran data (*data flow*) yang masuk ke dan keluar dari proses tersebut telah didefinisikan secara lengkap dan sesuai dengan entitas data di Kamus Data (*Data Dictionary*).

- Proses tidak dapat lagi dipecah menjadi fungsi-fungsi yang lebih kecil secara logis dan bermakna untuk tujuan pemrograman. Penurunan lebih lanjut hanya akan menghasilkan langkah-langkah prosedural yang lebih cocok dideskripsikan dalam Spesifikasi Proses (*Process Specification*) atau Minispec.
- Seluruh data *store* (penyimpanan data) yang relevan telah diidentifikasi dan dihubungkan ke proses yang sesuai (misalnya, proses "Pencatatan Pasien" akan membaca/menulis ke *data store* "Data Pasien").

Idealnya, DFD harus diturunkan hingga Level 2 atau Level 3. Namun, ini tergantung pada kompleksitas sistem. Jika suatu proses pada Level 2 masih sangat kompleks, maka perlu diturunkan ke Level 3.

Dampak DFD yang Terlalu Sederhana atau Terlalu Kompleks

1. DFD Terlalu Sederhana (Kurang Detail)

Dampak Negatif	Penjelasan
Ambiguitas Sistem	Fungsi sistem tidak jelas. Sulit bagi pengembang untuk memahami persyaratan fungsional secara lengkap.
Kesalahan Desain	Berpotensi menyebabkan kesalahan desain *database* dan antarmuka karena proses dan aliran data yang sesungguhnya tidak terwakili.
Keterlambatan Implementasi	Pengembang harus sering kembali ke analis untuk mengklarifikasi proses, memperlambat tahap desain dan pemrograman.
Kurangnya *Traceability*	Sulit untuk melacak kebutuhan pengguna (*user requirement*) ke fungsi program yang spesifik.

2. DFD Terlalu Kompleks (Terlalu Rinci)

Dampak Negatif	Penjelasan
Pemborosan Waktu dan Biaya	Analis menghabiskan terlalu banyak waktu untuk membuat detail yang tidak perlu dalam DFD (misalnya, pemecahan menjadi langkah prosedural dasar).
Kehilangan Pandangan Besar	Kesulitan dalam memahami gambaran umum sistem. Diagram menjadi terlalu penuh dan sulit dibaca.
Tumpang Tindih dengan Spesifikasi Proses	DFD menjadi terlalu prosedural, melanggar prinsip DFD sebagai model logis yang berfokus pada APA yang dilakukan sistem, bukan BAGAIMANA langkah-langkahnya (yang merupakan tugas Spesifikasi Proses).
Kekakuan	Diagram yang terlalu rinci lebih sulit diubah jika persyaratan di awal berubah.

Pendekatan Terbaik untuk Menjaga Keseimbangan

Pendekatan terbaik adalah menerapkan prinsip Hierarki dan Modularitas dalam perancangan DFD:

- Mulai dari Atas (Top-Down Approach): Selalu mulai dengan Diagram Konteks, lalu turun ke Level 1 (fungsi utama), dan seterusnya. Ini memastikan Anda tidak kehilangan gambaran besar.
- Kriteria Keseimbangan: Prinsip Primitif: Hentikan dekomposisi ketika setiap proses hanya mencakup satu fungsi logis yang dapat diimplementasikan sebagai modul kode. Jangan turunkan DFD lebih lanjut jika detailnya lebih cocok untuk Spesifikasi Proses (menggunakan *Structured English*, *Decision Table*, atau *Decision Tree*).
  - *Contoh untuk SehatMed:* Proses "Pencatatan Pasien" dipecah menjadi "Input Data Pasien", "Validasi Data Pasien", dan "Simpan Data Pasien" (Level 2). Jika "Validasi Data Pasien" hanya terdiri dari beberapa aturan bisnis yang jelas, proses tersebut bisa dianggap primitif dan *rule-based check*-nya dijelaskan di Spesifikasi Proses, bukan di DFD Level 3.
- Kesesuaian Kamus Data: Pastikan setiap aliran data dan *data store* memiliki definisi yang lengkap dan konsisten di Kamus Data. DFD yang seimbang harus selalu divalidasi dengan kamus datanya.
- Validasi dengan Pengguna dan Pengembang:
  - DFD tingkat tinggi (Level 0, 1) harus mudah dipahami oleh pengguna (staf administrasi, dokter) untuk memverifikasi fungsionalitas.
  - DFD tingkat rendah (Primitif) harus cukup rinci untuk digunakan oleh pengembang sebagai panduan untuk merancang struktur program dan *database*.
- Konsistensi Penomoran (*Balancing*): Selalu pastikan bahwa aliran data yang masuk dan keluar dari suatu proses pada level N harus sama dengan total aliran data yang masuk dan keluar dari semua sub-prosesnya pada

level $N+1$. Ini adalah kunci untuk menjaga keakuratan dan keseimbangan DFD.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh AJAY SUPRIADI 04001670 - Sabtu, 1 November 2025, 08:00

Jawaban Anda mencerminkan pemahaman mendalam terhadap proses analisis sistem, tambahkan referensi agar jawaban lebih ilmiah
referensi utamakan dari Universitas terbuka

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh 053984087 ANIS LISTYADI - Sabtu, 1 November 2025, 09:12

1. Sampai Kapan Analisis DFD Dilakukan?
Analisis DFD dilakukan hingga setiap proses dalam sistem dapat:
• Dipahami secara logis oleh tim pengembang dan pemangku kepentingan
• Diterjemahkan langsung ke dalam modul atau fungsi pemrograman
• Tidak menimbulkan ambiguitas dalam alur data dan proses
Biasanya, DFD diturunkan hingga Level 2 atau Level 3, tergantung kompleksitas sistem. Jika proses di Level 1 masih terlalu umum, maka perlu diturunkan lagi.

Tanda DFD Sudah Cukup Detail:
• Setiap proses memiliki input dan output yang jelas
• Tidak ada proses yang “ajaib” (black box)
• Dapat digunakan sebagai dasar pembuatan flowchart atau pseudocode
• Tim pengembang tidak lagi bertanya “proses ini ngapain?”

