TUGAS RESUME

Mata Kuliah : Rekayasa dan Kebutuhan Perangkat Lunak

Kelas : A11. 4604

Anggota Kelompok : - Z11.2023.00130 Farel Putra Albana

- Z11.2023.00143 Rovy Saputra Nugeraha

- Z11.2023.00180 Joseph Dani Christian Fallo

Analisis Dampak Perubahan (*Strategies for Impact Analysis*)

6.2 Strategi Analisis Dampak

Strategi-strategi untuk analisis dampak dapat dibagi menjadi dua kategori utama: *strategi yang dapat diotomatisasi* dan *strategi manual*. Strategi yang dapat diotomatisasi meliputi analisis Penelusuran atau ketergantungan dan teknik pemotongan, yang memerlukan infrastruktur yang terperinci namun dapat menghasilkan estimasi dampak yang sangat rinci meskipun seringkali menghasilkan banyak positif palsu. Sebaliknya, strategi manual seperti konsultasi spesifikasi desain dan wawancara dengan pengembang berpengalaman memerlukan infrastruktur yang lebih sedikit dan lebih mudah diadopsi, meskipun estimasi dampaknya cenderung lebih kasar. Penelitian sebelumnya menunjukkan bahwa analisis dampak oleh pengembang seringkali terlalu optimis, sehingga penting untuk juga mempertimbangkan prediksi konservatif yang lebih realistis. Idealnya, pendekatan analisis dampak akan memberikan estimasi optimis dan konservatif, dan melalui pengumpulan dan analisis data empiris, dapat ditentukan dimana prediksi yang tepat berada di antara kedua estimasi tersebut.

6.2.1 Strategi Yang Dapat Diotomatisasi

Terdapat strategi tertentu dalam analisis dampak yang dapat diotomatisasi dengan menggunakan metode algoritma untuk mengidentifikasi penyebaran perubahan dan dampak tidak langsung. Strategi ini memerlukan sistem yang terstruktur agar dapat digunakan untuk melakukan analisis dampak secara otomatis. Contoh dari strategi ini termasuk analisis *penelusuran dan ketergantungan*, serta teknik pemotongan. Strategi-strategi ini berguna untuk menilai bagaimana perubahan utama pada sistem dapat menyebabkan perubahan sekunder. Namun, hal ini juga kurang cocok untuk mengidentifikasi dampak langsung dari perubahan tersebut.

6.2.1.1 Analisis Penelusuran/Ketergantungan (*Traceability/Dependency Analysis*)

Analisis penelusuran dan analisis ketergantungan adalah dua strategi yang melibatkan pemeriksaan hubungan antar entitas dalam perangkat lunak, namun dengan cakupan dan tingkat detail yang berbeda. Analisis Penelusuran melibatkan semua jenis entitas dalam perangkat lunak, sedangkan analisis ketergantungan fokus pada hubungan tingkat rendah yang diekstrak dari kode sumber. Analisis ketergantungan dapat memberikan prediksi dampak yang sangat akurat karena didasarkan pada representasi sistem yang paling tepat, tetapi hanya bisa dilakukan pada tahap akhir proyek. Di sisi lain, analisis Penelusuran

bergantung pada kelengkapan dan konsistensi hubungan yang diidentifikasi sejak awal pengembangan. Meskipun informasi Penelusuran yang lengkap dapat membuat analisis ini sangat kuat, hubungan yang besar dan kompleks dapat membuat analisis menjadi sulit. Penggunaan matriks penelusuran dengan semantik yang ditingkatkan dan algoritma yang disesuaikan dapat membantu membatasi dan memvisualisasikan dampak tidak langsung. Analisis ini penting dalam rekayasa kebutuhan yang berlangsung sepanjang siklus hidup perangkat lunak, memungkinkan pembentukan hubungan antara berbagai entitas perangkat lunak seiring berkembangnya desain dan implementasi.

Berikut adalah penjelasan tentang tahapan-tahapan teknik *Traceability/Dependency Analysis* :

1. **Pemeriksaan Hubungan:**

   Langkah pertama adalah memeriksa dan memahami hubungan antara komponen-komponen dalam perangkat lunak. Kita perlu mengetahui bagaimana fungsi-fungsi, variabel, dan modul saling terhubung.

