# Analisis Inkonsistensi Pada Kebutuhan Perangkat Lunak

Evi Lestari Pratiwi
Politeknik Hasnur Kalimantan Selatan
evi.krusty@gmail.com

Abstract - Specifies the software requirements specification is a very complex process and takes a long time. Software (software) computer is a collection of electronic data that is stored and managed by computer, electronic data which is stored by the computer may be programs or instructions that will execute a command. The software is also called interpreter commands that run computer users to be routed or processed by the hardware. Through software or software of a computer can run a command. Software requirements describe the requirements and limitations - limitations that exist in software products that contribute to the solution of some real-world problems. In the software requirements, the need for consistency between the software requirements specification, as a software company is successful if it has the consistency of software requirements specification itself. Data inconsistency is the incompatibility of data on some related information due to data changes. There are many methods used to measure the level of inconsistency software needs.

***Keyword: software, software requirements, Consistency, Inconsistency, Methods Inconsistency***

**Abstrak -** Menentukan spesifikasi kebutuhan perangkat lunak adalah suatu proses yang sangat kompleks dan membutuhkan waktu yang panjang. Perangkat lunak (software) komputer adalah sekumpulan data elektronik yg disimpan dan diatur oleh komputer, data elektronik yg disimpan oleh komputer itu dapat berupa program atau instruksi yg akan menjalankan suatu perintah. Perangkat lunak disebut juga sebagai penerjemah perintah-perintah yang dijalankan pengguna komputer untuk diteruskan atau diproses oleh perangkat keras. Melalui software atau perangkat lunak suatu komputer dapat menjalankan suatu perintah. Kebutuhan perangkat lunak menggambarkan kebutuhan dan batasan – batasan yang ada pada produk perangkat lunak yang berkontribusi terhadap solusi dari beberapa masalah dunia nyata. Dalam kebutuhan perangkat lunak, perlu adanya konsistensi antara spesifikasi kebutuhan perangkat lunak, karena suatu perangkat lunak dikatakan berhasil jika mempunyai konsistensi spesifikasi kebutuhan dari perangkat lunak itu sendiri. Inkonsistensi data adalah tidak sesuainya data pada beberapa informasi yang saling berhubungan disebabkan adanya perubahan data. Ada banyak metode yang digunakan untuk mengukur tingkatan inkonsistensi kebutuhan perangkat lunak.

***Keyword : Perangkat lunak, Kebutuhan perangkat lunak, Konsistensi, Inkonsistensi, Metode Inkonsistensi***

## 1.2. Latar Belakang

Menentukan spesifikasi kebutuhan perangkat lunak adalah untuk menjelaskan apa yang diinginkan dan diperlukan oleh pengguna. Hal ini menyebabkan kesalahan dan kegagalan dalam menentukan spesifikasi kebutuhan perangkat lunak itu menjadi sangat mahal karena perangkat lunak yang dihasilkan menjadi tidak sesuai dengan apa yang diinginkan oleh pengguna atau menjadi tidak menyelesaikan masalah dari pengguna sehingga harus dikerjakan ulang yang akan menghabiskan pengeluaran tambahan.

Menurut hasil survei DeMarco (1993) kegagalan proyek perangkat lunak menyebabkan biaya yang dikeluarkan terbuang percuma, karena mahalnya kesalahan dan kegagalan dalam penentuan spesifikasi kebutuhan perangkat lunak, perlu diperhatikan beberapa aspek, salah satunya adalah konsistensi dalam penulisan dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL). Suatu dokumen SKPL dapat dikatakan baik jika memiliki konsistensi baik dalam isi maupun penulisan. Jika terjadi inkonsistensi dalam penulisan SKPL, ada beberapa kemungkinan yang akan terjadi, salah satunya adalah pengguna menjadi sulit mengerti maksud dari SKPL tersebut. Hal ini dapat mengakibatkan pengguna selalu komplain terhadap spesifikasi kebutuhan yang sudah dibuat sebelumnya dan akan memperlambat proses implementasi. Penentuan kebutuhan memiliki konten yang tidak konsisten, hal ini mengakibatkan proses pengimplementasian kebutuhan akan menjadi kacau sebab isi dari kebutuhannya tidak jelas atau saling bertubrukan dengan kebutuhan yang lain.