2. Dampak DFD Terlalu Sederhana atau Terlalu Kompleks
DFD terlalu sederhana memiliki dampak
- Proses tidak jelas
- Risiko kesalahan implementasi
- Sulit mengidentifikasi kebutuhan sistem
- Dokumentasi tidak cukup untuk pengembangan

DFD terlalu kompleks memiliki dampak
- Membingungkan tim
- Membutuhkan waktu analisis lebih lama
- Menyulitkan komunikasi antar tim
- Bisa membuat proses analisis tidak efisien

3. Pendekatan Terbaik Menjaga Keseimbangan
Untuk menjaga keseimbangan antara kejelasan dan efisiensi, gunakan pendekatan berikut:
• Iteratif dan Bertahap: Mulai dari Context Diagram → Level 1 → Level 2. Hanya turunkan proses yang kompleks.
• Validasi Setiap Level: Libatkan stakeholder untuk memastikan pemahaman dan kesepakatan.
• Gunakan Notasi Standar: Seperti Gane-Sarson atau Yourdon untuk konsistensi.
• Dokumentasi Pendukung: Sertakan deskripsi proses, entitas, dan data store agar tidak hanya bergantung pada diagram.
• Fokus pada Proses Kritis: Prioritaskan proses yang berdampak langsung pada fungsionalitas utama sistem (misalnya: pencatatan pasien, jadwal dokter, transaksi pembayaran).

Sumber :

BMP MSIM4303 Modul 4
https://serupa.id/dfd-data-flow-diagram-komponen-fungsi-level-langkah-merancangnya/#google_vignette
https://www.konsepkoding.com/2024/03/pengertian-dfd-tingkatan-level-0-1-n.html
https://localstartupfest.lokercepat.id/faq/perbedaan-dfd-level-1-dan-2/

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh MUHAMMAD FERDIANSYAH 053535258 - Sabtu, 1 November 2025, 12:27

Izin menjawab

Sistem ini akan mencatat data pasien, mengatur jadwal dokter, dan mengelola transaksi pembayaran. Karena sistemnya cukup besar dan melibatkan banyak pihak (staf, dokter, pasien), maka diperlukan Data Flow Diagram (DFD) untuk memahami bagaimana data mengalir di dalam sistem sebelum sistem benar-benar dikembangkan.

1. Sampai Kapan Analisis DFD Dilakukan?

Penjelasan:

Analisis DFD dilakukan sampai setiap proses dapat dijelaskan secara logis dan dapat diterjemahkan menjadi modul program tanpa kehilangan makna. Artinya, DFD harus dikembangkan hingga level di mana setiap proses sudah dapat diimplementasikan oleh programmer.

Biasanya, DFD diturunkan sampai Level 2 atau Level 3, tergantung pada kompleksitas sistem.

- **Level 0 (Context Diagram):** Menjelaskan sistem secara umum dan hubungan dengan entitas luar (misalnya pasien, dokter, staf).
- **Level 1:** Menjabarkan proses utama seperti *pendaftaran pasien, penjadwalan dokter, dan pembayaran*.
- **Level 2:** Menjabarkan lebih rinci, misalnya *verifikasi data pasien*, *pembuatan jadwal dokter harian*, atau *pencatatan transaksi pembayaran*.

Jika setiap proses di Level 2 sudah cukup jelas untuk dijadikan modul program, maka analisis bisa dianggap selesai.

2. Dampak DFD yang Terlalu Sederhana atau Terlalu Kompleks

Penjelasan:

a. DFD terlalu sederhana → informasi penting hilang.
Jika DFD hanya berisi gambaran umum tanpa rincian proses, pengembang bisa salah menafsirkan kebutuhan sistem. Akibatnya, fitur yang dikembangkan bisa tidak sesuai kebutuhan pengguna.

b. DFD terlalu kompleks → analisis berlebihan dan membuang waktu.
Jika setiap proses dijabarkan terlalu dalam hingga ke detail yang tidak relevan, dokumen analisis menjadi rumit dan sulit dipahami. Hal ini bisa memperlambat pengembangan dan meningkatkan risiko kesalahan interpretasi.

3. Pendekatan Terbaik untuk Menjaga Keseimbangan

Penjelasan:

Untuk menjaga keseimbangan antara kejelasan dan efisiensi, analis sistem dapat menggunakan pendekatan berikut:

1. Gunakan prinsip “Top-Down Analysis” → Mulai dari gambaran besar (context diagram) lalu pecah ke level lebih rinci hanya jika memang dibutuhkan.
2. Pastikan setiap proses punya tujuan jelas dan hasil yang terukur.

3. Lakukan review bersama tim pengembang dan pengguna. Jika semua pihak sudah memahami proses tanpa kebingungan, maka detail sudah cukup.
4. Gunakan notasi standar (Gane & Sarson atau Yourdon) agar diagram mudah dibaca.

Sumber referensi:

- MSIM 4303 / Modul Rekayasa Perangkat Lunak
- Jogiyanto, H.M. (2005). *Analisis dan Desain Sistem Informasi: Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis*. Andi Offset.
- Pressman, R.S. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill.
- Kendall, K. E., & Kendall, J. E. (2014). *Systems Analysis and Design* (9th Edition). Pearson Education.

Tautan permanen Tampilkan induk Balas

Hide sidebars
Course dashboard

Re: Diskusi.4

oleh TRIANA PUTRI WAHYUNI 053839459 - Sabtu, 1 November 2025, 12:42

1. Sampai kapan analisis DFD dilakukan? Bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu diturunkan lebih lanjut?
Analisis DFD dilakukan hingga setiap proses di dalam sistem sudah dapat dijelaskan secara jelas dan bisa diterjemahkan ke dalam modul program. DFD dianggap cukup detail jika:

- Setiap proses sudah memiliki input dan output yang jelas.

- Tidak ada proses yang masih bersifat umum atau abstrak.

- Setiap proses bisa langsung dijadikan dasar pembuatan modul pada tahap desain pemrograman.

Penurunan DFD dilakukan secara bertahap dari diagram konteks → DFD level 0 → DFD level 1 → DFD level n, dan dihentikan ketika proses sudah tidak perlu diuraikan lagi karena sudah dapat diimplementasikan secara langsung.

2. Apa dampak dari DFD yang terlalu sederhana atau terlalu kompleks dalam pengembangan sistem berbasis pemrograman terstruktur?

- DFD terlalu sederhana: membuat sistem sulit dipahami, hubungan antarproses tidak jelas, dan kebutuhan pengguna bisa terabaikan. Akibatnya, hasil desain program bisa tidak sesuai dengan sistem sebenarnya.

- DFD terlalu kompleks: analisis menjadi memakan waktu lama, dokumentasi sulit dibaca, dan proses implementasi bisa terhambat karena terlalu banyak detail yang tidak penting.