2. **Ekstraksi Ketergantungan:**

   Selanjutnya, kita mengekstrak ketergantungan dari kode sumber untuk membuat diagram yang menunjukkan hubungan ini. Grafik panggilan akan menunjukkan fungsi mana yang memanggil fungsi lain, struktur kontrol akan menunjukkan alur keputusan dalam kode, dan grafik data akan menunjukkan bagaimana data bergerak melalui program.

3. **Prediksi Dampak:**

   Terakhir, kita menggunakan informasi dari grafik ketergantungan untuk memprediksi dampak perubahan. Ini membantu kita memahami bagaimana perubahan pada satu bagian kode akan mempengaruhi bagian lainnya, sehingga kita dapat membuat keputusan yang lebih baik tentang perubahan tersebut.

Berikut cara yang dapat digunakan pada teknik *Traceability/Dependency Analysis* :

1. **Ekstrak Ketergantungan:**

   Langkah pertama adalah mengekstrak ketergantungan dari kode sumber untuk membuat grafik yang menunjukkan hubungan antara berbagai bagian dari kode. Grafik panggilan menunjukkan fungsi mana yang memanggil fungsi lain, struktur kontrol menunjukkan bagaimana keputusan diambil dalam kode, dan grafik data menunjukkan bagaimana data mengalir melalui program.

2. **Menganalisis Hubungan:**

   Selanjutnya, analisis hubungan antara berbagai entitas dalam perangkat lunak menggunakan informasi dari grafik ketergantungan. Tujuannya adalah untuk memahami bagaimana perubahan pada satu bagian kode dapat mempengaruhi bagian lain.

3. **Memprediksi Dampak:**

Gunakan informasi yang diperoleh dari analisis ketergantungan untuk memprediksi dampak perubahan secara akurat. Dengan memahami ketergantungan yang ada, kita bisa melihat implikasi dari modifikasi yang diusulkan pada seluruh sistem perangkat lunak.

Kelebihan *Traceability/Dependency Analysis* :

1. **Presisi:**

Analisis ketertelusuran/ketergantungan memberikan pemahaman yang sangat tepat mengenai bagaimana bagian-bagian dalam sistem perangkat lunak saling berhubungan dan bergantung.

2. **Prediksi Dampak yang Akurat:**

Teknik ini memungkinkan kita untuk memprediksi dampak dari perubahan secara akurat dengan memahami ketergantungan yang ada dalam kode.

3. **Strategi yang Matang:**

Analisis ketergantungan adalah strategi yang telah terbukti efektif untuk analisis dampak, memberikan wawasan mendalam tentang implikasi dari setiap perubahan.

Kekurangan *Traceability/Dependency Analysis*:

1. **Ketersediaan yang Terlambat:**

Analisis ketergantungan memerlukan kode sumber yang mungkin tidak tersedia sampai tahap akhir proyek, sehingga sulit dilakukan lebih awal.

2. **Kompleksitas dalam Sistem Besar:**

Sistem yang sangat besar dengan banyak ketergantungan dapat membuat analisis menjadi sangat rumit dan sulit dikelola.

3. **Bidang Aplikasi yang Sempit:**

Analisis ketergantungan mungkin kurang efektif jika kode sumber tidak tersedia atau lengkap, sehingga penerapannya terbatas pada situasi di mana kode sumber sudah tersedia.

6.2.1.2 Teknik Pemotongan (*Slicing)*

Teknik pemotongan *(Slicing)* bertujuan untuk memahami ketergantungan dengan menggunakan potongan-potongan program yang independen. Potongan program dibagi menjadi potongan dekomposisi, yang berisi tempat perubahan, dan sisa program, potongan komplementer. Pemotongan didasarkan pada ketergantungan data dan kontrol dalam

program. Perubahan yang dilakukan pada potongan dekomposisi di sekitar variabel yang menjadi dasar potongan dijamin tidak akan memengaruhi potongan komplementer. Teknik ini membatasi ruang lingkup penyebaran perubahan dan menjadikan ruang lingkup tersebut eksplisit. Pemotongan arsitektur, yang merupakan konsep serupa, dapat digunakan pada tahap awal pengembangan untuk menganalisis dampak perubahan pada arsitektur perangkat lunak. Dalam rekayasa kebutuhan, teknik pemotongan dapat digunakan untuk mengisolasi dampak perubahan kebutuhan pada bagian tertentu dari sistem.

