iskusi.2

Lakukan: Kirim balasan: 1

Jatuh tempo: Minggu, 26 Oktober 2025, 23:59

menampilkan balasan dalam bentuk bertingkat

Setelan

Diskusi.2
Rabu, 28 Mei 2025, 10:22

Setelah mempelajari materi pada sesi kedua tentang Kerangka Kerja dan Paradigma Interaksi, diskusikanlah topik berikut ini.

Topik Diskusi 2

1. Jika Anda adalah seorang perancang dalam pembuatan sebuah aplikasi, jelaskan kompleksitas kerangka kerja yang saudara alami/hadapi saat membuat suatu aplikasi tersebut!
2. Jelaskan, bagaimana cara Anda mengatasi kompleksitas kerangka kerja tersebut!

Saat memberikan tanggapan diskusi, harap diingat hal-hal berikut:

- tanggapan diskusi merupakan hasil tulisan dan pemikiran sendiri, **bukan hasil *copy* dan *paste* pekerjaan teman atau sumber belajar lain (internet)**.
- jika menggunakan/mengambil cuplikan materi dari sumber lain, cantumkan judul sumbernya (artikel/buku) atau *link* alamat dari internet.
- jika mengambil referensi dari berbagai sumber, tuliskan ulang sesuai pemahaman sendiri, **bukan hasil *copy* dan *paste* secara utuh.**
- tanggapan diskusi yang hasil *copy* and *paste* akan diberikan nilai minimal dan berpotensi mendapat nilai 0 (nol)

Selamat berdiskusi...

Tautan permanen Balas

Re: Diskusi.2

oleh SHERLIN FRISCILLA 053760206 - Senin, 13 Oktober 2025, 06:07

Izin menjawab diskusi.

1. Sebagai perancang aplikasi, saya sering menghadapi kompleksitas pada kerangka kerja (framework) yang digunakan. Tantangan yang muncul biasanya berkaitan dengan banyaknya fitur dan konfigurasi bawaan, struktur proyek yang kaku, serta penyesuaian antara kebutuhan aplikasi dengan aturan framework. Misalnya, framework tertentu memiliki pola dan cara kerja yang harus diikuti, sehingga sulit jika ingin melakukan kustomisasi di luar aturan tersebut. Selain itu, penggunaan *library* tambahan dari pihak ketiga terkadang menimbulkan masalah, seperti ketidakcocokan versi dengan framework utama atau munculnya error yang sulit ditemukan penyebabnya.

2. Berikut adalah cara untuk mengatasi kompleksitas kerangka kerja:

1. Memahami dasar dan arsitektur framework secara mendalam melalui dokumentasi dan pelatihan singkat.
2. Membuat perencanaan arsitektur aplikasi sejak awal, agar penggunaan framework tidak bertabrakan dengan kebutuhan sistem.
3. Melakukan eksperimen dan uji coba bertahap, supaya kesalahan bisa ditemukan lebih cepat.
4. Menggunakan komunitas atau forum pengembang, karena banyak solusi bisa ditemukan dari pengalaman orang lain.
5. Mendokumentasikan setiap konfigurasi dan solusi agar memudahkan perawatan di kemudian hari.

Dengan cara tersebut, kompleksitas framework dapat dikendalikan dan aplikasi bisa dikembangkan dengan lebih efisien serta mudah dipelihara.

Sumber referensi:

Santosa, P. I. (2021). Interaksi Manusia dan Komputer (Edisi 1). Tangerang Selatan : Universitas Terbuka.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh 053586444 ZEKY BOY NATA SANDRA - Senin, 13 Oktober 2025, 14:02

Izin menjawab diskusi

1. Kurva Pembelajaran yang Curam: Setiap kerangka kerja memiliki sintaks, konsep, dan praktik terbaiknya sendiri. Mempelajari kerangka kerja baru membutuhkan waktu dan usaha yang signifikan, terutama jika Anda belum familiar dengan paradigma yang digunakan (misalnya, pemrograman reaktif, arsitektur berbasis komponen).
2. Konfigurasi yang Rumit: Beberapa kerangka kerja memerlukan konfigurasi yang ekstensif, yang bisa menjadi rumit dan rentan terhadap kesalahan. Konfigurasi yang salah dapat menyebabkan masalah kinerja, keamanan, atau fungsionalitas.
3. Ketergantungan yang Kompleks: Kerangka kerja sering kali memiliki ketergantungan pada pustaka dan modul lain. Mengelola ketergantungan ini bisa menjadi rumit, terutama jika ada konflik versi atau masalah kompatibilitas.
4. Abstraksi yang Berlebihan: Beberapa kerangka kerja menyediakan abstraksi yang tinggi, yang dapat menyederhanakan pengembangan tetapi juga menyembunyikan detail implementasi yang penting. Hal ini dapat membuat sulit untuk memahami bagaimana kerangka kerja bekerja di bawah permukaan dan memecahkan masalah yang kompleks.
5. Keterbatasan Kerangka Kerja: Setiap kerangka kerja memiliki keterbatasan dan trade-off-nya sendiri. Memilih kerangka kerja yang tepat untuk proyek Anda membutuhkan pemahaman yang mendalam tentang kebutuhan proyek dan kemampuan kerangka kerja yang tersedia.

Cara Mengatasi Kompleksitas Kerangka Kerja:

1. Pemilihan Kerangka Kerja yang Tepat:
- Analisis Kebutuhan: Identifikasi kebutuhan proyek Anda secara rinci, termasuk fungsionalitas, kinerja, keamanan, dan skalabilitas.
- Evaluasi Kerangka Kerja: Evaluasi berbagai kerangka kerja berdasarkan kebutuhan proyek Anda. Pertimbangkan faktor-faktor seperti popularitas, dukungan komunitas, dokumentasi, dan contoh kode.
- Uji Coba: Lakukan uji coba dengan kerangka kerja yang berbeda untuk melihat mana yang paling cocok untuk proyek Anda.
2. Pembelajaran Bertahap:
- Mulai dari Dasar: Pelajari dasar-dasar kerangka kerja sebelum mencoba fitur yang lebih kompleks.
- Ikuti Tutorial dan Contoh: Manfaatkan tutorial, contoh kode, dan dokumentasi resmi untuk mempelajari cara menggunakan kerangka kerja.
- Berpartisipasi dalam Komunitas: Bergabunglah dengan forum, grup diskusi, atau konferensi untuk berinteraksi dengan pengembang lain dan mendapatkan bantuan.
3. Konfigurasi yang Terstruktur:
- Gunakan Alat Bantu Konfigurasi: Manfaatkan alat bantu konfigurasi yang disediakan oleh kerangka kerja untuk menyederhanakan proses konfigurasi.
- Dokumentasikan Konfigurasi: Dokumentasikan semua pengaturan konfigurasi untuk memudahkan pemeliharaan dan pemecahan masalah.
- Gunakan Konvensi: Ikuti konvensi konfigurasi yang direkomendasikan oleh kerangka kerja untuk memastikan konsistensi dan mengurangi kesalahan.
4. Manajemen Ketergantungan yang Efektif:
- Gunakan Manajer Paket: Manfaatkan manajer paket seperti npm, pip, atau Maven untuk mengelola ketergantungan proyek Anda.
- Tentukan Versi yang Tepat: Tentukan versi ketergantungan yang kompatibel dengan kerangka kerja Anda.
- Perbarui Ketergantungan Secara Teratur: Perbarui ketergantungan Anda secara teratur untuk mendapatkan perbaikan bug, peningkatan kinerja, dan fitur baru.
5. Memahami Abstraksi:
- Pelajari Kode Sumber: Jika memungkinkan, pelajari kode sumber kerangka kerja untuk memahami bagaimana abstraksi bekerja di bawah permukaan.
- Gunakan Alat Bantu Debugging: Manfaatkan alat bantu debugging untuk melacak alur eksekusi dan memahami bagaimana kerangka kerja berinteraksi dengan kode Anda.
- Jangan Takut untuk Menyelam Lebih Dalam: Jika Anda menghadapi masalah yang kompleks, jangan takut untuk menyelam lebih dalam ke dalam kerangka kerja untuk mencari solusi.
6. Pola Desain yang Tepat:
- Pelajari Pola Desain: Pelajari pola desain yang umum digunakan dalam pengembangan perangkat lunak, seperti MVC, MVP, atau MVVM.
- Gunakan Pola Desain yang Sesuai: Gunakan pola desain yang sesuai untuk memecahkan masalah yang spesifik dalam proyek Anda.
- Konsisten: Terapkan pola desain secara konsisten di seluruh kode Anda untuk meningkatkan keterbacaan dan pemeliharaan.

Dengan pendekatan yang tepat, kompleksitas kerangka kerja dapat diatasi, memungkinkan Anda untuk membangun aplikasi yang kuat, efisien, dan mudah dipelihara.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh 054378345 MAROLOP BACTIAR SIMANJUNTAK - Senin, 13 Oktober 2025, 18:47

Nama : Marolop Bactiar Simanjuntak
Nim : 054378345
Upbjj : UT Batam

Sebagai perancang aplikasi, kompleksitas kerangka kerja yang dihadapi mencakup tantangan dalam pengelolaan struktur sistem, integrasi antar komponen, serta keseimbangan antara fleksibilitas dan stabilitas dalam pengembangan perangkat lunak. Kerangka kerja (framework) yang digunakan sering kali membawa lapisan abstraksi

yang tinggi, yang mempermudah pengembangan awal namun dapat menyulitkan pemahaman mendalam terhadap alur eksekusi sistem, terutama saat terjadi kesalahan atau kebutuhan modifikasi mendalam.

Kompleksitas dalam Penggunaan Kerangka Kerja

A. Akumulasi dependensi, Yaitu ketergantungan antar modul yang tinggi sehingga perubahan pada satu bagian dapat memengaruhi bagian lain secara tidak terduga.

B. Ketidaksesuaian antara kebutuhan aplikasi dan fitur bawaan framework yang dapat menyebabkan pemborosan sumber daya atau penulisan kode tambahan yang tidak efisien (boilerplate code). Penggunaan framework yang terlalu besar untuk proyek kecil, atau sebaliknya, dapat mengakibatkan over-engineering atau scalability issues di masa depan. Kompleksitas juga muncul dalam bentuk pemeliharaan kode jangka panjang, terutama ketika framework mengalami pembaruan yang tidak kompatibel dengan versi sebelumnya (breaking changes).

C. Integrasi API, Aplikasi modern sering kali memerlukan integrasi dengan berbagai Application Programming Interface (API) eksternal. Setiap API memiliki cara kerja yang berbeda, dan mengintegrasikannya ke dalam aplikasi bisa menimbulkan masalah kompatibilitas dan keamanan.

D. Manajemen State, Aplikasi perlu mengelola state atau kondisi data aplikasi secara efisien. Kerangka kerja sering kali menyediakan mekanisme untuk manajemen state, tetapi memahami dan mengimplementasikannya dengan benar bisa menjadi kompleks, terutama dalam aplikasi yang besar dan kompleks

Strategi Mengatasi Kompleksitas

Untuk mengatasi kompleksitas tersebut, diterapkan pendekatan sistematis berbasis prinsip rekayasa perangkat lunak:

-Desain Modular dan Berlapis (Layered Architecture): Aplikasi dirancang dalam bentuk modul-modul terpisah seperti *presentation layer, business logic layer*, dan *data access layer*, yang memungkinkan pengembangan, pengujian, dan pemeliharaan secara independen.

-Penerapan Prinsip SOLID: Prinsip desain seperti *Single Responsibility Principle dan Dependency Inversion* diterapkan untuk memastikan kode bersifat fleksibel, mudah diuji, dan tahan terhadap perubahan.