Perancangan yang terlalu sederhana menyebabkan informasi tidak lengkap, sedangkan yang terlalu kompleks akan menimbulkan redundansi dan kebingungan pada tahap desain.

3. Bagaimana pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD?
Pendekatan terbaik adalah menggunakan metode top-down refinement, yaitu memulai dari proses yang paling umum lalu menurunkannya hanya jika dibutuhkan untuk memperjelas fungsi. Selain itu, lakukan review bersama pengguna dan pengembang untuk memastikan bahwa setiap level sudah cukup menjelaskan alur data tanpa berlebihan.

Keseimbangan dicapai dengan memastikan DFD memenuhi prinsip "jelas, ringkas, dan mudah ditelusuri" dari konteks

hingga level akhir.

Sumber Referensi:

Universitas Terbuka. (2021). Buku Materi Pokok STSI4202 – Rekayasa Perangkat Lunak.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh 054546411 AIMAN YUSUF WICAKSONO - Sabtu, 1 November 2025, 14:14

1. Sampai Kapan Analisis DFD Dilakukan? Bagaimana Menentukan DFD Sudah Cukup Detail?, Analisis DFD dilakukan sampai proses terpecah menjadi unit-unit logis terkecil yang sesuai. DFD dianggap cukup detail jika Atomisitas Tercapai: Setiap proses di level terendah hanya melakukan satu tugas tunggal yang tidak dapat dipecah lebih lanjut secara logis dan penting bagi pengguna. Siap untuk Spesifikasi: Setiap proses dapat dijelaskan sepenuhnya dan tanpa ambigu menggunakan satu alat spesifikasi proses (seperti Structured English atau Decision Table), yang kemudian akan menjadi cetak biru bagi programmer.
2. Terlalu Sederhana (Kurang Detail), Menyebabkan ambiguitas logika bagi pengembang, menghasilkan asumsi yang salah, kesalahan *coding*, dan pemborosan waktu klarifikasi selama fase pemrograman. Terlalu Kompleks (Terlalu Rinci) Menyebabkan pemborosan waktu untuk analis dan desainer, DFD sulit dimaintenance, dan dapat mengyurangi fleksibilitas implementasi yang seharusnya menjadi wewenang seorang *programmer*.
3. Bagaimana Pendekatan Terbaik untuk Menjaga Keseimbangan?

   Pendekatan terbaik adalah menggunakan Dekomposisi Fungsional Bertahap (Top-Down):
   - Mulai dari Umun : Tentukan sub-sistem utama (seperti Manajemen Pasien, Manajemen Pembayaran).
   - Penurunan Selektif: Hanya pecah proses yang kompleks dan mengandung logika bisnis penting (seperti validasi data atau perhitungan) hingga Level 2 atau 3.
   - Verifikasi Kriteria Primitif: Hentikan penurunan segera setelah proses di level terendah dapat diubah menjadi satu modul program yang jelas. Ini menjaga efisiensi sambil memastikan semua logika bisnis penting sudah terekam.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh ILHAN RAHARJA NALANKO 051475472 - Sabtu, 1 November 2025, 15:20

Izin Bapak Tutor menjawab diskusi kali ini

1.Sampai Kapan Analisis DFD Dilakukan?

Analisis Data Flow Diagram (DFD) dilakukan pada tahap analisis sistem untuk menggambarkan aliran data dan proses dalam sistem secara logis. Berdasarkan **modul MSIM4303 Rekayasa Perangkat Lunak Universitas Terbuka Modul 4**, tujuan DFD adalah memudahkan pemahaman alur data dari entitas luar ke dalam sistem hingga menghasilkan informasi yang dibutuhkan.

Analisis DFD dimulai dari **diagram konteks (Level 0)**, lalu dikembangkan ke **DFD Level 1** dan **Level 2** jika proses masih kompleks. Penurunan level dihentikan ketika setiap proses sudah cukup jelas dan dapat dijelaskan secara spesifik tanpa perlu dipecah lagi, atau disebut **proses primitif**.

DFD dianggap **cukup detail** ketika seluruh aliran data, penyimpanan data, dan entitas eksternal telah tergambar lengkap, serta tidak ada ambiguitas dalam proses bisnis. Selain itu, semua pihak yang terlibat seperti analis, pengguna, dan pengembang — telah memahami dan menyetujui diagram tersebut.

Analisis DFD tidak perlu diteruskan jika penambahan detail tidak lagi memberikan manfaat berarti, justru menambah kerumitan. Dalam kasus sistem manajemen klinik **SehatMed**, DFD dianggap selesai ketika proses utama seperti pendaftaran pasien, penjadwalan dokter, dan transaksi pembayaran telah tergambarkan jelas dan siap dijadikan dasar dalam tahap desain dan pemrograman.

2. Dampak DFD yang Terlalu Sederhana atau Terlalu Kompleks

DFD yang **terlalu sederhana** maupun **terlalu kompleks** dapat menimbulkan masalah dalam pengembangan sistem berbasis pemrograman terstruktur. Jika DFD terlalu sederhana, aliran data dan proses penting bisa terlewat, sehingga pengembang kesulitan memahami kebutuhan sistem secara lengkap. Hal ini dapat menyebabkan kesalahan pada desain modul, basis data, dan implementasi program karena informasi logis tidak cukup jelas.

Sebaliknya, DFD yang terlalu kompleks membuat diagram sulit dipahami oleh analis, pengembang, maupun pengguna. Terlalu banyak detail membuat proses analisis dan desain menjadi lambat, membingungkan, serta menghabiskan waktu untuk bagian yang kurang penting.

Oleh karena itu, DFD harus **cukup tepat dan seimbang** mampu menggambarkan proses utama, aliran data, dan penyimpanan dengan jelas. Dengan demikian, DFD menjadi alat komunikasi yang efektif antara analis, pengguna, dan pengembang, sekaligus memudahkan perancangan modul dan implementasi program dalam sistem berbasis pemrograman terstruktur.

3.pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD

Pendekatan terbaik untuk menjaga keseimbangan antara **kejelasan sistem** dan **efisiensi** dalam perancangan DFD adalah dengan menggunakan prinsip **dekomposisi bertahap (top-down)**. DFD dimulai dari **diagram konteks (Level 0)** untuk menunjukkan interaksi sistem dengan entitas eksternal. Selanjutnya, diagram dijabarkan menjadi **Level 1** dan **Level 2** hanya jika proses tertentu masih terlalu kompleks.