Teknik *Slicing* adalah metode dalam rekayasa perangkat lunak yang membantu memisahkan dan mengelola bagian-bagian program yang terpengaruh oleh perubahan. Berikut adalah penjelasan tentang tahapan-tahapan Teknik *Slicing* :

1. **Pembuatan Irisan Dekomposisi:**

   Tahap ini melibatkan pemecahan program menjadi beberapa bagian (irisan) berdasarkan area perubahan yang dibutuhkan. Irisan ini mencakup kode spesifik yang terpengaruh oleh perubahan kebutuhan atau fitur baru.

2. **Identifikasi Irisan Pelengkap:**

   Setelah membuat irisan dekomposisi, bagian program yang tidak terpengaruh oleh perubahan tersebut membentuk irisan pelengkap. Ini berarti bahwa perubahan pada irisan dekomposisi tidak akan mempengaruhi bagian lain dari program yang ada di irisan pelengkap.

3. **Analisis Ketergantungan Data dan Kontrol:**

   Slicing dilakukan berdasarkan analisis ketergantungan data dan kontrol dalam program. Analisis ini membantu mengidentifikasi bagaimana data mengalir melalui program dan bagaimana berbagai bagian program saling berinteraksi. Dengan isolasi dampak perubahan pada irisan tertentu, dapat membatasi dan menjelaskan ruang lingkup perubahan, sehingga lebih mudah untuk mengelolanya.

Berikut cara yang dapat digunakan pada Teknik *Slicing* :

1. **Identifikasi Area Perubahan:**

   Tentukan bagian spesifik dalam program yang perlu diubah berdasarkan kebutuhan baru atau pembaruan fitur. Misalnya, perlu memperbarui cara menghitung diskon di aplikasi e-commerce. Identifikasi bagian kode yang menangani perhitungan diskon.

2. **Membuat Irisan Dekomposisi:**

   Potong atau pisahkan bagian program yang terkait langsung dengan perubahan yang telah diidentifikasi. Ini disebut irisan dekomposisi. Misalnya, Ambil semua kode yang terkait dengan perhitungan diskon dan buatlah sebagai irisan terpisah dari sisa program.

3. **Menganalisis Ketergantungan:**

Analisis bagaimana data dan alur kontrol (seperti pernyataan if dan loop) berinteraksi dalam irisan dekomposisi untuk memahami dampak perubahan. Misalnya, Periksa variabel dan fungsi yang digunakan dalam perhitungan diskon dan bagaimana mereka berinteraksi satu sama lain.

4. **Mengisolasi Dampak:**

Pastikan bahwa perubahan yang dibuat dalam irisan dekomposisi tidak mempengaruhi bagian lain dari program (irisan komplemen). Misalnya, Jika mengubah cara diskon dihitung, pastikan perubahan tersebut tidak mempengaruhi fitur lain seperti penanganan pembayaran atau pengiriman barang.

5. **Batasi Perambatan Perubahan:**

Dengan menggunakan teknik Slicing, batasi dampak perubahan hanya pada irisan dekomposisi, sehingga ruang lingkup perubahan tidak menyebar ke seluruh program. Misalnya, Pastikan bahwa perubahan hanya mempengaruhi bagian perhitungan diskon dan tidak merambat ke fungsi lain, sehingga kode lainnya tetap berfungsi dengan baik.

Kelebihan Teknik *Slicing*:

1. **Presisi:**

Teknik Slicing memberikan pemahaman yang sangat tepat tentang bagaimana bagian-bagian program saling bergantung dan area mana yang akan terpengaruh oleh perubahan.

2. **Isolasi:**

Teknik ini memungkinkan untuk membuat perubahan pada satu bagian program (irisan dekomposisi) tanpa mempengaruhi bagian lainnya (irisan komplemen).

3. **Cakupan Eksplisit:**

Teknik Slicing membatasi dampak perubahan hanya pada bagian tertentu dari program, sehingga tahu persis bagian mana yang akan terpengaruh.