-Pemilihan Teknologi yang Proporsional**:** Framework dipilih berdasarkan skala proyek, kebutuhan performa, dan kemampuan tim pengembang, bukan semata-mata berdasarkan popularitas.

-Penggunaan Metrik Kualitas Kode: Metrik seperti *cyclomatic complexity* dan *code coupling* digunakan untuk menilai kualitas kode secara objektif dan mengidentifikasi area yang memerlukan refaktorisasi.

Dokumentasi Arsitektur dan Standar Pengkodean**:** Dokumentasi yang komprehensif dan standar pengkodean yang konsisten diterapkan untuk memastikan keberlanjutan proyek dan kemudahan transfer pengetahuan antar anggota tim.

Dengan menerapkan strategi-strategi tersebut, kompleksitas kerangka kerja dapat dikelola secara efektif, sehingga menghasilkan aplikasi yang tidak hanya stabil dan aman, tetapi juga siap untuk pengembangan dan skala di masa depan. Pendekatan ini telah terbukti efektif dalam berbagai konteks pengembangan perangkat lunak, baik dalam skala kecil maupun sistem enterprise.

Sumber Referensi:
-BMP Modul 3 MSIM4208/Interaksi Manusia dan Komputer.

-https://appmaster.io/id/blog/kerangka-kerja-pengembangan-perangkat-lunak-yang-efektif

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh IQBAL FADILLAH 053532435 - Senin, 13 Oktober 2025, 20:55

Yth.Bapak/Ibu Tutor dan Teman-teman Mahasiswa UT
izinkan saya untuk menanggapi hasil diskusi tersebut sebagai berikut:

Pertanyaan:
1. Jika Anda adalah seorang perancang dalam pembuatan sebuah aplikasi, jelaskan kompleksitas kerangka kerja yang saudara alami/hadapi saat membuat suatu aplikasi tersebut!
2. Jelaskan, bagaimana cara Anda mengatasi kompleksitas kerangka kerja tersebut!

Jawab:

1. Jika saya memposisikan diri saya sebagai perancang aplikasi,saya sangat mengerti apabila kompleksitas kerangka kerja sangat sering muncul saat dimana saya harus merancang sebuah fitur yang mampu memenuhi sebuah kebutuhan pengguna / user dengan teknologi dan sumber daya yang terbatas.
tantangan yang saya harus hadapi adalah untuk memastikan alur kerja aplikasi tetap berjalan logis dan intuitif,seperti misalnya pengelolaan input dan juga pemprosesan datanya. misalnya pada letak tombol icon "save" yang mudah ditemukan oleh user. karena jika tombol tersebut tidak ada dimenu yang di harapakan atau dibagian yang mudah ditemukan , nantinya hal itu akan menjadi masalah yang disebut jarak pemisah eksekusi yaitu perbedaan antara keinginan pengguna dengan apa yang
bisa dilakukan sistem. lalu permasalahan lainnya adalah ketika sebuah respon dari sistem itu tidak jelas, sehingga user menjadi kesulitan untuk mengevaluasi hasil tindakanya, yang tercermin pada jarak pemisah evaluasi.

2.Untuk mengatasi kompleksitas tersebut, yang saya lakukan adalah saya akan melakukan analisis kebutuhan pengguna secara mendalam serta berupaya memahami sudut pandang mereka selama
proses desain. saya juga mencoba mendekatkan desain dengan cara pengguna berpikir dan berinteraksi dengan aplikasi, agar proses belajar dan penggunaannya terasa lebih alami.
Selain itu, saya melakukan uji interface sedini mungkin untuk mengidentifikasi sebuah masalah pemetaan dan affordance, sehingga pengguna lebih mudah menemukan fungsi yang diinginkan.
Menurut sumber, dalam interaksi manusia-komputer, penting bagi perancang untuk melihat permasalahan dari sudut pandang pengguna supaya aplikasi yang dibuat lebih efektif dan efisien ketika digunakan.

Sumber Referensi:

Dewi Widianti, U. (2016). KERANGKA KERJA DAN PARADIGMA INTERAKSI (https://repository.unikom.ac.id/49098/1/Kerangka%20Kerja%20dan%20Paradigma%20Interaksi.pdf)

Diklatkerja.com, Penjelasan Tentang Interaksi Manusia dan Komputer, 2024 (https://diklatkerja.com/blog/penjelasan-tentang-interaksi-antara-manusia-dan-komputer)

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh 054017943 RIFKI SETIAWAN - Senin, 13 Oktober 2025, 21:45

Selamat Malam, perkenalkan saya Rifki Setiawan (054017943) Prodi S1 Sistem Informasi UPBJJ Purwokerto.

Izin menjawab diskusi :

1. Kompleksitas kerangka kerja yang saudara alami/hadapi saat membuat suatu aplikasi tersebut

berikut beberapa kompleksitas Kerangka kerja yang dihadapi dalam membuat suatu aplikasi :

1. Kompleksitas Kognitif: Kesenjangan antara Model Pengguna dan Model Sistem

Kompleksitas ini muncul dari perbedaan antara cara pengguna memahami suatu tugas (model mental pengguna) dan cara sistem sebenarnya bekerja (model sistem). Seperti yang diungkapkan Donald Norman, kesenjangan ini dapat menyebabkan kebingungan, frustrasi, dan kesalahan. Misalnya, pengguna mungkin mengira fitur "arsip" sama dengan "hapus", sementara sistem memperlakukan kedua perintah tersebut secara sangat berbeda.

2. Kompleksitas Fungsional: Menyederhanakan Fitur yang Banyak ke dalam Antarmuka yang Ramah

Aplikasi modern sering kali memiliki ratusan fitur. Tantangannya adalah bagaimana merancang antarmuka yang tidak terlihat penuh atau rumit, sekaligus memastikan semua fungsi yang penting dapat ditemukan dan diakses dengan mudah oleh pengguna yang akan menggunakan fitru tersebut.

3. Kompleksitas Teknis: Adaptasi dengan Batasan Kerangka Kerja Pengembangan

Sebagai perancang, saya harus bekerja dalam ekosistem teknis tertentu (seperti React, Flutter, dll.). Setiap kerangka kerja memiliki komponen, pola, dan batasannya sendiri. Desain yang ideal di kepala sering kali harus disesuaikan dengan realitas teknis agar dapat diimplementasikan secara efisien tanpa mengorbankan pengalaman pengguna secara signifikan.

4. Kompleksitas Proses: Mengelola Siklus Desain yang Iteratif
Perancangan yang efektif adalah proses berulang. Kompleksitas terletak pada bagaimana mengelola umpan balik dari pengguna, hasil pengujian kegunaan (usability testing), dan perubahan kebutuhan proyek secara terstruktur, tanpa menyebabkan kebingungan .

2. Cara Mengatasi Kompleksitas Kerangka Kerja dalam membuat suatu aplikasi Tersebut :

1. Mengatasi Kompleksitas Kognitif dengan Pendekatan User-Centered Design (UCD)

Menerapkan prinsip User-Centered Design yang ditekankan dalam modul IMK dengan membuat persona atau scenario dan Menerapkan metafora yang tepat misalnya "keranjang belanja" di e-commerce untuk menjembatani kesenjangan antara model pengguna dan model sistem.

2. Mengatasi Kompleksitas Fungsional dengan Penerapan Prinsip Desain Universal

Menggunakan prinsip-prinsip desain interaksi untuk menyederhanakan kerumitan.

Seperti dengan :

- Hierarki Visual & Pengelompokan (Chunking): Mengatur informasi dan fungsi ke dalam kelompok-kelompok yang logis. Misalnya, semua pengaturan akun dikelompokkan dalam satu menu "Profil".
- Konsistensi: Memastikan tombol, ikon, dan terminologi memiliki makna dan perilaku yang sama di seluruh aplikasi. Prinsip ini sejalan dengan "Aturan Emas" Shneiderman yang menekankan konsistensi.
- Kesederhanaan: Menghilangkan elemen-elemen yang tidak relevan dan menyajikan hanya yang dibutuhkan pengguna pada saat yang tepat.

3. Mengatasi Kompleksitas Teknis melalui Kolaborasi dan Prototiping

Dengan kolaborasi Awal dengan Pengembang dan melibatkan pengembang sejak fase perancangan awal untuk memahami batasan dan peluang teknis. Bisa dengan Memahami Design System kemudian mempelajari dan menggunakan komponen dari design system kerangka kerja teknis yang dipilih (seperti Material Design) untuk memastikan desain dapat diimplementasikan secara konsisten dan efisien.

4. Mengatasi Kompleksitas Proses dengan Evaluasi Iteratif

Dengan mengadopsi siklus iteratif yang berkelanjutan seperti Prototipe -> Evaluasi -> Perbaiki , mengevaluasi heristik dan Memberikan feedback secara berkala terhadap design yang disempurnakan sebelum melalu tahap finishing.

Kesimpulan:
Kompleksitas kerangka kerja aplikasi bisa diatasi dengan perencanaan matang, pemilihan framework yang sesuai, pendekatan modular, serta pemanfaatan sumber daya pembelajaran dari dokumentasi dan komunitas. Langkah-langkah ini membantu pengembang menghindari kendala dan mempercepat proses pengembangan tanpa mengabaikan kualitas ataupun kebutuhan pengguna akhir.

Referensi:

- Materi Sesi 2, Modul Interaksi Manusia dan Komputer (Edisi 1). Tangerang Selatan: Universitas Terbuka.

- Norman, D. A. (2013). The Design of Everyday Things.
- Shneiderman, B., et al. (2016). Designing the User Interface: Strategies for Effective Human-Computer Interaction.
- Nielsen, J. (1994). 10 Usability Heuristics for User Interface Design.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh MUHAMMAD RAIHAN 052550297 - Senin, 13 Oktober 2025, 23:28

1. Jika saya seorang perancang dalam pembuatan aplikasi, menurut saya tantangan paling terasa itu di bagian menyesuaikan antara kebutuhan pengguna sama batas kemampuan sistem. Kadang ide desainnya sudah bagus, tapi saat diterapkan ternyata tidak cocok di semua perangkat, atau bahkan bikin pengguna bingung. Karena dalam kerangka kerja interaksi itu juga harus memikirkan bagaimana cara orang berinteraksi dengan aplikasi. Mulai dari tampilan, alur menu, sampai respon sistem. Di situ lah kompleksitasnya muncul, karena menurut saya tiap pengguna punya kebiasaan dan ekspektasi yang beda-beda.
2. Menurut saya, untuk mengatasi hal itu yaitu dengan buat rancangan awal atau *prototype* dulu, jadi bisa dicoba langsung sama pengguna. Dari situ lah kelihatan mana bagian yang kurang pas dan perlu diperbaiki. Selain itu, menerapkan prinsip kemudahan dan kenyamanan penggunaan agar aplikasinya tidak hanya bagus secara tampilan, tapi juga enak dipakai. Walau prosesnya memang agak rumit, tapi kalau dijalani secara bertahap sambil mendengarkan masukan pengguna, hasilnya bisa jauh lebih baik.

Referensi:

Santosa, P. I. (2021). *Interaksi manusia dan komputer.* Universitas Terbuka.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh VENCIA ALFIONITA 054761859 - Selasa, 14 Oktober 2025, 10:07

1. Kalau saya menjadi perancang aplikasi , tantangan atau kompleksitas yang saya alami biasanya muncul saat mengatur Alur kerja dan fitur fitur aplikasi supaya semuanya bisa berjalan lancar.

Misalnya, Ketika saya membuat aplikasi pencatatan keuangan , saya agak kesulitan untuk menyusun bagian tampilan, logika dan penyimpanan data agar saling terhubung dengan baik. Selain masalah itu, ada juga masalah saat aplikasi di jalankan di perangkat yang berbeda, karena perangkat yang berbeda membuat tampilannya tidak selalu sama. Kesulitan lainnya adalah saat bekerja dengan tim misalnya alat yang di gunakan , jadi butuh penyesuaian agar hasil akhirnya tetap sesuai.