Setiap proses dijabarkan hingga mencapai **proses primitif**, yaitu level yang cukup detail untuk dipahami dan diimplementasikan tanpa menimbulkan kebingungan. Dekomposisi dihentikan ketika tambahan detail tidak lagi menambah pemahaman, sehingga DFD tetap efisien.

Untuk menjaga keterbacaan, jumlah proses dalam satu diagram dibatasi, biasanya maksimal 6–9 proses per level. Review bersama pengguna dan pengembang penting dilakukan untuk memastikan diagram mudah dipahami dan sesuai kebutuhan sistem.

Dengan pendekatan ini, DFD menjadi **cukup jelas tanpa berlebihan**, memudahkan komunikasi antar pemangku kepentingan, mendukung desain sistem yang terstruktur, dan efisien untuk tahap implementasi.

Sekian Dari pendapat saya jika ada kekurangan nya mohon di berikan tambahan nya Bapak/ibu Tutor Terimakasih.

Sumber refrensi :

Universitas Terbuka. (2021). *Modul STSI4202 – Rekayasa Perangkat Lunak*. Tangerang Selatan: Universitas Terbuka.Modul 4 Rekayasa Perangkat Lunak Untuk Pemrograman Terstruktur.

https://www.webdesign-inspiration.com/article/data-flow-diagrams-best-practices-and-common-mistakes-to-avoid/

https://www.atlassian.com/work-management/project-management/data-flow-diagram

https://idcloudhost.com/blog/apa-itu-data-flow-diagram/

https://www.sekawanmedia.co.id/blog/dfd-adalah/

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh MUHAMMAD RIZQI PRATAMA 053580633 - Sabtu, 1 November 2025, 20:47

1. Analisis dilakukan secara mendalam dan bertahap, dalam penggunaan DFD aliran data yang terjadi pada sistem adalah sejauh apa tingkat pandang penggunaan diagram dan pengalaman dari yang menggunakan diagram untuk merepresentasikan jalannya aliran data. Analisis DFD dilakukan berjenjang dari:

- Level 0 (Context Diagram) menggambarkan sistem yang akan dibuat sebagai entitas tunggal yang saling berinteraksi dengan sistem lain.
- Level 1 untuk menggambarkan modul-modul yang ada didalam sistem yang akan dikembangkan
- Level 2 dan seterusnya adalah pecahan/hasil breakdown lanjutan dari Level sebelumnya menjadi lebih detail.

Analisis DFD berhenti ketika setiap proses sudah cukup detail untuk dijelaskan logikanya secara utuh tanpa perlu dipecah lagi. Dengan kata lain setiap proses memiliki input, output dan kamus data yang sudah jelas maka modul sudah tidak perlu dipecah/breakdown lagi.

2. **Dampak DFD terlalu sederhana**

Jika DFD dibuat terlalu sederhana maka Programmer yang membuat aplikasi SehatMed akan kebingungan dalam menjabarkan logika bisnisnya. Misalkan dalam aplikasi Sehat Med terjadi proses kelola transaksi, programmer kebingungan apa maksud dan tujuan proses transaksi, apakah menghitung transaksi, mencetak diskon, atau menyimpan data transaksi. Sehingga dampak yang terjadi adalah kegagalan dalam pengimplementasian, karena aliran data tidak teralir dengan baik.

Dampak DFD terlalu kompleks

Jika DFD dibuat terlalu kompleks akan menghabiskan waktu dalam proses analisis, tim pengembang akan kesulitan menerjemahkan detail-detail kecil dalam proses. DFD yang memiliki banyak level cenderung dapat membingungkan pengguna non teknis yang dalam hal ini Dokter, Administrasi dan Pasien.

3. Dalam Perancangan DFD, pendekatan terbaik agar terjadi keseimbangan sistem dan efesien adalah dengan cara bertahap dengan metode top-down:

- Mulai dengan context diagram level 1 untuk menggambarkan sistem yang akan dibuat sebagai entitas tunggal yang saling berinteraksi dengan sistem lain.
- Turunkan ke level 2 jika masih terdapat proses yang kompleks dan perlu dibreakdown, dan diteruskan ke level selanjutnya.
- Dan dekomposisi dihentikan saat seluruh persyaratan fungsi dan pengecualian sudah tercakup. dan Proses input/output serta kamus data sudah jelas.
- Kemudian libatkan pemangku kepentingan dari klinik SehatMed dalam setiap tahapan proses pengembangan.

Contoh breakdown DFD untuk aplikasi klinik SehatMed

Level 0

- Entitas luar: Pasien, Dokter, Admin.
- Sistem utama : Sistem Manajamen Klinik SehatMed

Level 1

- Pendaftaran & Manajemen data pasien
- Manajemen Data penjadwalan dokter
- Pelayanan & catatan Rekam Medis
- Pembayaran & Tagihan

Level 2 (sub proses dari Penjadwalan Janji)

- Informasi Ketersediaan Dokter (Input: Request jadwal kosong) >(Output: Jadwal tersedia)
- Konfirmasi & Booking (Input: Jadwal tersedia, Data Pasien) > (Output: update jadwal terbooking, memberikan notifikasi)

Sumber: Universitas Terbuka. (2021). BMP STSI4202 – Rekayasa Perangkat Lunak.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh ELVIRA CITRA PARDENY 051115567 - Sabtu, 1 November 2025, 21:12

1. Sampai kapan analisis DFD dilakukan? Bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu diturunkan lebih lanjut?
Analisis DFD dilakukan hingga setiap proses dalam sistem dapat dijelaskan secara jelas, logis, dan dapat diimplementasikan pada tahap desain pemrograman. Menurut modul BMP MSIM4303 Rekayasa Perangkat Lunak (Universitas Terbuka), DFD dikembangkan secara bertingkat (leveling), mulai dari diagram konteks, DFD level 0, dan dapat diturunkan ke level 1, 2, dan seterusnya sesuai kompleksitas sistem.
Proses penurunan DFD dapat dihentikan ketika setiap proses sudah cukup jelas menggambarkan aliran data dan dapat diimplementasikan tanpa menimbulkan ambiguitas. Artinya, jika semua input, output, dan penyimpanan data sudah teridentifikasi dan tidak ada proses yang membingungkan, maka DFD sudah dianggap cukup detail.