Kekurangan Teknik *Slicing*:

1. **Kompleksitas:**

Teknik Slicing bisa sangat rumit untuk diterapkan dan dianalisis, terutama untuk program yang besar dan kompleks.

2. **Ketergantungan Alat:**

Teknik ini sering membutuhkan alat khusus untuk melakukan Slicing dengan efektif, yang mungkin tidak selalu tersedia atau mudah digunakan.

3. **Presentasi Berbasis Karakter:**

Beberapa alat Slicing menggunakan teknik presentasi berbasis karakter, yang bisa membuat analisis lebih sulit karena tidak selalu mudah dimengerti.

6.2.2 Strategi Manual

Strategi analisis dampak manual tidak bergantung pada spesifikasi terstruktur seperti strategi yang dapat diotomatisasi. Oleh karena itu, ada risiko bahwa mereka kurang presisi dalam prediksi dampaknya. Namun, strategi manual ini mungkin lebih mudah diterapkan dalam proses manajemen perubahan dan sering digunakan di industri tanpa memperhatikan presisi mereka. Strategi manual, seperti penggunaan dokumentasi desain dan wawancara, biasanya digunakan untuk menilai Set Dampak Awal dengan mengidentifikasi dampak langsung. Meskipun identifikasi dampak sekunder mungkin dimungkinkan, strategi yang dapat diotomatisasi lebih baik untuk menanganinya. Perlu dicatat bahwa strategi manual, seperti yang dijelaskan di sini, juga dapat digunakan untuk menangkap hubungan ketertelusuran antara SLO yang dapat digunakan dalam analisis ketertelusuran.

6.2.2.1 Desain Dokumentasi

Dokumentasi desain memiliki berbagai bentuk, seperti sketsa arsitektur, model arsitektur berbasis tampilan, diagram UML berbasis objek, dan deskripsi teks komponen perangkat lunak. Kualitas dokumentasi desain sangat tergantung pada tujuan penulisannya, frekuensi pembaruan, dan informasi yang terkandung di dalamnya. Namun, seringkali di industri, dokumentasi desain ditulis hanya sebagai formalitas atau bahkan setelah proyek selesai, tanpa memperhatikan kebutuhan aktual. Untuk melakukan analisis dampak, dokumentasi tersebut harus terbaru dan konsisten dengan implementasi yang ada. Keberhasilan dan presisi analisis ini juga tergantung pada pengetahuan dan keterampilan analis, ketersediaan dokumentasi, dan jumlah informasi yang disampaikan dalam dokumentasi tersebut. Dokumentasi yang jelas dan konsisten sangat penting dalam analisis dampak karena dapat menghindari ruang interpretasi yang ambigu dan meningkatkan ketidakpastian terkait dampak dari perubahan yang diusulkan. Dengan mempertimbangkan faktor-faktor ini, analisis dampak dari perubahan kebutuhan dapat dilakukan dengan mengidentifikasi SLO desain yang terpengaruh oleh perubahan tersebut. Langkah-langkah tambahan yang dapat dilakukan untuk meredakan upaya analisis dampak antara lain adalah menjaga alasan desain dan memperkirakan dampak kebutuhan segera setelah dikembangkan. Dokumentasi desain yang terstruktur juga dapat digunakan dengan analisis penelusuran untuk mengidentifikasi dampak tidak langsung.

Berikut adalah penjelasan tentang tahapan-tahapan teknik *Design Documentation* :

1. **Tinjauan Dokumentasi:**

Langkah pertama adalah meninjau berbagai bentuk dokumentasi desain yang ada. Ini bisa berupa sketsa arsitektur, diagram UML *(Unified Modeling Language)*, dan deskripsi tekstual dari komponen perangkat lunak. Tujuannya adalah untuk memahami tujuan dari dokumentasi tersebut, seberapa sering diperbarui, dan informasi apa saja yang disertakan.

2. **Identifikasi Dampak Langsung:**

Setelah memahami dokumentasi, identifikasi dampak langsung dari perubahan atau persyaratan baru. Gunakan dokumentasi untuk melihat bagaimana perubahan tersebut akan mempengaruhi bagian-bagian lain dari sistem.

3. **Menangkap Tautan Ketertelusuran:**

Gunakan dokumentasi desain untuk menangkap dan memahami hubungan ketertelusuran antara berbagai komponen sistem. Pastikan dokumentasi selalu diperbarui dan konsisten agar dapat digunakan untuk menilai dampak perubahan dengan akurat.