2. Cara mengatasi kompleksitas tersebut adalah

- Membuat rencana dan rancangan yang jelas di awal seperti bagan alur dan pembagian tugas.

- Melalukan uji coba (testing) secara bertahap setelah menambahkan fitur baru.

- mendiskusikan dan menyakan versi alat atau framework yang di gunakan dengan tim supaya tidak ada perbedaan hasil.

Sumber :

- pressman,R.S & Maxim, B.R (2020) , Software Engineering : A practitioner's Approach. McGraw-Hill Education.

- Sommerville, I (2016). Software Engineering. Person Education

- GeekaforGeeks. (2024). Common Challenges in application Development. https://www.geeksforgeeks.org

Terimakasih.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh RAHMAWATI FARIED 052616039 - Selasa, 14 Oktober 2025, 12:20

Assalamualaikum.. Maaf ijin menjawab
Nama : Rahmawati Faried
Nim : 052616039

1. Ketika seseorang akan merancang aplikasi, maka seseorang tersebut akan memformulasikan suatu ide tentang bagaimana aplikasi tersebut akan bekerja dan menilai seberapa jauh benda tersebut mampu membantu pengguna dalam menyelesaikan suatu pekerjaan yang dibutuhkannya.
kompleksitas kerangka kerja dapat muncul dari beberapa aspek:
- Skalabilitas: Memastikan aplikasi dapat menangani peningkatan jumlah pengguna dan data tanpa penurunan kinerja.
- Keamanan: Melindungi data sensitif dan mencegah akses yang tidak sah.
- Integrasi: Menggabungkan berbagai layanan dan API eksternal dengan lancar.
- Kinerja: Mengoptimalkan kecepatan dan responsifitas aplikasi.
- Pemeliharaan: Memastikan kode mudah dipahami, diubah, dan diuji untuk pembaruan dan perbaikan bug di masa mendatang.

2. cara mengatasi kompleksitas kerangka kerja tersebut!
Proses untuk menciptakan kerangka kerja tentang suatu proses atau cara kerja suatu benda disebut model mental. Model mental Adalah penyajian kognitif suatu proses atau objek yang menyatakan suatu perkiraan logis dan dapat diterima tentang bagaimana suatu benda dibentuk atau bagaimana benda berfungsi
Berikut adalah beberapa cara untuk mengatasi kompleksitas kerangka kerja:
- Perencanaan yang matang: Merencanakan arsitektur aplikasi dengan cermat, memilih kerangka kerja yang sesuai, dan membuat desain yang modular.
- Penggunaan alat dan teknologi yang tepat: Memanfaatkan alat bantu pengembangan, pustaka, dan kerangka kerja yang dapat menyederhanakan tugas-tugas kompleks.
- Pengujian yang komprehensif: Melakukan pengujian unit, pengujian integrasi, dan pengujian sistem secara teratur untuk memastikan kualitas dan keandalan kode.
- Dokumentasi yang baik: Membuat dokumentasi yang jelas dan terperinci untuk memudahkan pemahaman, pemeliharaan, dan kolaborasi.
- Pelatihan dan pengembangan: Memberikan pelatihan kepada tim pengembangan untuk meningkatkan keterampilan dan pengetahuan mereka tentang kerangka kerja dan teknologi yang digunakan.

Referensi :
BMP MSIM4208 modul 03

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh RIVI ANDIXAN PURNADI 053565988 - Selasa, 14 Oktober 2025, 14:43

Sebagai seorang perancang dalam pembuatan aplikasi, menurut pengalaman saya, kompleksitas kerangka kerja (framework) yang sering saya hadapi meliputi:

1. Kurva Pembelajaran yang Curam:
Framework tertentu seperti Laravel atau React memiliki banyak fitur dan konvensi yang perlu dipahami. Di awal, saya perlu waktu untuk menyesuaikan diri dengan struktur dan pola yang digunakan agar bisa mengembangkan aplikasi secara efisien.

2. Integrasi dengan Sistem atau Teknologi Lama:
Saat aplikasi baru harus berinteraksi dengan sistem lama, menyesuaikan struktur framework agar kompatibel menjadi tantangan tersendiri. Misalnya, penyesuaian format data atau API lama agar bisa bekerja dengan framework baru.

3. Keterbatasan atau Kekakuan Framework:
Beberapa framework terkadang terlalu mengikat pada aturan tertentu, sehingga sulit menerapkan logika atau desain yang berbeda dari pola bawaan framework tersebut.

4. Manajemen Dependensi dan Versi:
Kompleksitas muncul saat mengelola banyak library atau package yang saling bergantung. Konflik versi sering kali terjadi ketika melakukan pembaruan (update) framework atau dependensi.

Cara Saya Mengatasi Kompleksitas Kerangka Kerja:

1. Pelatihan dan Dokumentasi:
Saya meluangkan waktu untuk membaca dokumentasi resmi framework dan mengikuti pelatihan singkat agar memahami fitur penting yang paling sering digunakan.

2. Arsitektur Modular dan Adaptif:
Saya menggunakan pendekatan modular seperti Repository Pattern atau Dependency Injection agar aplikasi lebih fleksibel dan mudah diintegrasikan dengan sistem lain.

3. Membuat Wrapper atau Abstraction Layer:
Jika framework terasa terlalu kaku, saya membuat lapisan abstraksi untuk memisahkan logika kustom aplikasi, sehingga lebih mudah beradaptasi di masa depan.

4. Standardisasi dan Otomasi:
Saya menggunakan version control (Git) dan alat manajemen paket seperti Composer atau npm, serta menerapkan Continuous Integration/Deployment (CI/CD) agar setiap pembaruan bisa diuji otomatis dan terkontrol.

Sumber Referensi
FreeCodeCamp. (2023). How to Manage Complexity in Software Development
Fowler, M. (2018). Patterns of Enterprise Application Architecture. Addison-Wesley

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh DICKI FIRMANSYAH 043801977 - Selasa, 14 Oktober 2025, 17:06

1. Kompleksitas Kerangka Kerja yang Dihadapi

Sebagai perancang aplikasi, kompleksitas yang sering dihadapi meliputi tiga hal utama. Pertama, kompleksitas fungsional, yaitu sulitnya mengintegrasikan berbagai fitur agar berjalan sinkron. Kedua, kompleksitas interaksi pengguna, karena perbedaan kemampuan pengguna dalam memahami teknologi menuntut desain yang mudah digunakan semua kalangan. Ketiga, kompleksitas teknis, seperti penyesuaian framework dan integrasi API yang kadang menyebabkan kendala dalam pengembangan.

2. Cara Mengatasi Kompleksitas

Langkah untuk mengatasinya antara lain:

a. Merancang arsitektur aplikasi secara sistematis sebelum pengembangan.
b. Menerapkan prinsip User-Centered Design agar fokus pada kenyamanan pengguna.
c. Melakukan modularisasi supaya kesalahan mudah dilacak dan diperbaiki.
d. Melakukan pengujian berkala untuk meminimalkan kesalahan sejak dini.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh 052984686 AMELIA SYABILA - Selasa, 14 Oktober 2025, 21:08

Assalamualaikum, izin menjawab.
Nama: Amelia Syabila
NIM: 052984686

1. Jika saya adalah seorang perancang aplikasi kompleksitas kerangka kerja yang akan saya hadapi adalah adanya jurang atau jarak pemisah antara pengguna dengan sistem. Tantangan pertama adalah bagaimana caranya agar pengguna tidak bingung pas mau melakukan sesuatu. Saya harus bisa jawab pertanyaan, "Gimana ya caranya biar orang yang baru pertama kali buka aplikasi ini langsung ngerti kalau mau kirim pesan harus pencet tombol yang mana?". Kalau pengguna sampai bingung, berarti ada "jurang" antara keinginan mereka dan fitur aplikasi saya. Setelah pengguna mengklik sesuatu, pengguna harusnya langsung tau apa yang terjadi. Tantangannya adalah jangan sampai pengguna mikir, "Loh, ini datanya udah kesimpan apa belum yaa? Kok tidak ada tanda apa-apa".
   Di kepala saya sebagai perancang ada peta (model konseptual) tentang cara kerja aplikasi ini. Tapi di kepala pengguna, mereka akan membuat peta mereka sendiri (model mental) sambil mencoba-coba aplikasi saya. Tantangan terbesarnya adalah bagaimana caranya agar tampilan aplikasi (gambaran sistem) bisa jadi semacam GPS biar peta di kepala pengguna bisa sama persis dengan peta di kepala saya?.

2. Bagaimana cara saya mengatasi kompleksitas kerangka kerja tersebut?.
   Untuk menghadapi tantangan-tantangan tadi. Pertama, saya akan membangun jembatan di atas jurang antara pengguna dengan sistem. Saya akan pastikan hubungan tombol dan aksinya itu logis. Misalnya, tombol panah ke kanan itu untuk lanjut ke halaman berikutnya bukan untuk kembali. Desain elemen tombol harus kelihatan bisa ditekan, link harus kelihatan bisa di klik. Ini biasa disebut dengan affordance yang biasa beri petunjuk cara pakainya. Selalu kasih feedback setiap kali pengguna melakukan aksi, aplikasi harus kasih respon, misalnya, setelah berhasil upload foto, muncul tulisan "upload berhasil!". Ini penting agar pengguna tidak menebak-nebak. Kedua, pilih gaya desain yang tepat untuk bikin peta yang benar. Jangan bikin aplikasi yang ribet dan maksa pengguna mikir kayak komputer. Gunakan gaya dunia nyata biar gampang dipelajari, saya bisa pakai ikon-ikon yang familiar, seperti gambar tong sampah untuk menghapus atau folder untuk menyimpan dokumen. Selain itu ajarkan pengguna gerakan baru yang gampang diingat. Contohnya seperti geser (swipe) untuk menghapus, atau cubit (pinch) untuk memperbesar gambar. Sekali belajar, interaksi ini jadi sangat cepat dan efisien.

Intinya, jadi perancang itu bukan cuma soal bikin tampilan yang bagus, tapi lebih ke soal memahami cara pikir manusia. Tujuannya adalah membuat aplikasi yang terasa seperti teman yang membantu, bukan mesin yang bikin pusing.

Sumber Referensi: Materi Sesi 2

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh BALQIS AISYAH PUTRI 051860428 - Selasa, 14 Oktober 2025, 22:10

1. Mengatasi Kompleksitas Kognitif dengan Pendekatan User-Centered Design (UCD)

Menerapkan prinsip User-Centered Design yang ditekankan dalam modul IMK dengan membuat persona atau scenario dan Menerapkan metafora yang tepat misalnya "keranjang belanja" di e-commerce untuk menjembatani kesenjangan antara model pengguna dan model sistem.

2. Mengatasi Kompleksitas Fungsional dengan Penerapan Prinsip Desain Universal

Menggunakan prinsip-prinsip desain interaksi untuk menyederhanakan kerumitan.

Seperti dengan :

Hierarki Visual & Pengelompokan (Chunking): Mengatur informasi dan fungsi ke dalam kelompok-kelompok yang logis. Misalnya, semua pengaturan akun dikelompokkan dalam satu menu "Profil".
Konsistensi: Memastikan tombol, ikon, dan terminologi memiliki makna dan perilaku yang sama di seluruh aplikasi. Prinsip ini sejalan dengan "Aturan Emas" Shneiderman yang menekankan konsistensi.
Kesederhanaan: Menghilangkan elemen-elemen yang tidak relevan dan menyajikan hanya yang dibutuhkan pengguna pada saat yang tepat.
3. Mengatasi Kompleksitas Teknis melalui Kolaborasi dan Prototiping

Dengan kolaborasi Awal dengan Pengembang dan melibatkan pengembang sejak fase perancangan awal untuk memahami batasan dan peluang teknis. Bisa dengan Memahami Design System kemudian mempelajari dan menggunakan komponen dari design system kerangka kerja teknis yang dipilih (seperti Material Design) untuk memastikan desain dapat diimplementasikan secara konsisten dan efisien.