2. Apa dampak dari DFD yang terlalu sederhana atau terlalu kompleks dalam pengembangan sistem berbasis pemrograman terstruktur?
• DFD terlalu sederhana:
o Menyebabkan pemahaman sistem menjadi kabur.
o Hubungan antar proses, aliran data, dan entitas eksternal tidak tergambar jelas.
o Pengembang berisiko salah menafsirkan kebutuhan sistem, sehingga dapat menyebabkan kesalahan dalam desain atau implementasi.
• DFD terlalu kompleks:
o Membuat dokumen analisis sulit dipahami dan memakan waktu lebih lama untuk dibuat.
o Bisa menimbulkan kebingungan, terutama bagi tim pengembang baru.
o Menyulitkan proses validasi bersama pengguna karena terlalu banyak detail teknis yang belum relevan di tahap analisis.
Dengan demikian, keseimbangan antara tingkat detail dan keterbacaan sangat penting agar DFD efektif digunakan sebagai alat komunikasi antara analis, pengembang, dan pengguna.

3. Bagaimana pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD?
Pendekatan terbaik adalah dengan menggunakan top-down approach:
1. Mulai dari konteks diagram untuk menggambarkan sistem secara umum dan hubungan dengan entitas luar.
2. Turunkan ke DFD level 0 dan level 1 hanya pada bagian yang kompleks atau memerlukan rincian tambahan.
3. Hentikan penurunan jika proses sudah cukup spesifik untuk diterjemahkan menjadi algoritma atau modul program.
Selain itu, penting untuk melibatkan pengguna (user involvement) dalam setiap tahap validasi DFD agar model yang dibuat tetap relevan dan mudah dipahami.

Kesimpulan
DFD harus dikembangkan hingga tingkat yang menjamin pemahaman penuh tentang aliran data tanpa menimbulkan ambiguitas, tetapi tidak perlu terlalu rinci hingga membingungkan. Tujuan utama DFD adalah kejelasan komunikasi dan dasar kuat untuk desain sistem.

Sumber Referensi:
Universitas Terbuka. (2021). Modul 4 : Rekayasa Perangkat Lunak Untuk Pemograman Terstruktur,(MSIM4303 Rekayasa Perangkat Lunak). Tangerang Selatan: Universitas Terbuka. Halaman 4.6 – 4.9.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4
oleh 054208558 SASTRO MANURUNG - Sabtu, 1 November 2025, 23:04

izin menjawab pada diskusi ini
Nama: Sastro Manurung
Nim: 054208558

Analisis DFD dilakukan secara bertahap dan iteratif selama fase analisis sistem. Proses ini berhenti ketika DFD telah mencapai tingkat detail yang memadai untuk:

1. **Memahami Proses Bisnis Utama:** DFD harus mampu menggambarkan alur informasi dan proses utama dalam sistem, seperti pendaftaran pasien, penjadwalan dokter, konsultasi medis, pembayaran, dan pelaporan.
2. **Mengidentifikasi Entitas dan Data Penting:** DFD harus menunjukkan entitas eksternal yang berinteraksi dengan sistem (misalnya, pasien, dokter, laboratorium), serta data yang mengalir antara entitas dan proses (misalnya, data pasien, jadwal, hasil pemeriksaan).
3. **Menentukan Fungsi-Fungsi Sistem:** Setiap proses dalam DFD harus merepresentasikan fungsi sistem yang spesifik dan terdefinisi dengan baik. Fungsi-fungsi ini nantinya akan menjadi dasar untuk modul-modul program.
4. **Memvalidasi Kebutuhan Pengguna:** DFD harus divalidasi dengan pengguna (staf administrasi, dokter) untuk memastikan bahwa diagram tersebut akurat dan mencerminkan kebutuhan mereka.

Kriteria DFD Cukup Detail:

- **Setiap Proses Dapat Dijelaskan dengan Algoritma Sederhana:** Jika setiap proses dalam DFD dapat dipecah menjadi langkah-langkah algoritma yang jelas dan mudah dipahami, maka DFD sudah cukup detail.
- **Data Store Mewakili Entitas Data yang Terstruktur:** Data store (penyimpanan data) dalam DFD harus mewakili entitas data yang terstruktur dan dapat diimplementasikan dalam basis data.
- **Tidak Ada Ambiguitas:** DFD harus bebas dari ambiguitas atau ketidakjelasan. Setiap elemen dalam diagram harus memiliki makna yang jelas dan konsisten.
- **Pengguna Memahami dan Menyetujui:** Pengguna (staf klinik, dokter) memahami dan menyetujui DFD sebagai representasi yang akurat dari sistem.

Dampak DFD Terlalu Sederhana atau Terlalu Kompleks

- **DFD Terlalu Sederhana:**
  - **Kurang Jelas:** Sistem menjadi kurang jelas, sehingga sulit bagi pengembang untuk memahami kebutuhan dan fungsi sistem secara detail.
  - **Implementasi Tidak Lengkap:** Fungsi-fungsi penting mungkin terlewatkan, menyebabkan sistem tidak lengkap atau tidak memenuhi kebutuhan pengguna.
  - **Kesulitan dalam Desain:** Desainer sistem akan kesulitan merancang modul-modul program karena kurangnya informasi detail.
- **DFD Terlalu Kompleks:**
  - **Membuang Waktu:** Analisis menjadi terlalu lama dan memakan sumber daya yang berlebihan.
  - **Sulit Dipahami:** DFD menjadi sulit dipahami dan dikelola, bahkan oleh analis sistem itu sendiri.
  - **Over-Engineering:** Sistem mungkin menjadi terlalu kompleks dan sulit dipelihara karena terlalu banyak detail yang tidak perlu.

Pendekatan Terbaik untuk Menjaga Keseimbangan

1. **Pendekatan Top-Down:** Mulai dengan DFD level 0 (diagram konteks) yang memberikan gambaran umum sistem. Kemudian, dekomposisi setiap proses menjadi DFD level yang lebih detail (level 1, level 2, dst.) sesuai kebutuhan.
2. **Fokus pada Proses Utama:** Prioritaskan analisis DFD untuk proses-proses utama yang paling penting bagi bisnis klinik. Proses-proses yang kurang penting dapat dianalisis dengan tingkat detail yang lebih rendah.
3. **Iterasi dan Validasi:** Lakukan iterasi dan validasi DFD secara berkala dengan pengguna. Dapatkan umpan balik dari mereka untuk memastikan bahwa diagram tersebut akurat dan relevan.
4. **Gunakan Notasi Standar:** Gunakan notasi DFD yang standar (misalnya, Gane & Sarson atau Yourdon & Coad) untuk memastikan konsistensi dan kemudahan pemahaman.

5. **Dokumentasi yang Baik:** Dokumentasikan setiap elemen dalam DFD dengan jelas dan ringkas. Berikan penjelasan mengenai tujuan, input, output, dan logika setiap proses.