Berikut cara yang dapat digunakan pada Teknik *Design Documentation* :

1. **Tinjau Dokumentasi:**

Langkah pertama adalah memeriksa berbagai bentuk dokumentasi desain yang ada. Ini termasuk sketsa arsitektur, diagram UML (Unified Modeling Language), dan deskripsi teks tentang komponen perangkat lunak. Tujuan dari langkah ini adalah untuk memahami tujuan dari dokumentasi tersebut, seberapa sering diperbarui, dan informasi apa saja yang disertakan.

2. **Menilai Dampak:**

Gunakan dokumentasi desain untuk menilai dampak dari persyaratan baru atau yang diubah pada sistem. Caranya adalah dengan menghubungkan persyaratan tersebut dengan elemen-elemen dalam dokumentasi desain untuk mengidentifikasi dampak langsung dan memahami bagaimana perubahan ini akan mempengaruhi sistem secara keseluruhan.

3. **Menangkap Tautan Ketertelusuran:**

Pastikan menangkap hubungan ketertelusuran antara berbagai elemen dalam dokumentasi desain untuk melakukan analisis ketertelusuran. Ini berarti memastikan dokumentasi selalu diperbarui dan konsisten dengan implementasi sistem yang ada, sehingga penilaian dampak perubahan dapat dilakukan dengan akurat.

Kelebihan Teknik *Design Documentation*:

1. **Kejelasan:**

Dokumentasi desain memberikan gambaran yang jelas tentang arsitektur dan komponen sistem. Ini membantu semua anggota tim memahami bagaimana sistem dirancang dan bagaimana komponen saling berinteraksi.

2. **Ketertelusuran:**

Dokumentasi membantu menangkap hubungan ketertelusuran antara persyaratan dan elemen desain. Ini memungkinkan tim untuk melacak bagaimana perubahan dalam persyaratan akan mempengaruhi elemen-elemen desain tertentu.

3. **Informasi Terstruktur:**

Dokumentasi desain menyediakan informasi yang terstruktur dan terorganisir tentang desain sistem. Ini memudahkan tim untuk menilai dampak langsung dari setiap perubahan yang dilakukan.

Kekurangan *Design Documentation* :

1. **Dokumentasi yang Sudah Ketinggalan Zaman:**

Jika dokumentasi tidak diperbarui secara teratur, informasi yang ada mungkin tidak lagi mencerminkan kondisi sistem saat ini. Ini bisa mengarah pada keputusan yang tidak akurat.

2. **Aksesibilitas:**

Jika dokumentasi sulit diakses atau tersembunyi di dalam sistem manajemen dokumen, informasi penting mungkin terlewatkan selama analisis dampak.

3. **Ambiguitas:**

Dokumentasi yang ambigu atau tidak konsisten dapat menyebabkan salah tafsir dan ketidakpastian dalam menilai dampak perubahan. Hal ini dapat menghambat proses analisis dan implementasi perubahan.

6.2.2.2 Wawancara

Wawancara dengan pengembang yang berpengetahuan adalah cara yang paling umum digunakan untuk mendapatkan informasi tentang kemungkinan efek dari kebutuhan baru atau yang berubah menurut sebuah studi tentang analisis dampak. Studi tersebut menemukan bahwa pengembang menganggapnya sangat efektif secara biaya untuk bertanya kepada orang yang berpengetahuan daripada mencari di dokumen atau sumber informasi lainnya. Komunikasi yang intens antara pengembang juga disebutkan oleh pengembang sebagai faktor keberhasilan untuk proyek pengembangan perangkat lunak. Analisis kode sumber merupakan cara kedua yang paling umum digunakan untuk memperoleh informasi tentang kemungkinan

dampak dari kebutuhan baru atau yang berubah. Meskipun semua pengembang mengatakan mereka mewawancarai pengembang lain dan berkonsultasi dengan kode sumber, sekitar setengah dari pengembang menjawab bahwa mereka juga berkonsultasi dengan informasi, seperti model use-case dan model objek, yang disimpan dalam alat *CASE* yang digunakan.