4. Mengatasi Kompleksitas Proses dengan Evaluasi Iteratif

Dengan mengadopsi siklus iteratif yang berkelanjutan seperti Prototipe -> Evaluasi -> Perbaiki , mengevaluasi heristik dan Memberikan feedback secara berkala terhadap design yang disempurnakan sebelum melalu tahap finishing.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh SELESTINUS IVAN ANGELO TETURAN 051280119 - Rabu, 15 Oktober 2025, 00:49

ijin menjawab diskusi 2

1.kompleksitas yang saya hadapi saat membuat suatu aplikasi yaitu Ketidaksesuaian antarmuka pengguna (UI) dan Keterbatasan framework.
masalah yang timbul dalam ketidasesuaian antarmuka pengguna (UI) karena Setiap framework memiliki cara kerjanya sendiri untuk mengelola komponen UI dan alurnya dan cara kerja yang terkadang tidak sesuai dengan desain ideal yang dibuat.
masalah yang timbul dalam kompleksitas keterbatasan framework yaitu framework memiliki batasan bawaan. saya sebagai perancang mungkin ingin menciptakan pengalaman unik yang tidak dapat diimplementasikan dengan mudah menggunakan fitur standar yang disediakan oleh framework, sehingga menuntut implementasi kustom yang lebih kompleks.

2. cara saya untuk mengatasi berbagai masalah kompleksitas tersebut adalah :
untuk mengatasi masalah dalam ketidaksesuaian antarmuka pengguna (UI) saya menggunakan sistem desain untuk menciptakan komponen UI yang konsisten di seluruh aplikasi. Ini membantu menjaga konsistensi visual dan fungsionalitas. dan untuk mengatasi masalah kompleksitas keterbatasan framework,saya memastikan dokumentasi yang jelas dan terperinci untuk semua komponen, pola, dan pedoman desain.

terima kasih

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh RAYHAN AULIA PUTRI 044244801 - Rabu, 15 Oktober 2025, 11:12

Ijin menjawab

1.Kompleksitas dalam pengembangan sebuah aplikasi dapat muncul dari berbagai sumber, seperti struktur kode yang rumit, terlalu banyak fitur yang tidak perlu, atau perubahan kebutuhan yang konstan.

Jenis-jenis kompleksitas :
- Kompleksitas kognitif : Terjadi akibat kode yang sulit dipahami, seperti perulangan bersarang yang berbelit-belit atau konvensi penamaan yang buruk.
- Kompleksitas struktural : Muncul dari arsitektur yang buruk, seperti ketergantungan antar komponen yang rumit, desain monolitik, atau kurangnya modularitas.
- Kompleksitas dinamis : Disebabkan oleh perubahan lingkungan dan kebutuhan proyek yang tidak terduga, yang dapat memengaruhi jadwal, anggaran, dan tumpukan teknologi.
- Kompleksitas yang tidak disengaja : Terjadi karena pilihan yang dibuat selama pengembangan, seperti penggunaan framework yang terlalu rumit atau struktur kode yang kaku.

2. Cara mengatasi kompleksitas
- Perencanaan yang matang : Lakukan analisis kebutuhan yang komprehensif sebelum memulai pengembangan untuk menghindari perubahan di kemudian hari.
- Pilih teknologi yang sesuai : Pilih framework yang tepat untuk proyek dan pertimbangkan untuk menggunakan pustaka atau komponen UI pihak ketiga yang sudah teruji untuk tugas-tugas umum seperti validasi data atau desain antarmuka.
- Gunakan desain modular : Pecah aplikasi menjadi komponen-komponen yang lebih kecil dan independen agar lebih mudah dikelola, dipahami, dan diuji.
- Terapkan prinsip-prinsip pengembangan : Gunakan prinsip seperti KISS (Keep It Simple, Stupid) dan YAGNI (You Aren't Gonna Need It) untuk menghindari penambahan fitur yang tidak perlu.
- Kelola proyek dengan efektif : Gunakan metodologi seperti Agile untuk mengelola proyek secara fleksibel dan beradaptasi dengan perubahan.
- Prioritaskan pengujian : Lakukan pengujian yang menyeluruh untuk memastikan fungsionalitas dan mengurangi potensi bug.
- Refactoring secara berkala : Sering-seringlah merapikan kode dan arsitektur untuk mencegah kompleksitas menumpuk seiring waktu.

Sumber referensi : https://www.studocu.id/id/document/universitas-terbuka/metode-statistik-i/diskusi-2-imk/73370387 , https://ozvid.com/blog/218/role-of-frameworks-in-software-development#:~:text=Kerangka%20kerja%20tersebut%20harus%20menawarkan,meningkatkan%20proses%20penge

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh ADELIA ROSA 056279433 - Rabu, 15 Oktober 2025, 20:01

1. Jika Anda adalah seorang perancang dalam pembuatan sebuah aplikasi, jelaskan kompleksitas kerangka kerja yang saudara alami/hadapi saat membuat suatu aplikasi tersebut!

Salah satu tantangan yang ada saat membuat suatu aplikasi adalah memiliki library tambahan yang tidak sesuai dengan framework yang digunakan. Hal ini menyebabkan beberapa fungsi yang ada tidak dapat berjalan dengan baik bahkan menimbulkan error. Biasanya library yang tidak sesuai merupakan versi lama yang sudah ketinggalan dan digantikan dengan versi terbarunya.

2. Jelaskan, bagaimana cara Anda mengatasi kompleksitas kerangka kerja tersebut!

Dengan kendala yang ada, hal yang dilakukan adalah menentukan framework/teknologi yang akan digunakan dari awal pembangunan. Pilih framework dengan versi terbaru dan stabil agar sistem dapat berjalan dengan lancar untuk waktu yang lama. Baca juga dokumentasi framework dan compatibility nya untuk mencari library tambahan yang sesuai dan mencegah hal yang sama terjadi kembali.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh 054312815 MUHAMMAD SYAFA'AT ABDUL GHOFUR - Rabu, 15 Oktober 2025, 20:30

Assalamualaikum awarohmatullahi Wabarokatuh Selamat Malam Salam Sejahtera ,
Sebagai seorang perancang dalam pembuatan aplikasi, saya menghadapi beberapa kompleksitas kerangka kerja. Berikut adalah beberapa kompleksitas dan cara saya mengatasinya:

1. Pemilihan Kerangka Kerja yang Tepat:

- Kompleksitas: Ada banyak sekali kerangka kerja (framework) yang tersedia, masing-masing dengan kelebihan dan kekurangan. Memilih yang paling sesuai dengan kebutuhan proyek bisa jadi rumit. Faktor-faktor yang perlu dipertimbangkan meliputi:
- Jenis aplikasi (web, mobile, desktop)
- Skala aplikasi (sederhana, kompleks)
- Keahlian tim pengembang
- Ketersediaan sumber daya dan dukungan komunitas
- Performa dan skalabilitas
- Cara Mengatasi:
- Riset Mendalam: Lakukan riset komprehensif tentang berbagai kerangka kerja yang relevan. Baca dokumentasi, tinjauan, dan studi kasus.
- Evaluasi Kebutuhan Proyek: Definisikan dengan jelas kebutuhan dan tujuan proyek. Buat daftar fitur, persyaratan performa, dan batasan anggaran.
- Uji Coba: Cobalah beberapa kerangka kerja yang menjanjikan dengan membuat prototipe sederhana. Ini membantu memahami bagaimana kerangka kerja tersebut bekerja dalam praktik.
- Konsultasi: Diskusikan pilihan dengan tim pengembang dan ahli lainnya. Minta masukan mereka tentang kerangka kerja mana yang paling cocok.

2. Kurva Pembelajaran:

- Kompleksitas: Setiap kerangka kerja memiliki sintaks, konsep, dan praktik terbaiknya sendiri. Mempelajari kerangka kerja baru membutuhkan waktu dan usaha. Tim pengembang mungkin perlu pelatihan tambahan.
- Cara Mengatasi:
- Pelatihan dan Dokumentasi: Sediakan pelatihan yang memadai bagi tim pengembang. Manfaatkan dokumentasi resmi, tutorial online, dan kursus.
- Proyek Percontohan: Mulailah dengan proyek kecil dan sederhana untuk membiasakan diri dengan kerangka kerja.
- Mentorship: Pasangkan pengembang yang berpengalaman dengan yang baru. Mentorship membantu mempercepat proses pembelajaran.
- Komunitas: Bergabunglah dengan komunitas online kerangka kerja. Forum, grup diskusi, dan saluran Slack adalah sumber daya yang berharga untuk mendapatkan bantuan dan berbagi pengetahuan.

3. Konfigurasi dan Kustomisasi:

- Kompleksitas: Kerangka kerja seringkali memerlukan konfigurasi yang rumit agar sesuai dengan kebutuhan proyek. Kustomisasi mungkin diperlukan untuk menambahkan fitur atau mengubah perilaku default.
- Cara Mengatasi:
- Dokumentasi yang Jelas: Pastikan dokumentasi kerangka kerja lengkap dan mudah dipahami.
- Alat Konfigurasi: Gunakan alat konfigurasi yang disediakan oleh kerangka kerja. Alat ini membantu menyederhanakan proses konfigurasi.
- Pola Desain: Terapkan pola desain yang baik untuk meminimalkan kompleksitas dan meningkatkan pemeliharaan kode.
- Modularitas: Rancang aplikasi secara modular sehingga komponen dapat diubah atau diganti tanpa memengaruhi bagian lain dari aplikasi.

4. Pembaruan dan Kompatibilitas:

- Kompleksitas: Kerangka kerja sering diperbarui dengan fitur baru, perbaikan bug, dan peningkatan keamanan. Memutakhirkan ke versi baru dapat menyebabkan masalah kompatibilitas dengan kode yang ada.
- Cara Mengatasi:
- Rencanakan Pembaruan: Rencanakan pembaruan kerangka kerja secara teratur. Jangan menunda pembaruan terlalu lama, karena ini dapat meningkatkan risiko keamanan.
- Uji Coba: Uji coba pembaruan di lingkungan pengujian sebelum menerapkannya ke lingkungan produksi.
- Catatan Rilis: Baca catatan rilis dengan cermat untuk memahami perubahan dan potensi masalah kompatibilitas.
- Versi Kontrol: Gunakan sistem kontrol versi (seperti Git) untuk melacak perubahan kode dan memudahkan pengembalian jika terjadi masalah.

5. Ketergantungan:

- Kompleksitas: Kerangka kerja sering bergantung pada pustaka dan komponen pihak ketiga. Mengelola ketergantungan ini bisa jadi rumit, terutama jika ada konflik versi.
- Cara Mengatasi:
- Manajer Paket: Gunakan manajer paket (seperti npm, pip, atau Maven) untuk mengelola ketergantungan. Manajer paket membantu menginstal, memperbarui, dan menghapus ketergantungan dengan mudah.
- Lingkungan Virtual: Gunakan lingkungan virtual untuk mengisolasi ketergantungan proyek. Ini mencegah konflik dengan ketergantungan proyek lain.
- Pindai Keamanan: Lakukan pemindaian keamanan secara teratur untuk mengidentifikasi kerentanan dalam ketergantungan pihak ketiga.