Referensi

Meskipun tidak ada referensi tunggal yang mencakup semua aspek ini, beberapa sumber yang relevan meliputi:

- "System Analysis and Design" by Kenneth E. Kendall and Julie E. Kendall
- "Modern Systems Analysis and Design" by Jeffrey A. Hoffer, Joey F. George, and Joseph S. Valacich
- Artikel-artikel tentang DFD dan analisis sistem di jurnal-jurnal ilmiah dan konferensi terkait rekayasa perangkat lunak.Tutorial dan panduan tentang DFD di situs web seperti Visual Paradigm, Lucidchart, dan Creately.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.4

oleh 053929596 FEIZA RAHMAH ASSYIFA - Sabtu, 1 November 2025, 23:53

Nama: Feiza Rahmah Assyifa
NIM: 053929596
Prodi: Sistem Informasi

Selamat malam pak Ajay Supriadi, S.Kom., M.Kom. Izinkan saya memberikan tanggapan atas soal pada diskusi sesi 4 ini.

DFD adalah alat visual yang digunakan untuk menggambarkan aliran data dalam sistem, dimulai dari level tinggi (context Diagram) hingga level detail. Dalam konteks sistem manajemen klinik SehatMed yang melibatkan pencatatan pasien, jadwal dokter, hingga transaksi pembayaran, DFD membantu memetakan interaksi antara staff administrasi, dokter, dan pasien.

1. Sampai kapan Analisis DFD Dilakukan ?
Analisis DFD dilakukan secara literatif hingga mencapai tingkat detail yang memungkinkan pemahaman penuh tentang fungsi sistem oleh semua stakeolder sambil memastikan bahwa diagram dapat digunakan sebagai dasar untuk spesifikasi teknis dan desain pemrograman. Biasanya, analisis dimulai dari context diagram (lelcel 0) dan diturunkan hingga 2 atau 3, tergantung kompleksitas sistem. Dan proses berhenti Ketika,
• Setiap proses utama telah di pecah menjadi suatu proses yang jelas dan dapat di implementasikan.
• Aliran data (data flows) dan penyimpanan data (data stores) telah terdefinisi dengan baik, tanpa ambigutas yang dapat menyebabkan kesalahan dalam pengembangan.
• Feedback stakeholder menunjukan bahwa diagram cukup untuk memandu desain tanpa kebutuhan penjelasan tambahan.
Analisis DFD tidak berhenti pada satu tahap, ia melibatkan validasi berkelanjutan selama fase analisis kebutuhan jika sistem seperti SehatMed melibatkan integrasi dengan sistem eksternal, misalnya pembayaran online. Analisis mungkin perlu di perluas hingga level yang lebih dalam untuk menangani kompleksitas tersebut.
Menurut DeMarco (1979), analisis DFD dilakukan hingga tingkat di mana spesifikasi fungsional sistem dapat di turunkan secara logis, memastikan bahwa diagram tidak hanya mengggambarkan “apa” yang terjadi, tetapi juga “bagaimana” data mengalir. Hal ini didukunh oleh Davis (1983) di “communications of the ACM”, yang menunjukan bahwa analisis DFD efektif hingga level 2-3 untuk sistem berbasis pemrograman terstruktur.
Lalu, bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu di turunkan lebih lanjut ?
DFD akan dianggap cukup detail Ketika,
• Setiap proses dalam diagram dapat dijelaskan dalam 5-7 langkah primitive (primitive processes) yang tidak memerlukan dekomposisi lebih lanjut, sesuai dengan aturan “rule of thumb” dalam analisis terstruktur.
• Data flows dan data stores telah di verifikasi melalui walkthrough engan stakeholder, memastikan konsistensi dan kelengkapan (misalnya, semua entitas eksternal seperti pasien dan dokter telah teridentifikasi).
• Diagram dapat langsung di terjemahkan ke spesifikasi fungsional tanpa kebutuhan asumsi tambahan, dan tidak ada redundansi atau konflik dalam aliran data.
• Jika penurunan lebih lanjut tidak menambah nilai pemahaman atau efisiensi, seperti dalam kasus proses sederhana

(misalnya, “catat pasien” yang sudah jelas pada level 2).
Untuk sistem SehatMed, misalnya level 0 menunjukan interaksi keseluruhan, level 1 memecah menjadi proses utama yaitu pencatatan, jadwal, pembayaran dan level 2 cukup jika subproses seperti validasi pasien sudah dapat diimplementasikan. Penentuan ini sering dilakukan melalui kriteria kualitatif, apakah diagram memungkinkan prototyiping atau desain modul tanpa revisi besar?.
Yourdon (1989) menyatakan bahwa DFD cukup detail Ketika proses dapat di petakan ke modul program tanpa kehilangan informasi, dengan dukungan dari penelitian dalam “Journal of System and Software” oleh Pfleeger (1998), yang menekankan validasi melalui inspeksi formal untuk menghindari over decomposition.

2. Apakah dampak dari DFD yang terlalu sederhana atau terlalu kompleks dalam pengembangan sistem berbasis pemrograman terstruktur?
Jika DFD terlalu sederhana, maka sistem menjadi kurang jelas, karena aliran data dan proses tidak terperinci. Sehingga risiko kesalahan implementasi meningkat, misalnya data pasien hilang atau jadwal dokter tumpeng tindih. Ini dapat menyembabkan biaya perbaikan tinggi selama pengujian, serta kesulitan dalam pemeliharaan sistem. Dalam konteks SehatMed, DFD sederhana mungkiin mengabaikan integrasi pembayaran, mengakibatkan bug dalam transaksi.

Dan jika DFD terlalu kompleks, membuang waktu dan sumber daya. Karena analisis terlalu mendalam tanpa memanfaatkan proporsional,membuat diagram sulit dipahami oleh stakeholder non-teknis. Ini meningkatkan kompleksitas analisis, risiko kesalahan manusia, dan biaya pengembangan. Untuk sistem tersturktur, kompleksitas berlebihan dapat memperlambat transisi ke desain kode, seperti dalam pemrograman modular dimana DFD level 4+ tidak di perlukan.