Ketika ditanya mengapa informasi dalam model objek tidak digunakan lebih luas, para pengembang menjawab bahwa informasi dalam model objek tidak cukup rinci untuk analisis dampak. Di antara beberapa pengembang, terutama para pendatang baru, sikap terhadap penggunaan model objek sebagai dasar untuk menentukan perubahan akibat kebutuhan baru atau yang berubah kurang positif. Namun, model objek dan alat *CASE* tertentu yang digunakan disebut sebagai alat yang baik untuk mendokumentasikan analisis dampak dan untuk menjawab pertanyaan tentang hubungan antara kebutuhan dan objek desain.

Berikut adalah penjelasan tentang tahapan-tahapan Teknik *Interviews* :

1. **Akuisisi Pengetahuan:**

   Langkah pertama adalah mengumpulkan informasi dengan berbicara langsung kepada pengembang yang sangat memahami sistem. Ini membantu mendapatkan pandangan mendalam tentang bagaimana persyaratan baru atau perubahan dapat mempengaruhi sistem.

2. **Penilaian Dampak:**

   Setelah mendapatkan informasi, gunakan wawasan ini untuk menilai dampak perubahan. Identifikasi efek langsung dan sekunder yang mungkin muncul dari perubahan tersebut.

3. **Komunikasi dan Kolaborasi:**

   Gunakan wawancara untuk meningkatkan komunikasi antar tim. Bagikan informasi yang diperoleh sehingga semua pengembang memiliki pemahaman yang sama tentang dampak perubahan dan bisa bekerja sama dalam menyusun solusi.

Berikut cara yang dapat digunakan pada Teknik *Interviews* :

1. **Pengumpulan Pengetahuan:**

   Langkah pertama adalah melakukan wawancara dengan pengembang yang memiliki pengetahuan mendalam tentang sistem. Tujuannya adalah untuk mendapatkan wawasan tentang bagaimana persyaratan baru atau perubahan dapat mempengaruhi sistem.

2. **Penilaian Dampak:**

   Gunakan informasi yang diperoleh dari wawancara untuk menilai dampak perubahan persyaratan. Analisis informasi tersebut untuk memahami dampak langsung dan dampak sekunder yang mungkin terjadi pada sistem.

3. **Kolaborasi dan Komunikasi:**

Bagikan informasi yang diperoleh dari wawancara dengan tim pengembang lainnya untuk memastikan pemahaman yang sama dan mendorong kolaborasi. Gunakan wawancara sebagai alat untuk memfasilitasi komunikasi dan pertukaran pengetahuan dalam tim.

Kelebihan Teknik *Interviews* :

1. **Informasi yang Berwawasan Luas:**

Wawancara memberikan kesempatan untuk mendapatkan wawasan langsung dari pengembang yang berpengetahuan luas. Mereka dapat menjelaskan dampak potensial dari perubahan persyaratan dengan detail yang mendalam.

2. **Pemahaman Kontekstual:**

Pengembang dapat memberikan informasi yang sesuai dengan konteks spesifik dari proyek yang mungkin tidak tersedia dalam dokumen atau sumber lainnya. Ini membantu dalam memahami bagaimana perubahan akan mempengaruhi sistem secara keseluruhan.

3. **Komunikasi:**

Wawancara mendorong komunikasi dan kolaborasi antara anggota tim. Dengan berdiskusi langsung, pengembang dapat berbagi ide dan wawasan, yang meningkatkan proses analisis dampak.

Kekurangan Teknik *Interviews*:

1. **Subjektivitas:**

Tanggapan dalam wawancara mungkin bersifat subjektif dan dipengaruhi oleh pengalaman atau bias pribadi. Ini dapat mempengaruhi akurasi informasi yang diperoleh.

2. **Memakan Waktu:**

Melakukan wawancara dapat memakan banyak waktu, terutama jika melibatkan banyak pengembang atau jika wawancara dilakukan berulang kali untuk mendapatkan informasi yang lengkap.

3. **Cakupan Terbatas:**

Wawancara mungkin tidak mencakup semua aspek dari analisis dampak dan dapat mengabaikan detail tertentu yang lebih baik diperoleh melalui metode lain seperti analisis kode sumber.

Sumber: *E-Book (Engineering and Managing Software Requirements)*