Dengan pendekatan yang sistematis dan terencana, kompleksitas kerangka kerja dapat dikelola dengan efektif, menghasilkan aplikasi yang berkualitas tinggi dan mudah dipelihara.
Sumber Belajar
1. Modul Interaksi Manusia Komputer , Tanggerang, Universitas Terbuka

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh MUHAMMAD LUTHFI 052480968 - Rabu, 15 Oktober 2025, 22:00

Selamat Malam.
Izin memberikan jawaban diskusi 2.

1. Kompleksitas kerangka kerja yang dihadapi saat membuat aplikasi
Dalam proses merancang sebuah aplikasi, saya menghadapi beberapa kompleksitas yang berkaitan dengan kerangka kerja (framework), yaitu:
1. Membuat struktur program lebih rapi – Framework memiliki aturan dan pola penulisan tertentu. Saya harus menyesuaikan logika program agar sesuai dengan struktur tersebut supaya program tetap rapi dan mudah dikelola.
2. Tampilan aplikasi – Tantangan muncul saat harus menyesuaikan antarmuka pengguna (user interface) agar sesuai dengan kebutuhan pengguna serta memperhatikan prinsip-prinsip dalam Interaksi Manusia dan Komputer (IMK) agar aplikasi mudah digunakan dan memberikan pengalaman pengguna yang baik.
3. Pemeliharaan aplikasi – Ketika framework diperbarui, sering kali ada perubahan fungsi yang membuat kode lama perlu disesuaikan kembali agar aplikasi tetap berjalan dengan baik.
4. Pengembangan aplikasi – Dalam menambahkan fitur baru, saya harus memastikan struktur dan tampilan tetap konsisten agar aplikasi mudah dikembangkan dan tidak menimbulkan konflik dalam sistem.

2. Cara mengatasi kompleksitas kerangka kerja yang dihadapi
Untuk mengatasi kompleksitas di atas, saya melakukan beberapa langkah, yaitu:
1. Mempelajari struktur dasar framework melalui panduan dan dokumentasi agar memahami alur kerjanya secara menyeluruh.
2. Menerapkan prinsip Interaksi Manusia dan Komputer (IMK) seperti kemudahan navigasi, konsistensi tampilan, dan kejelasan informasi untuk meningkatkan kenyamanan pengguna.
3. Menjaga pemeliharaan kode dengan membuat dokumentasi perubahan dan menggunakan versi framework yang

stabil.
4. Menulis kode secara rapi dan modular, sehingga proses pengembangan aplikasi selanjutnya menjadi lebih mudah dan efisien.

Referensi:
1. Universitas Terbuka. (2024). Modul Interaksi Manusia dan Komputer (STSI4203). Jakarta: UT Press.
2. Wahana Komputer. (2021). Pemrograman dan Pembuatan Aplikasi Berbasis Framework. Yogyakarta: Andi Offset.
3. Kurniawan, D. (2020). Rekayasa Perangkat Lunak dan Pengembangan Aplikasi. Jakarta: Mitra Wacana Media.
4. Rahardjo, B. (2022). Dasar-Dasar Pengembangan Aplikasi dan Framework. Bandung: Informatika.

Terima Kasih

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh MAZZIYAH MUTIARA YULSARIFAH 053701148 - Kamis, 16 Oktober 2025, 05:37

assalamualaikum tutor izin menjawab diskusi di atas

1. Jika saya menjadi perancang aplikasi, kompleksitas yang saya alami biasanya ada pada perancangan antarmuka (UI/UX) dan pengelolaan database.
Pada bagian UI/UX, tantangannya adalah menyesuaikan tampilan agar mudah digunakan semua pengguna. Sedangkan pada database, kompleksitas muncul saat mengatur relasi antar data agar tidak terjadi error atau duplikasi. Selain itu, koordinasi tim juga sering jadi kendala, karena setiap bagian punya tanggung jawab berbeda tapi harus saling terhubung.

2. Cara saya mengatasi kompleksitas tersebut adalah dengan membuat perencanaan dan dokumentasi kerja yang jelas, seperti flowchart, use case, dan desain sistem sebelum mulai coding. Selain itu, saya menggunakan kerangka kerja (framework) seperti Laravel atau Flutter untuk mempermudah proses pengembangan. Komunikasi rutin antar anggota tim juga penting supaya tidak ada miskomunikasi dan kesalahan desain.

Sumber refrensi:
1. Pressman, R. S. (2010). Rekayasa Perangkat Lunak: Pendekatan Praktisi. Yogyakarta: Andi.
2. Sommerville, I. (2011). Software Engineering (9th ed.). Pearson Education.
3. Kadir, A. (2014). Pemrograman Web dengan HTML, PHP, dan MySQL. Yogyakarta: Andi.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh VERALISAWATY MARBUN 056603452 - Kamis, 16 Oktober 2025, 12:01

Selamat siang Tutor dan teman teman,
Izin menjawab diskusi 2,

Setiap aplikasi memiliki struktur konseptual yang sama yaitu memiliki rancangan tujuan aplikasi, eksekusi aplikasi dan evaluasi aplikasi namun kerangka kerja dan kompleksitas yang berbeda beda.

Contohnya Kompleksitas kerangka kerja perancangan Aplikasi Pembukaan Tabungan Nasabah :
1. Aplikasi harus memastikan keamanan data nasabah terkait informasi pribadi dan keuangan nasabah
2. Aplikasi harus saling berhubungan dengan aplikasi yang lain seperti sistem laporan keuangan dan sistem laporan marketing dan sistim laporan data nasabah
3. Aplikasi harus dapat mengelola dokumen-dokumen nasabah mulai dari membaca data nasabah dan menyimpan data nasabah
4. Aplikasi harus dapat memberikan notifikasi verifikasi kepada nasabah sebagai status pembukaan rekening
5. perancangan aplikasi juga harus memperhatikan kepatuhan akan kebijakan perbankan yang berlaku
6. Aplikasi harus mudah di gunakan dan menggunakan bahasa yang mudah di pahami

Solusi untuk mengatasi Kompleksitas diatas :
1. Menggunakan Framework yang terintegrasi dengan sistim perbankan
2. Menggunakan teknologi keamanan yang canggih untuk melindungi data nasabah
3. Menggunakan proses verifikasi dan validasi yg otomatis untuk mempercepat proses pembukaan rekening
4. Memahami kepatuhan perbankan dan menerapkan aplikasi sesuai regulasi perbankan

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh NEIZA INDIRA PRATAMA 053730267 - Kamis, 16 Oktober 2025, 13:12

1.Pemilihan dan Kurva Belajar Kerangka Kerja : Kompleksitas yang Saya Alami: Saat memulai proyek, salah satu tantangan pertama adalah memilih kerangka kerja yang tepat. Ada banyak pilihan seperti React (untuk web frontend), Flutter (untuk mobile cross-platform), Django (untuk backend Python), atau Laravel (untuk backend PHP). Setiap kerangka kerja memiliki kekuatan dan kelemahan, dan pemilihan yang salah bisa menimbulkan masalah nanti. Misalnya, Kurva belajarnya bisa curam bagi pemula, karena React mengharuskan pemahaman yang dalam tentang JavaScript dan konsep fungsional seperti immutability.
2.Kompleksitas Arsitektur dan Struktur : Kompleksitas yang Saya Alami: Setelah memilih framework, tantangan berikutnya adalah mengintegrasikan elemen-elemen arsitektur. Kerangka kerja sering kali datang dengan struktur yang kaku, yang bisa membantu tapi juga membingungkan. Contoh: Di Flutter, saya harus menangani widget tree yang kompleks untuk membangun UI. Jika tidak dikelola dengan baik, ini bisa menyebabkan performa lambat atau bug sulit dilacak, seperti memory leaks.
3.Manajemen Dependensi dan Integrasi : Kompleksitas yang Saya Alami: Kerangka kerja jarang berdiri sendiri; mereka bergantung pada pustaka eksternal, library, dan tools lain. Ini bisa menjadi sumber masalah besar. misalnya, dua library yang memerlukan versi yang berbeda) bisa menyebabkan aplikasi crash atau error tak terduga.
4.Isu Performa, Keamanan, dan Skalabilitas : Kompleksitas yang Saya Alami: Saat aplikasi berkembang, kompleksitas semakin terasa di aspek ini.
Perubahan dan Maintenance : Kompleksitas yang Saya Alami: Dunia pengembangan aplikasi berubah cepat, dan framework sering kali diperbarui. Saya "mengalami" bahwa update framework bisa merusak kode lama (breaking changes), seperti yang sering terjadi di React dengan perubahan API-nya.

Cara Mengatasi Kompleksitas Kerangka Kerja:
• Memulai dengan Prototyping: Gunakan framework sederhana untuk prototype awal, lalu skalakan jika diperlukan.
• Best Practices: Ikuti prinsip seperti KISS (Keep It Simple, Stupid) dan gunakan tools seperti linter atau testing framework untuk menjaga kode tetap bersih.
• Belajar Secara Kontinu: Ikuti kursus, dokumentasi resmi, dan komunitas untuk mengatasi kurva belajar.
• Pilih Framework yang Sesuai: Pertimbangkan kebutuhan proyek; misalnya, gunakan React Native untuk aplikasi mobile cepat, atau Express untuk backend ringan.
Referensi :
React Documentation. (n.d.). React Official Website
Flutter Team. (n.d.). Flutter Documentation

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh 054118997 ALVIN CELPIN - Kamis, 16 Oktober 2025, 17:20

Nama: Alvin Celpin
NIM: 054118997

saya akan mengambil contoh dalam membangun dan merancang kerangka aplikasi e-commerce

1. Kompleksitas Kerangka Kerja dalam Perancangan Aplikasi E-commerce

Menjembatani Jarak Pemisah Eksekusi dan Evaluasi pada Proses Belanja:

- **Jarak Pemisah Eksekusi:** Kompleksitas ini muncul saat pengguna memiliki tujuan atau *gol* yang jelas, misalnya membeli sepatu lari berwarna biru, namun kesulitan dalam melakukan eksekusi pada antarmuka dimana tantangannya adalah merancang fitur pencarian dan filter yang intuitif, jika pengguna harus menebak kata kunci yang tepat atau jika opsi filter untuk warna dan ukuran tersembunyi, maka Jarak Pemisah Eksekusi menjadi lebar dan tentunya pengguna tahu apa yang mereka inginkan, tetapi sistem tidak memfasilitasi cara untuk mencapainya dengan mudah
- **Jarak Pemisah Evaluasi:** Setelah pengguna melakukan tindakan, misalnya menekan tombol Tambah ke Keranjang, kompleksitas berikutnya adalah memberikan umpan balik yang jelas jika tidak ada notifikasi visual atau perubahan pada ikon keranjang belanja, pengguna akan kesulitan mengartikan status sistem mereka tidak dapat melakukan pencocokkan hasil dengan gol semula, sehingga menimbulkan ketidakpastian dan merusak pengalaman berbelanja

- **Mengelola Fase Interaksi dalam Alur Checkout**

  - **Fase Eksekusi:** Saat pengguna mengisi formulir alamat, ini adalah langkah artikulasi, di mana pengguna menerjemahkan golnya ke dalam bahasa masukan atau teks alamat ketika mereka menekan Simpan Alamat, sistem melakukan pengerjaan dengan memvalidasi dan menyimpan data tersebut ke dalam bahasa mesin kemudian sistem melakukan penyajian dengan menampilkan kembali alamat yang telah disimpan sebagai konfirmasi dalam bahasa keluaran

- **Menyelaraskan Model Mental Toko Online dengan Model Konseptual Sistem:**
  Menurut Norman, terdapat tiga model yang harus diselaraskan seperti

  - **Model Konseptual Perancang:** Saya memahami sistem e-commerce sebagai struktur data, manajemen inventaris, dan gerbang pembayaran