Secara keseluruhan kedua ekstrem dapat menggangu efisiensi pengembangan, dengan dampak negative pada jadwal proyek dan kualitas perangkat lunak.
Pressman (2010), membahas dampak ini dalam konteks rekayasa perangkat lunak, menunjukan bahwa DFD sederhana meningkatkan risiko defek hingga 30% (berdasarkan studi empiris), sementara kompleksitas berlebihan dapat menambah waktu analisis hingga 50%. Dukungan tambahan dari jurnal “IEEE Transactions on Software Engineering” oleh basili dan Weiss (1984), yang menganalisis trade-off dalam diagram alisan data dan menemukan bahwa keseimbangan optimal mengurangi defek implementasi.

3. Bagaimana pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD?
Pendektatan terbaik melibatkan litersi bertahap dan partisipasi stakeholder, yaitu :
• Dimulai dari tingkat tinggi
• Menggunakan kriteria keseimbangan
• Literasi dan feedback
• Praktik tambahan
Pendekatan ini memastikan DFD mendukung pemrograman terstruktur dengan modul yang jelas, sambil menghindari over-engineering.
Yourdon (1989) merekomendasikan pendekatan literatif untuk keseimbangan, di dukung oleh penelitian dalam “information and software technology” oleh sommerville (2007), yang menunjukan bahwa literasi dengan feedback stakeholder mengurangi kompleksitas DFD hingga 40% sambil meningkatkan kejelasan. Tambahan dari jurnal “ACM Computing Surveys” oleh Davis (1993), yang mengadvokasi penggunaan alat otomatis untuk efisiensi dalam perancangan DFD.

Sumber Referensi :

- Ariani Sukamto, Rosa. Rekayasa Perangkat Lunak untuk Pemrograman Terstrukur. BMP Rekayasa Perangkat Lunak MSIM4303, 4.44.9. Tangerang Selatan: Universitas Terbuka.
- Sekawan Media. Robith Adani, Muhammad. (2025). Data Flow Diagram (DFD): Pengertian, Jenis, Fungsi & Contoh. Diakses pada tanggal 01 November 2025 dari https://www.sekawanmedia.co.id/blog/dfd-adalah/ YouTube.com.
- Fajar Pradana. (2020). Belajar Sistem Informasi | 5. Mengenal Data Flow Diagram (DFD). Diakses pada tanggal 01 November 2025 dari

Tautan permanen Tampilkan induk Balas

Diskusi.4

oleh MIA AULIA 051618649 - Minggu, 2 November 2025, 00:50

Assalamualaikum bapak, Izin melampirkan jawaban diskusi 4

Sampai kapan analisis DFD dilakukan? Bagaimana menentukan bahwa DFD sudah cukup detail dan tidak perlu diturunkan lebih lanjut

Sampai kapan analisis DFD dilakukan?

Analisis DFD dilanjutkan hingga seluruh alur proses bisnis sistem tergambarkan secara utuh, jelas, dan logis, yaitu:

Analisis Diagram Alir Data (DFD) dianggap mencapai titik penyelesaiannya ketika beberapa kriteria kunci telah terpenuhi. Kondisi ini mencakup kemampuan untuk melacak seluruh aliran data dari sumber hingga tujuannya, terdefinisikannya input dan output untuk setiap proses secara eksplisit, serta tidak adanya lagi ambiguitas mengenai fungsi dari setiap proses tersebut. Dalam praktiknya, kegiatan analisis DFD ini dilaksanakan sepanjang tahap analisis sistem, yang mendahului fase perancangan program. Proses ini dinyatakan selesai ketika setiap komponen proses telah dapat diuraikan secara logis dan dengan tingkat kedetailan yang memadai untuk dijadikan dasar dalam pembuatan modul program.

Kriteria akhir analisis DFD bukanlah ketika semua diagram telah selesai dibuat, melainkan ketika hasilnya telah mencapai tingkat kedetailan yang memadai untuk menjadi acuan langsung dalam perancangan basis data, antarmuka, dan modul pemrograman.

Data Flow Diagram (DFD) memiliki sejumlah level hierarkis yang masing-masing memberikan gambaran detail mengenai aliran data dalam sistem. Berikut penjelasan tiap level DFD:

DFD Level 0 (Context Diagram)

Pada level tertinggi yang bersifat simplistis ini, keseluruhan sistem direpresentasikan sebagai sebuah proses tunggal. Tujuannya adalah untuk mengilustrasikan batas-batas sistem serta aliran data utama yang melintas antara sistem dengan entitas eksternal, sebelum proses-proses di dalamnya diuraikan lebih detail pada level-level selanjutnya.

DFD Level 1

Level ini memberikan gambaran yang lebih mendetail dengan cara memecah proses utuh dari Level 0 ke dalam subproses-subproses penyusunnya. Diagram Level 1 mengungkap aliran data antar proses utama tersebut serta interaksinya dengan berbagai data store, sehingga memberikan gambaran awal mengenai bagaimana data diproses dan disimpan dalam sistem.

DFD Level 2

Level ini memberikan granularitas tertinggi dengan cara mendekomposisi satu subproses Level 1 menjadi langkah-langkah elementer. Diagram Level 2 mengungkap logika proses dan pergerakan data secara mendalam, sehingga memberikan fondasi yang jelas untuk perancangan logika pemrograman.

DFD Level 3 dan seterusnya

Dekomposisi dapat dilanjutkan ke level yang lebih rendah (Level 3, 4, dst.) selama prosesnya masih kompleks. Namun, sebagai pedoman praktis, Level 3 seringkali menjadi batas akhir yang disarankan. Diagram yang terlalu detail justru akan kehilangan nilai komunikasinya dan menjadi sulit untuk dikelola serta dipahami.

Bagaimana menentukan bahwa DFD sudah cukup detail?

Keterlacakan proses

Setiap proses dalam DFD harus memiliki fungsi yang terdefinisi dengan jelas, yang dapat diungkapkan dalam satu kalimat operasional sederhana. Kalimat ini menggambarkan transformasi spesifik yang dilakukan terhadap data.

Keterpahaman

Kualitas sebuah DFD diukur dari kejelasan dan kemudahan pemahamannya. Diagram yang baik memungkinkan semua pihak, terlepas dari latar belakang teknisnya, untuk memahami alur proses bisnis secara langsung dan tanpa memerlukan elaborasi lebih lanjut.

Kejelasan input–output

Setiap proses dalam DFD harus memiliki integritas fungsional, yang ditandai dengan kejelasan dan ketepatan aliran data. Input yang diterima dan output yang dihasilkan harus terdefinisi dengan baik dan selaras secara logis dengan tujuan serta kebutuhan sistem secara keseluruhan.