Dari permasalahan inkonsistensi, banyak peneliti yang melakukan riset dibidang inkonsistensi untuk mengurangi atau mendeteksi inkonsistensi dari kebutuhan perangkat lunak. Pada penelitian sebelumnya,

dengan melakukan prototyping semua spesifikasi sistem (Acharya,2005), beberapa penelitian lain menggambarkan seperangkat mekanisme untuk menangani inkonsistensi yang dihasilkan dengan menggunakan teknik multi-agen (Mu, 2008). Penelitian menggunakan pseudo code untuk merepresentasikan kalimat-kalimat yang ada di dalam dokumen SPKL, kemudian mencocokkan setiap kebutuhannya apakah terjadi inkonsistensi atau tidak (Yikun, 2007). Pengembangan yang dilakukan dengan menggunakan ontologi, di mana setiap kata dan kalimat diartikan persatuan kemudian dicocokkan apakah diantara maksud yang sudah diketahui tersebut ada yang tidak sesuai satu dengan yang lain atau inkonsisten (Kroha, 2009). Metode pengembangan dilakukan dengan menggunakan Automated Software Tool untuk memeriksa inkonsistensi yang terjadi dan direpresentasikan dalam bentuk teks, visual, formal dan informal (Kamalrudin, 2010).

### 1.2. Kajian Pustaka

#### a. Persyaratan Spesifikasi Domain Konsistensi

Penelitian yang dilakukan oleh (Acharya & George, 2005) mengadopsi pendekatan dengan menggunakan berbagai macam teknik seperti spesifikasi inspeksi, prototipe dan pengujian untuk spesifikasi kebenaran dan untuk memastikan bahwa konsistensi sistem dapat dipertahankan. Penelitian yang dilakukan membahas teknik yang berbeda untuk mendapatkan kepercayaan dalam kebenaran spesifikasi dengan mengimplementasikan prototipe yang dihasilkan oleh penerjemah secara otomatis.

Penelitian yang dipaparkan menghasilkan kondisi dengan kehati – hatian dalam pengujian perangkat lunak yang akan mendeteksi berbagai macam kondisi yang memungkinkan terjadinya inkonsistensi terhadap suatu perangkat lunak.

Konsistensi yang diuji dengan keadaan awal dari suatu sistem operasi dengan menggunakan kasus uji coba pada spesifikasi executable. Hasil pengujian konsistensi pada sistem dengan menggunakan kasus uji coba pada spesifikasi executable. Uji kasus dikembangkan untuk spesifikasi juga dapat digunakan dalam pengujian implementasi akhir dari sistem untuk memeriksa bahwa pelaksanaan mempertahankan konsistensi dalam sistem.

**Table 1 Hasil Ujicoba Metode Lightweight**

| *Module Name* | *Test Cases* | *Errors Found* | 4 |
|---|---|---|---|
| | | *Typos* | *Con.** |
| *RESOURCES* | 63 | 0 | 4 |
| *RESREQBAG* | 10 | 0 | 0 |
| *RASSIGN* | 9 | 0 | 0 |
| *HOARD* | 11 | 0 | 0 |
| *TASK* | 43 | 0 | 1 |
| *MSSS* | 45 | 2 | 1 |
| *MHS* | 48 | 1 | 8 |
| *SYS/MOBICHART* | 90 | 5 | 6 |
| *TOTAL =* | 319 | 8 | 20 |

Penelitian yang dilakukan dengan contoh sistem yang cukup kompleks, membahas bagaimana kondisi konsistensi yang ditemukan selama awal spesifikasi selama tahap awal dari pengembangan sistem.

Pada penelitian dengan menggunakan metode yang dilakukan ditemukan masalah pada tahap awal dari siklus pengembangan perangkat lunak sangat penting. Saat mengembangkan sistem yang kompleks, seorang designer harus mempertimbangkan 2 jenis spesifikasi dalam inkonsistensi : (a) kegagalan untuk menentukan apa yang diinginkan dan (b) kegagalan pada waktu verifikasi

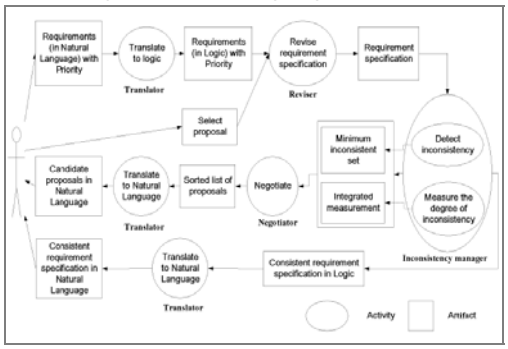
#### b. Model Proses Pengukuran Pengelolaan Konsistensi Perangkat Lunak