  - **Model Mental Pengguna:** Pengguna memahami aplikasi ini sebagai sebuah toko digital dimana model mental mereka didasarkan pada pengalaman di toko fisik: melihat barang, memasukkan ke keranjang, dan membayar di kasir

  - Kompleksitasnya adalah merancang **gambaran sistem** (antarmuka web) yang sesuai dengan model mental pengguna, sebagai contoh menampilkan status Stok 0 kurang efektif dibandingkan Produk Habis Terjual istilah ini yang sesuai dengan model mental toko

2. Cara Mengatasi Kompleksitas Tersebut dalam Aplikasi E-commerce

Meminimalkan Jarak Pemisah dengan Desain yang Terarah:

- **Menerapkan Pemetaan (Mapping) yang Logis:** merancang hubungan yang intuitif antara kontrol dan fungsinya Contohnya, tombol Lanjutkan ke Pembayaran akan ditempatkan di bagian paling akhir dari halaman keranjang belanja, setelah rincian total harga dimana hal Ini adalah pemetaan yang benar karena secara logis pengguna akan membayar setelah selesai memeriksa pesanannya

- **Menggunakan Affordance:** Setiap elemen harus secara visual mengindikasikan fungsinya contohnya Tombol Beli Sekarang akan didesain dengan warna yang mencolok dan efek bayangan supaya terlihat menonjol dan jelas untuk diklik dan juga Kolom untuk memasukkan kode voucher akan memiliki batas yang jelas, memberikan *affordance* untuk diisi dengan teks

Memilih Paradigma Perancangan yang Tepat untuk E-commerce:

- **Menghindari Paradigma Berpusat Pada Implementasi:** Pendekatan ini secara khusus harus dihindari dimana jika terjadi kegagalan pembayaran, sistem tidak boleh menampilkan pesan error teknis seperti Error 503: Payment Gateway Unreachable dan sebaliknya pesan yang ditampilkan harus berorientasi pada pengguna, seperti Terjadi gangguan pada sistem pembayaran ilakan coba lagi beberapa saat jadi Pengguna tidak perlu dan tidak ingin tahu cara kerja sistem di belakang layar
- **Mengadopsi Paradigma Idiomatik:**
  Daripada terpaku pada metafora dunia nyata yang terbatas, lebih baik memprioritaskan perancangan idiomatik yang mengandalkan idiom visual dalam berbentuk ikon yang telah dipelajari pengguna Contoh: Ikon kaca

pembesar untuk fungsi pencarian, ikon hati (♡) untuk Wishlist atau favorite, dan ikon tiga garis (≡) untuk menu navigasi ataupun menu adalah contoh idiom yang efektif jadi Pengguna modern telah mempelajari makna dari ikon-ikon ini secara alamiah tanpa perlu penjelasan, sehingga ini menjadi pendekatan yang lebih fleksibel dan efisien untuk antarmuka ecommerce

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh WANAWATI RARAS DINA AULIYA 052186543 - Jumat, 17 Oktober 2025, 11:16

Assalamu'alaikum wr.wb. Selamat siang Bapak dosen pengampu dan teman teman diskusi, izin menjawab soal diskusi 2 diatas.

Menurut saya, ketika menjadi seorang perancang aplikasi, kompleksitas utama dalam kerangka kerja interaksi manusia dan komputer terletak pada menjembatani perbedaan antara cara berpikir pengguna dan cara kerja sistem. Berdasarkan Materi Inisiasi Sesi 2 – Kerangka Kerja dan Paradigma Interaksi, kerangka kerja interaksi melibatkan empat komponen utama yaitu pengguna, sistem, masukan, dan keluaran, yang saling berhubungan melalui dua fase utama: fase eksekusi dan fase evaluasi. Dalam praktiknya, kompleksitas muncul karena sulitnya menciptakan antarmuka yang mudah dipahami pengguna sekaligus efisien secara teknis.

Salah satu tantangan terbesar adalah “jarak pemisah eksekusi dan evaluasi” (execution & evaluation gap). Misalnya, pengguna memiliki tujuan tertentu (goal) dalam aplikasi, tetapi fitur atau navigasi yang disediakan tidak sesuai dengan ekspektasinya. Hal ini menyebabkan frustrasi karena pengguna kesulitan menemukan atau memahami fungsi sistem. Selain itu, perbedaan model mental antara perancang dan pengguna sering menimbulkan masalah. Pengguna mungkin berasumsi aplikasi bekerja dengan cara tertentu, padahal logika sistem berbeda, sehingga menyebabkan kebingungan.

Untuk mengatasi kompleksitas ini, menurut saya ada beberapa langkah yang perlu dilakukan. Pertama, memahami model mental pengguna melalui riset pengguna (user research) agar rancangan antarmuka sesuai dengan cara berpikir dan kebutuhan mereka. Kedua, meminimalkan jarak eksekusi dan evaluasi dengan membuat desain antarmuka yang sederhana, menggunakan ikon intuitif, dan menyediakan umpan balik visual yang jelas setelah pengguna melakukan tindakan. Ketiga, menerapkan paradigma perancangan idiomatik, seperti yang dijelaskan pada materi, yaitu membuat antarmuka yang memanfaatkan idiom atau kebiasaan pengguna dalam interaksi sehari-hari agar lebih mudah dipelajari tanpa perlu memahami mekanisme sistem secara teknis.

Jadi, menurut saya kompleksitas dalam kerangka kerja interaksi dapat diatasi dengan mengutamakan pemahaman terhadap pengguna, membuat antarmuka yang konsisten dan mudah dipelajari, serta memastikan setiap interaksi memberikan umpan balik yang jelas. Dengan begitu, pengalaman pengguna akan menjadi lebih efektif dan efisien.

Sekian hasil diskusi saya, jika teman-teman memiliki tambahan, saran, atau tanggapan, silakan dituliskan agar diskusi ini bermanfaat bagi kita semua. Terimakasih, wassalamu'alaikum wr.wb.

Referensi:
- Materi Sesi 2 – Kerangka Kerja dan Paradigma Interaksi. Program Studi Sistem Informasi, Fakultas Sains dan Teknologi, Universitas Terbuka.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh 054130385 IVAN MAHENDRA YULIANTO - Jumat, 17 Oktober 2025, 13:45

Izin Menjawab diskusi 2
1. Kompleksitas kerangka kerja yang dihadapi dalam pembuatan aplikasi
Sebagai seorang perancang aplikasi, kompleksitas utama yang saya hadapi biasanya terletak pada pemilihan dan

integrasi kerangka kerja (framework) yang sesuai dengan kebutuhan sistem. Setiap framework memiliki kelebihan dan keterbatasan tersendiri. Misalnya, ketika menggunakan React.js untuk front-end dan Laravel untuk back-end, saya harus memastikan keduanya dapat berkomunikasi dengan baik melalui API.
Selain itu, kompleksitas lain muncul dalam pengelolaan arsitektur aplikasi — seperti pengaturan state management, keamanan data, serta penyesuaian tampilan agar responsif di berbagai perangkat. Proses debugging juga menjadi tantangan tersendiri karena seringkali bug muncul akibat ketidaksesuaian versi library atau dependensi framework yang digunakan.
2. Cara mengatasi kompleksitas kerangka kerja:
- Memahami dokumentasi resmi dari framework yang digunakan sebelum memulai proyek. Hal ini membantu saya mengetahui batasan dan fitur yang tersedia.
- Membagi pekerjaan berdasarkan modul atau komponen agar lebih mudah dalam pengelolaan dan pengujian. Dengan pendekatan modular, setiap bagian dapat dikembangkan dan diuji secara terpisah.
- Menggunakan paradigma desain yang jelas, seperti Model-View-Controller (MVC) atau Component-Based Architecture, agar alur kerja aplikasi terstruktur dan mudah dikelola.
- Melakukan kolaborasi dan code review dengan rekan satu tim untuk memastikan standar kode tetap konsisten dan kesalahan dapat diminimalkan.
- Memanfaatkan tools otomatisasi seperti Git, Docker, atau CI/CD pipeline untuk mempermudah proses pengembangan dan deployment.

Dengan pendekatan tersebut, kompleksitas framework dapat dikelola dengan lebih efektif sehingga proses pengembangan menjadi lebih efisien dan hasil aplikasi lebih stabil.

Referensi:
- Sommerville, Ian. Software Engineering (10th Edition). Pearson Education, 2015.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh 053940079 RYAN SOFIYAN - Jumat, 17 Oktober 2025, 15:00

Bismillah,

1. Kompleksitas Kerangka Kerja yang Dihadapi Perancang
Sebagai seorang perancang aplikasi, tantangan utama yang saya hadapi adalah kompleksitas dalam menghubungkan cara berpikir pengguna dengan cara kerja sistem. Dalam proses ini, saya harus memahami kerangka kerja interaksi yang melibatkan beberapa komponen: pengguna, masukan, sistem, dan keluaran. Tantangan muncul karena pengguna sering memiliki model mental yang berbeda dari cara sistem sebenarnya bekerja, sehingga desain harus mampu menjembatani perbedaan ini.

Selain itu, ada juga tantangan jarak eksekusi dan evaluasi, yaitu kesenjangan antara apa yang diinginkan pengguna dengan apa yang sistem bisa lakukan, serta kesulitan pengguna dalam memahami hasil yang ditampilkan. Hal-hal inilah yang membuat proses perancangan menjadi kompleks dan perlu perencanaan matang

2. Cara Mengatasi Kompleksitas Kerangka Kerja
Untuk mengatasi kompleksitas tersebut, langkah pertama yang saya lakukan adalah memahami kebutuhan dan cara berpikir pengguna sejak awal. Hal ini membantu saya menyusun desain antarmuka yang mudah dipahami dan digunakan.

Selanjutnya, saya meminimalkan jarak eksekusi dan evaluasi dengan memastikan tampilan dan fungsi sistem jelas, intuitif, dan sesuai harapan pengguna. Saya juga menggunakan pendekatan uji coba antarmuka (user testing) agar bisa memperbaiki desain berdasarkan umpan balik nyata dari pengguna.

Terakhir, saya menerapkan prinsip pemetaan yang baik dan affordance, sehingga pengguna dapat langsung memahami fungsi dari setiap elemen tampilan tanpa harus berpikir terlalu keras.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh DEA MAILA PUTRI 053579123 - Sabtu, 18 Oktober 2025, 12:05

Izin Menanggapi Diskusi 2 : Kompleksitas dalam Kerangka Kerja Pengembangan sebuah aplikasi

1. Kompleksitas Kerangka Kerja (Framework)
Kompleksitas dalam pengembangan kerangka kerja (framework) adalah tantangan umum yang muncul dari interaksi dan ketergantungan berbagai teknologi, pustaka (library), dan arsitektur (Gamma et al., 1994).
• Penyebab Kompleksitas:
- Kurva Pembelajaran: Framework modern seringkali memiliki dokumentasi ekstensif dan membutuhkan waktu serta pengalaman untuk menguasai nuansa dan praktik terbaiknya.
- Integrasi dan Dependensi: Mengelola kompatibilitas berbagai komponen. dan dependensi dapat menimbulkan konflik, memerlukan versi yang tepat, dan menghabiskan waktu dan sumber daya.
- Arsitektur dan Desain: Pemilihan arsitektur yang tepat (misalnya, MVC, MVVM) dan pengimplementasiannya membutuhkan pemahaman mendalam tentang prinsip desain dan pola arsitektur.
- Pengujian dan Debugging: Menguji dan memperbaiki bug di kode yang tersebar di berbagai modul dan lapisan dapat membutuhkan alat dan teknik khusus.