Keterhubungan antarlevel

Sebuah DFD dikatakan konsisten secara hierarkis ketika tidak terjadi ketidakseimbangan aliran data antar level. Setiap input dan output yang dimiliki suatu proses di level parent harus persis sama dengan aliran data yang masuk dan keluar dari diagram child-nya, sehingga memastikan keterlacakan yang sempurna.

Kesiapan implementasi

Kualitas sebuah proses dalam DFD dapat diukur dari kemampuannya untuk dijadikan spesifikasi yang jelas bagi sebuah modul program. Jika sebuah proses telah terdefinisi dengan input, output, dan logika transformasi yang tepat, maka proses tersebut sudah siap untuk dikonversi menjadi sebuah fungsi atau kelas dalam implementasi

Jika semua kriteria di atas sudah terpenuhi, maka DFD sudah cukup detail dan tidak perlu diturunkan lagi.

Apa dampak dari DFD yang terlalu sederhana atau kompleks dalam pengembangan sistem berbasis pemrograman terstruktur

Dalam evaluasi kualitas DFD, sebuah diagram yang hanya berfokus pada proses utama tanpa dilengkapi dengan detail aliran data dan dekomposisi subproses dianggap tidak memadai. Kesederhanaan yang berlebihan justru mengaburkan kompleksitas sistem yang seharusnya dimodelkan, sehingga menyulitkan proses analisis lebih lanjut.

Dalam pengembangan berbasis pemrograman terstruktur, hal ini bisa menimbulkan beberapa dampak serius:

Dampak DFD yang terlalu sederhana

• Keterlacakan proses

Sebuah kelemahan fatal dalam DFD adalah ketika logika alur datanya tidak jelas. Kondisi ini menyebabkan diagram gagal mengkomunikasikan inti dari sistem yaitu, bagaimana data diproses dan ditransformasikan dari titik masukan hingga ke titik keluaran.

• Desain modul program

Ketiadaan spesifikasi yang jelas mengenai cakupan dan tanggung jawab masing-masing proses dalam DFD menghambat perancangan arsitektur program. Tanpa pemisahan concern yang jelas, pembagian sistem menjadi modul-modul yang independen dan terstruktur menjadi suatu hal yang sulit untuk direncanakan dan diimplementasikan.

• Komunikasi tim

Perbedaan interpretasi mengenai fungsi sistem di antara berbagai pemangku kepentingan menandakan perlunya penyelarasan persepsi yang lebih efektif, baik melalui dokumentasi yang lebih detail, komunikasi yang intensif, maupun prototipe yang dapat memvisualisasikan perilaku sistem dengan lebih jelas

Dampak DFD yang terlalu kompleks

• Waktu analisis

Durasi tahap analisis yang berlebihan dapat mengakibatkan inefisiensi proyek secara keseluruhan. Ketidakseimbangan antara waktu yang diinvestasikan dan nilai yang dihasilkan berpotensi mengganggu alur kerja, menunda tahap implementasi, serta mengurangi kelincahan proyek dalam merespons perubahan.

• Kejelasan dokumentas

Sebuah DFD seharusnya berfungsi sebagai panduan yang terfokus. Keberadaan detail yang berlebihan dan tidak berkontribusi langsung terhadap perancangan logika program justru mengalihkan perhatian dari proses-proses kunci dan hubungan data yang kritikal

• Fokus analisis

Sebuah proyek yang efektif harus berpusat pada bisnis, bukan pada teknis. Ketika diskusi tim didominasi oleh detail implementasi teknis dan mengabaikan esensi alur bisnis, maka yang terjadi adalah pembangunan sistem yang mungkin secara teknis solid, namun tidak memberikan nilai tambah yang optimal bagi pengguna.

Bagaimana pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD?

Pendekatan terbaik untuk menjaga keseimbangan antara kejelasan sistem dan efisiensi dalam perancangan DFD adalah dengan mengikuti prinsip-prinsip berikut:

• Mulai dari tingkat tinggi (Level 0): Buatlah context diagram yang sederhana untuk menunjukkan hubungan utama antara sistem dan entitas eksternal, tanpa terlalu banyak rincian.

• Lanjut ke level berikutnya sesuai kebutuhan: Setelah memahami gambaran umum, turunkan ke level yang lebih rinci (Level 1 dan Level 2) hanya untuk proses yang kompleks dan membutuhkan penjelasan lebih mendalam. Jangan menambahkan rincian yang tidak memberikan manfaat signifikan.

• Gunakan prinsip keseimbangan (balancing): Pastikan input dan output dari proses di level yang berbeda tetap konsisten dan tidak menimbulkan kekaburan. Setiap proses harus memiliki aliran data yang masuk dan keluar yang logis serta tidak berlebihan.

• Libatkan stakeholder: Validasi DFD secara aktif dengan pengguna akhir dan tim pengembang agar kedua pihak memahami diagram dan tidak ada yang merasa diagram terlalu sederhana atau terlalu rumit.

• Fokus pada proses penting dan data utama: Hindari memperlihatkan detail yang berlebihan—cukup tampilkan proses utama, aliran data kritis, dan data store yang relevan. Detail kecil bisa ditambahkan jika benar-benar diperlukan untuk pemahaman.

• Validasi keberhasilan secara berkala: Jika diagram sudah mampu merepresentasikan aliran data secara lengkap tanpa hilangnya kejelasan dan dengan tingkat detail yang memadai untuk pengembangan, maka DFD dapat dianggap cukup

Sumber:

https://www.slideshare.net/slideshow/pemodelan-sistem-dfd/49594328#6

https://www.alvindocs.com/blog/data-flow-diagram

https://www.sekawanmedia.co.id/blog/dfd-adalah/

https://www.codepolitan.com/blog/mengenal-data-flow-diagram-dfd/

Tautan permanen Tampilkan induk Balas

Hide sidebars
Course dashboard
Materi Inisiasi
Lompat ke...

Navigasi

Dasbor
Beranda situs
Laman situs
Kelasku
STSI4203.108
STSI4202.42
Peserta
Nilai
Pendahuluan
Sesi 1
Sesi 2
Sesi 3
Sesi 4
Kehadiran Sesi ke-4
Materi Inisiasi
Diskusi.4
STSI4103.119
MKKI4201.278
STSI4201.161
STSI4205.331
STSI4104.284
MKDI4202.1514
Kelas

Administrasi

Forum administrasi
Berlangganan dinonaktifkan

Follow Us:

Hide sidebars

IIVERSITAS TERBUKA ©2025

da masuk sebagai INDRAWAN LISANTO 053724113 (Keluar)

patkan aplikasi seluler

Course dashboard