2. Strategi Mengatasi Kompleksitas
Mengatasi kompleksitas memerlukan pendekatan yang terencana dan sistematis:
- Investasi Pembelajaran: Memahami prinsip-prinsip dasar framework melalui tutorial dan dokumentasi dapat mempercepat kurva pembelajaran.
- Pengelolaan Dependensi: Menggunakan alat manajemen dependensi untuk memastikan kompatibilitas dan versi yang tepat.
- Prinsip Desain: Menerapkan prinsip desain dan pola arsitektur yang kuat. untuk menghasilkan kode yang lebih

modular, mudah dipahami, dan dipelihara (maintainable).
- Alat Pengujian (Testing Tools): Menggunakan alat pengujian dan debugging yang tepat untuk mengidentifikasi dan memperbaiki bug dengan cepat.
Kolaborasi dan Komunikasi: Kolaborasi tim yang efektif, termasuk berbagi pengetahuan, memastikan semua anggota tim berada dalam pemahaman yang sama.

Sumber Referensi :
Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). John Wiley & Sons.
Tanenbaum, A. S., & Wetherall, D. J. (2021). Computer Networks (6th ed.). Pearson Education.
Walpole, R. E., Myers, R. H., Myers, S. L., & Ye, K. (2012). Probability and Statistics for Engineers and Scientists (9th ed.). Pearson Education.
Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.
McConnell, S. (2004). Code Complete (2nd ed.). Microsoft Press.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh 051175974 EROSS ARIFUDDIN FADLI - Sabtu, 18 Oktober 2025, 18:05

Dear, Bapak/Ibu Tutor & Rekan-rekan Mahasiswa/i,

Izin menanggapi topik diskusi 2 sebagai berikut;

1. Jika Anda adalah seorang perancang dalam pembuatan sebuah aplikasi, jelaskan kompleksitas kerangka kerja yang saudara alami/hadapi saat membuat suatu aplikasi tersebut!

Sebagai seorang perancang, kerangka kerja (framework) pada dasarnya adalah struktur yang saya gunakan untuk mengkonseptualisasikan sistem aplikasi secara menyeluruh. Kompleksitas utama yang saya hadapi dalam kerangka kerja interaksi ini adalah mengelola komunikasi dan "penerjemahan" yang harus terjadi antara empat komponen utamanya.

Kerangka kerja interaksi memiliki empat komponen:

1. Pengguna (P): Memiliki gol dan menggunakan "bahasa tugas" (atribut psikologis).
2. Sistem (S): Beroperasi menggunakan "bahasa mesin" (atribut komputasi).
3. Masukan (M): Bahasa yang digunakan untuk memasukkan perintah.
4. Keluaran (K): Bahasa yang digunakan untuk menampilkan hasil.

Kompleksitas muncul dalam dua fase utama yang harus dirancang dengan mulus:

A. Fase Eksekusi

Saya harus merancang alur agar keinginan pengguna (bahasa tugas) dapat diterjemahkan dengan benar. Ini melibatkan tiga langkah:

1. Artikulasi: Pengguna memformulasikan golnya ke dalam bahasa masukan.
2. Pengerjaan: Sistem menerjemahkan bahasa masukan itu ke bahasa mesin.
3. Penyajian: Sistem menyajikan hasil operasinya (dari bahasa mesin) ke bahasa keluaran.

B. Fase Evaluasi

Saya harus memastikan pengguna dapat memahami apa yang terjadi. Ini melibatkan langkah Observasi, di mana pengguna mengartikan bahasa keluaran di layar dan mencocokkannya dengan gol awal mereka.

Tantangan terbesar (kompleksitas) saya adalah menjembatani kesenjangan antara apa yang diinginkan pengguna (di P) dan apa yang dilakukan sistem (di S), serta memastikan bahasa masukan (M) dan keluaran (K) seintuitif mungkin.

2. Jelaskan, bagaimana cara Anda mengatasi kompleksitas kerangka kerja tersebut!

Untuk mengatasi kompleksitas tersebut, fokus saya adalah meminimalkan jarak antara pengguna dan sistem. Modul ini menyediakan beberapa konsep kunci untuk mencapainya:

A. Meminimalkan "Jarak Pemisah" (Gulfs)

- Jarak Pemisah Eksekusi: Saya harus mengurangi perbedaan antara keinginan pengguna dan apa yang bisa dilakukan oleh aplikasi. Jika aplikasi saya tidak mendukung tindakan yang ingin dilakukan pengguna, "jarak" ini melebar dan pengguna akan frustrasi.
- Jarak Pemisah Evaluasi: Saya harus meminimalkan tingkat kesulitan pengguna dalam mengartikan status sistem. Aplikasi harus memberikan umpan balik yang jelas sehingga pengguna tahu apa yang sedang terjadi.

B. Menyelaraskan Model Mental dan Model Konseptual

- Sebagai perancang, saya memiliki Model Konseptual (model saya tentang bagaimana sistem dirancang).
- Pengguna akan mengembangkan Model Mental mereka sendiri (pemahaman mereka tentang cara kerja sistem) saat berinteraksi.
- Tugas saya adalah menciptakan Gambaran Sistem (antarmuka, dokumentasi, dll.) yang sejelas mungkin. Tujuannya adalah agar Model Mental pengguna yang terbentuk menjadi "cukup dekat" dengan Model Konseptual saya. Jika berhasil, pengguna dapat menggunakan aplikasi tanpa kesulitan.

C. Memilih Paradigma Antarmuka yang Tepat

- Saya akan menghindari antarmuka Berpusat-Pada-Implementasi (BPI), karena pengguna tidak peduli bagaimana sistem bekerja; mereka hanya ingin tugasnya selesai.
- Saya bisa menggunakan Antarmuka Metaforik (menggunakan gambar/simbol intuitif) , tetapi saya harus hati-hati karena metafora memiliki keterbatasan, seperti sulit diskalakan untuk proses yang kompleks.
- Cara terbaik, menurut modul, adalah menggunakan Antarmuka Idiomatik. Paradigma ini tidak berfokus pada pengetahuan teknis atau metafora. Sebaliknya, saya akan merancang idiom visual dan perilaku sederhana yang mudah dipelajari pengguna. Ini memanfaatkan kemampuan alami manusia untuk belajar dengan cepat tanpa perlu memahami "bagaimana" dan "mengapa" sistem bekerja.

Sumber Referensi:

Santosa, P. I. (2025). *Interaksi manusia dan komputer* (MSIM4208). Universitas Terbuka.

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh 054595944 MUHAMAD YAHYA AYASH - Minggu, 19 Oktober 2025, 12:45

Assalamualaikum Wr. Wb
Yth. Dosen Tuton Interaksi Manusia dan Komputer
Bapak Arvin Claudy Frobenius

Selamat Siang Pak Arvin.
Mohon izin untuk menjawab soal dari diskusi 2.

Nomor 1.
Jika saya menjadi seorang perancang dalam pembuatan aplikasi, maka kemungkinan kompleksitas kerangka kerja yang saya alami adalah
1. Konsep/blueprint yang kurang matang
Karena kurangnya pengalaman dan ilmu pengetahuan memungkinan konsep yang sudah dibuat kurang matang.

2. Komunikasi antara tim dan klien yang masih belum solid
Di awal percobaan pembuatan aplikasi, pasti akan menemukan kesulitan bagaimana cara merealisasikan apa yang ada di pikiran, maupun ketika bekerja sama dengan customer.

3. Penambahan fitur diluar kontrak kerja

Ada istilah customer adalah raja, namun ini bisa menjadi kendala pada pembuatan aplikasi terlebih pada customer yang ingin menambahkan fitur ataupun keinginan yang berubah ubah.

Nomor 2.
Untuk mengatasi 3 point Utama kompleksitas kerangka kerja berikut cara mengatasinya :
1. Dengan memakai model mental. Yaitu memahami bagaimana cara system bekerja, memahami gambaran sistem yang akan dibuat dan memahami cara sistem berinteraksi dengan pengguna. Dengan memahami hal terebut, maka saya sebagai pembuat aplikasi bisa lebih mudah untuk merealisasikan keinginan user dan membuat aplikasi yang user friendly.

2. Selain dari sisi teknis, penyelesaian kendala tersebut dari sisi komunikasi dengan cara membuat komunikasi yang lebih efisien namun intens, dan juga menyamakan persepsi antar tim agar semua tim tahu tujuan bersama

3. Membuat kontrak kerja yang jelas dan disetujui kedua belah piahk saat awal proyek pembuatan aplikasi untuk menghindari tambahan kerja diluar kontrak kerja.

Referensi :
1.https://www.gamatechno.com/resources/11-permasalahan-yang-terjadi-saat-proses-pengembangan-software/
Diakses pada 19 Oktober 2025
2. Santosa.P.I, (2025), Interaksi Manusia dan Komputer, Tangerang Selatan, Universitas Terbuka, Hal : 3.15-3.23

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2
oleh MUHAMMAD ALI MUHSIN 044061498 - Minggu, 19 Oktober 2025, 16:21

Dalam proses pembuatan aplikasi, seorang perancang biasanya menghadapi sejumlah kompleksitas kerangka kerja yang cukup menantang. Kompleksitas ini dapat muncul dari berbagai aspek seperti struktur kode, desain antarmuka pengguna, proses pengujian dan integrasi, hingga pemilihan teknologi yang sesuai

1.Kompleksitas dalam Kerangka Kerja Aplikasi

Salah satu tantangan utama adalah refaktorisasi kode untuk meningkatkan desain dan kejelasan kode tanpa mengubah perilaku eksternal, yang sering kali memakan waktu dan membutuhkan pemahaman mendalam tentang kode yang sudah ada. Selain itu, menjaga prinsip clean code agar kode tetap mudah dipahami, dipelihara, dan mampu dikembangkan lebih lanjut juga merupakan tantangan tersendiri.

Desain antarmuka pengguna merupakan aspek penting lain yang memerlukan proses iteratif berdasarkan umpan balik pengguna agar terlihat intuitif dan mudah digunakan. Tidak kalah penting, proses penerapan, seperti otomatisasi CI/CD, memerlukan pengaturan alat dan infrastruktur yang kompleks, tetapi sangat penting untuk rilis yang cepat dan handal.

2.Mengatasi Kompleksitas Kerangka Kerja

Dalam menanggulangi kompleksitas tersebut, beberapa strategi yang umum dilakukan adalah:

- **Perencanaan dan analisis kebutuhan yang mendalam**: Memahami kebutuhan pengguna secara lengkap sehingga desain awal sudah matang dan memenuhi harapan.
- **Pemilihan teknologi yang tepat**: Memilih bahasa pemrograman, framework, dan alat yang sesuai sehingga mengurangi kerumitan pengembangan dan meningkatkan efisiensi kerja.
- **Desain modular**: Menggunakan pendekatan modular untuk memecah aplikasi menjadi bagian-bagian kecil agar lebih mudah dikembangkan dan diuji secara terpisah.
- **Implementasi prinsip pengembangan perangkat lunak yang baik**: Prinsip seperti SOLID dan DRY digunakan untuk menjaga kode tetap bersih dan mudah dipelihara.

Para perancang juga perlu senantiasa beradaptasi terhadap perkembangan teknologi dan metodologi pengembangan seperti agile, yang membantu memfokuskan pada iterasi cepat dan kolaborasi tim untuk mengurangi risiko dan kompleksitas.

Kesimpulan

Mengelola kompleksitas kerangka kerja dalam pengembangan aplikasi memerlukan perencanaan yang matang, pemilihan teknologi yang bijaksana, serta penerapan prinsip desain dan pengembangan yang baik. Pendekatan ini membantu memastikan aplikasi yang dikembangkan tidak hanya memenuhi kebutuhan fungsional tetapi juga mudah dikelola dan dikembangkan di masa depan.

Sumber referensi:

Santosa, P. I. (2021). Interaksi Manusia dan Komputer (Edisi 1). Tangerang Selatan : Universitas Terbuka.

https://repository.unikom.ac.id/49098/1/Kerangka%20Kerja%20dan%20Paradigma%20Interaksi.pdf

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh 053896466 HANIF AZHAR UTOMO - Minggu, 19 Oktober 2025, 22:02

1. Jika saya diibaratkan sebagai seorang perancang aplikasi, maka saya akan sering berhadapan dengan kompleksitas kerangka kerja (framework) yang signifikan. Seperti ketika harus mengintegrasikan paradigma interaksi manusia dan komputer (HCI), atau desain berbasis pengguna (user-centered design) dan paradigma GUI (Graphical User Interface) dengan kebutuhan teknis aplikasi. Misalnya saja dalam pengembangan aplikasi mobile seperti e-commerce, saya menggunakan framework seperti React Native untuk membangun antarmuka yang responsif.
Maka kompleksitas yang saya alami yaitu meliputi banyaknya komponen dan modul yang harus diintegrasikan agar aplikasi berjalan sesuai tujuan. Kerangka kerja sering memiliki banyak lapisan abstraksi, aturan, dan juga konvensi yang harus dipatuhi. Selain itu kompleksitas juga muncul dari kebutuhan untuk memastikan aplikasi responsif, mudah digunakan, serta kompatibel dengan berbagai perangkat dan platform. Kemudian menangani berbagai paradigma interaksi pengguna, serta memastikan keamanan dan performa yang optimal juga menambah tingkat kesulitan dalam menggunakan kerangka kerja tersebut. Kompleksitas ini bisa membuat proses pengembangan menjadi sedikit lebih lama dan menantang, karena perancang harus memahami dan mengelola banyak aspek teknis dan sekaligus menjaga kualitas pengalaman pengguna.

2. Dalam mengatasi berbagai hal kompleksitas tersebut ada beberapa pendekatan yang dapat saya akan lakukan seperti :
- Melakukan pemahaman mendalam terhadap kerangka kerja dengan mengikuti tata cara dokumentasi, tutorial, serta contoh implementasinya untuk menguasai fitur-fitur utama.
- Selanjutnya menerapkan prinsip desain modular yaitu memecah sistem menjadi modul-modul kecil dan komponen agar aplikasi dapat dikembangkan serta diuji secara terpisah sehingga mempermudah dalam pemeliharaan.
- Menggunakan alat bantu manajemen proyek dan kolaborasi dengan tim agar pengembangan dapat berjalan secara terorganisir dengan baik.
- Menerapkan prototyping dan iterasi dalam pengembangan untuk terus menguji dan memperbaiki desain interaksi dengan pengguna.
- Dan yang terakhir selalu mengikuti perkembangan terbaru dari kerangka kerja agar dapat memanfaatkan fitur baru yang mungkin bisa mengurangi kompleksitas secara signifikan.

Sumber Referensi :
-Ir. Paulus Insap Santosa, M.Sc., Ph.D., I.P.U., November 2021, “Kerangka Kerja dan Paradigma Interaksi” Penerbit Universitas Terbuka Tangerang Selatan. (Modul 3 Halaman 3.4 - 3.11, dan 3.15 - 3.33)
-https://www.nngroup.com/articles/complex-application-design-framework/
-https://repository.unikom.ac.id/49098/1/Kerangka%20Kerja%20dan%20Paradigma%20Interaksi.pdf
-https://appmaster.io/id/blog/kerangka-kerja-pengembangan-perangkat-lunak-yang-efektif

Tautan permanen Tampilkan induk Balas

Re: Diskusi.2

oleh ANGGO ABRIANSYAH 056179707 - Senin, 20 Oktober 2025, 13:45

Hide sidebars

Course dashboard

Dengan Hormat,

Dengan ini saya ingin menyampakan diskusi tetang materi tentang Kerangka Kerja dan Paradigma Interaksi yaitu sebagai berukut

1.seorang perancang dalam pembuatan sebuah aplikasi, jelaskan kompleksitas kerangka kerja yang saudara alami/hadapi saat membuat suatu aplikasi tersebut

1. Pemilihan Kerangka Kerja yang Tepat
Salah satu tantangan awal dalam merancang aplikasi adalah memilih kerangka kerja (framework) yang paling sesuai. Keputusan ini tidak hanya berdasarkan preferensi pribadi, tetapi juga harus mempertimbangkan kebutuhan proyek, kompatibilitas dengan teknologi lain, dukungan komunitas, dokumentasi, serta kemampuan framework untuk menangani skala besar. Misalnya, React memberikan fleksibilitas tinggi namun membutuhkan banyak keputusan arsitektur, sementara Angular menawarkan struktur yang lebih terpadu namun bisa terasa berat bagi sebagian pengembang.

2. Integrasi Antar Komponen
Aplikasi modern biasanya terdiri dari berbagai komponen yang saling terhubung, seperti frontend, backend, database, dan API eksternal. Kompleksitas muncul ketika semua komponen ini harus berkomunikasi secara lancar, aman, dan efisien. Selain itu, pengembang juga harus mengelola fitur tambahan seperti autentikasi, otorisasi, caching, dan logging, yang semuanya membutuhkan perhatian khusus dalam integrasi.

3. Keamanan Aplikasi
Meskipun banyak framework menyediakan fitur keamanan bawaan, pengembang tetap harus memastikan bahwa aplikasi terlindungi dari berbagai ancaman seperti XSS, CSRF, dan SQL Injection. Validasi input, pengelolaan token, serta manajemen sesi pengguna menjadi aspek penting yang harus dirancang dengan hati-hati agar tidak menimbulkan celah keamanan.

4. Responsivitas dan Kompatibilitas
Aplikasi harus mampu berjalan dengan baik di berbagai perangkat dan browser, mulai dari desktop hingga mobile, serta mendukung berbagai resolusi layar. Framework seperti Bootstrap atau Tailwind CSS memang membantu dalam hal ini, namun tetap diperlukan penyesuaian manual agar tampilan dan fungsionalitas aplikasi tetap optimal di semua platform.

5. Manajemen State dan Data
Dalam aplikasi yang kompleks, pengelolaan state atau data antar komponen menjadi tantangan tersendiri. Penggunaan alat seperti Redux, Vuex, atau Context API membantu dalam sinkronisasi data, namun tetap membutuhkan pemahaman mendalam agar tidak menimbulkan masalah performa atau inkonsistensi data.

6. Deployment dan CI/CD
Mengintegrasikan framework dengan proses deployment modern seperti Docker, GitHub Actions, atau Jenkins juga menambah kompleksitas. Pengembang harus memastikan bahwa aplikasi dapat berjalan dengan baik di berbagai lingkungan (staging dan production), serta memiliki sistem monitoring dan rollback yang handal jika terjadi kesalahan saat peluncuran.

7.Dokumentasi dan Maintainability
Framework yang canggih sekalipun akan sulit digunakan jika tidak didokumentasikan dengan baik. Dokumentasi yang jelas dan struktur kode yang rapi sangat penting agar tim pengembang lain dapat memahami, memelihara, dan mengembangkan aplikasi dengan efisien. Tanpa dokumentasi yang baik, proses debugging dan penambahan fitur bisa menjadi sangat rumit.

Sumber :
https://idcloudhost.com/blog/apa-itu-ci-dan-cd/
https://vibrant.web.id/state-management-di-frontend-redux-zustand-atau-context-api/
https://www.ibm.com/id-id/think/topics/application-security
https://applicanity.com/post/rahasia-membangun-aplikasi-dengan-kompleksitas-tinggi-tantangan-dan-strategi
https://www.lawencon.com/mengenal-framework-fungsi-tipe-jenis-dan-faktor-dalam-memilih/

2.Jelaskan, bagaimana cara Anda mengatasi kompleksitas kerangka kerja tersebut!

1. Analisis Kebutuhan dan Pemilihan Framework Secara Bijak
Langkah pertama adalah memahami kebutuhan aplikasi secara menyeluruh—baik dari sisi fungsionalitas, performa, maupun skalabilitas. Berdasarkan analisis tersebut, saya memilih framework yang paling sesuai. Misalnya, jika aplikasi membutuhkan interaktivitas tinggi di sisi frontend, saya cenderung memilih React atau Vue. Untuk backend yang membutuhkan struktur MVC dan keamanan bawaan, saya bisa memilih Laravel atau Django. Pemilihan ini juga mempertimbangkan dokumentasi, komunitas, dan pengalaman tim.

2. Desain Arsitektur Modular dan Terintegrasi
Untuk mengatasi kompleksitas integrasi antar komponen, saya merancang arsitektur aplikasi secara modular. Setiap komponen (frontend, backend, database, API) memiliki tanggung jawab yang jelas dan berkomunikasi melalui protokol standar seperti REST atau GraphQL. Saya juga menggunakan tools seperti Postman untuk menguji integrasi dan memastikan komunikasi antar bagian berjalan lancar.

3. Implementasi Keamanan Sejak Awal (Security by Design)
Keamanan tidak saya anggap sebagai tambahan, melainkan bagian inti dari desain aplikasi. Saya menerapkan validasi input di sisi frontend dan backend, menggunakan HTTPS, serta menerapkan autentikasi dan otorisasi berbasis token (JWT atau OAuth). Saya juga rutin melakukan audit keamanan dan menggunakan tools seperti OWASP ZAP untuk mendeteksi potensi kerentanan.

4. Penggunaan Framework Responsif dan Pengujian Multi-Platform
Untuk memastikan aplikasi responsif dan kompatibel di berbagai perangkat, saya menggunakan framework CSS seperti Tailwind atau Bootstrap. Saya juga melakukan pengujian lintas perangkat dan browser menggunakan tools seperti BrowserStack, serta menerapkan prinsip desain mobile-first agar tampilan tetap optimal di layar kecil.

5. Manajemen State yang Efisien
Dalam aplikasi yang kompleks, saya menggunakan library seperti Redux atau Vuex untuk mengelola state secara terpusat. Saya juga menerapkan prinsip reaktivitas dan memoization untuk menghindari rerender yang tidak perlu, serta memisahkan state global dan lokal agar lebih mudah dikelola.

6. Otomatisasi Deployment dengan CI/CD
Untuk mengurangi risiko kesalahan saat deployment, saya menggunakan pipeline CI/CD seperti GitHub Actions atau GitLab CI. Proses ini mencakup build otomatis, pengujian unit dan integrasi, serta deployment ke staging dan production. Saya juga menggunakan Docker untuk memastikan konsistensi lingkungan pengembangan dan produksi.

7. Dokumentasi dan Standar Kode yang Konsisten
Saya selalu menekankan pentingnya dokumentasi, baik dalam bentuk README, komentar kode, maupun dokumentasi API (misalnya dengan Swagger). Selain itu, saya menerapkan standar penulisan kode (linting dan formatting) agar kode mudah dibaca dan dipelihara oleh tim lain. Dokumentasi juga saya integrasikan dalam proses onboarding developer baru.
Sumber :
https://www.dikasihinfo.com/pendidikanhttps://vibrant.web.id/state-management-di-frontend-redux-zustand-atau-context-api/https://www.codepolitan.com/blog/pentingnya-dokumentasi-dan-cara-efektif-untuk-memahami-

dokumentasi-kode/

Demikian informasi yang dapat saya sampaikan. Atas perhatiannya saya ucapkan terima kasih.

Hormat Saya

Tautan permanen Tampilkan induk Balas

Hide sidebars
Course dashboard

◀ Materi Pengayaan Se...

Lompat ke...

Kehadiran Sesi ke-3 ▶

Navigasi

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

Administrasi

- Forum administrasi
  - Berlangganan dinonaktifkan