Masukan dan keluaran dari proses pengukuran inkonsistensi adalah kebutuhan perangkat lunak dalam natural language. Proses mekanisme pada metode ini adalah (Mu, Jin, & Zowghi, 2008) :

1) *Requirement Translation*
   Menerjemahkan kebutuhan perangkat lunak dari *natural language* menjadi *pseudo code.*
2) *Specification Revision*
   Mengurutkan specifikasi kebutuhan berdasarkan ketentuan, misalkan berdasarkan prioritas. Pada proses ini juga bisa menambahkan atau mengurangi kebutuhan perangkat lunak untuk menjaga konsistensi.
3) *Inconsistency Management*
   Pada tahap ini ada 2 fase yang dilalui, pertama adalah *inconsistency detector* untuk mendeteksi konsistensi, kemudian *inconsistency measurer*.
4) *Negotiation*
   Dirancang untuk merepresentasikan kebutuhan perangkat lunak dar setiap

individu untuk menyelesaikan masalah inkonsistensi sebelumnya

5) *Proposal Translation*
Merubah *pseudo code* yang telah diproses dan dicek inkonsistensinya menjadi *natural language*.



Gambar 1 Measurement-Driven Method

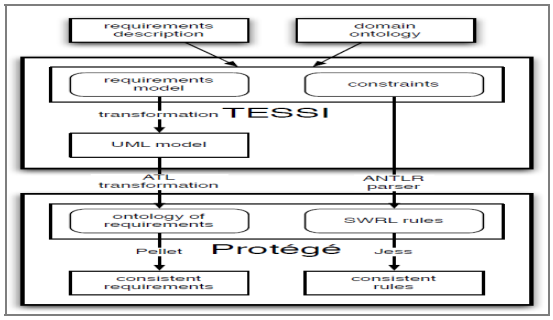
Dalam model proses yang dilakukan, ketika inkonsistensi terdeteksi, seperangkat mekanisme untuk menangani inkonsistensi akan dihasilkan menggunakan teknik Multi-agent automated negotiations dengan menggunakan pengukuran terhadap inkonsistensi. Sudut pandang yang terlibat dalam inkonsistensi kemudian akan memasuki negosiasi oleh sedang disajikan. Fungsi yang difasilitasi dalam proses yang dilakukan menganggap bahwa pernyataan persyaratan natural language pertama kali diterjemahkan ke pseudo code menggunakan software penerjemah. Penangani inkonsistensi akan diterjemahkan kembali dari logika formal ke dalam bahasa alami sebelum yang disajikan untuk diseleksi.

### c. Ontologi Persyaratan Inkonsistensi

Ontologi digunakan karena ontologi terdiri dari (Kroha, Janetzko, & Labra, 2009) :

1) Membangun ontologi menggunakan *Protege*
2) Melakukan pengecekan ontologi untuk konsistensi menggunakan hirarki dan rule
3) Sistem analis menulis deskripsi kebutuhan perangkat lunak
4) Sistem analis membangun UML model yang sesuai dengan perangkat lunak yang dibangun
5) Konversi kebutuhan perangkat lunak yang dipaparkan berupa UML menjadi sebuah *problem ontology* menggunakan konvertor ATL
6) Memeriksa *problem ontology* yang terdapat inkonsistensi di dalamnya
7) Menggabungkan *problem ontology* dengan *domain ontology* yang telah dibuat sebelumnya kemudian periksa lagi inkonsistensinya
8) Identifikasi inkonsistensinya
9) Temukan bagian yang error kemudian membetulkannya
10) Membangun UML baru yang kebutuhannya telah dihilangkan inkonsistensinya
11) Proses yang dilakukan selalu terjadi berulang-ulang sampai tidak ditemukan inkonsistensi lagi.

Sebelum UML model diimplementasikan, UML harus dibaca kembali oleh sistem analis, jika masih ada yang inkonsisten, maka proses akan dimulai dari awal lagi, jika tidak ada, UML telah dapat diimplementasikan. Gambar 2 memperlihatkan implementasi penggunaan ontologi dalam pengecekkan inkonsistensi pada kebutuhan perangkat lunak (Kroha et al., 2009)



Gambar 2 *Ontology-Based Method*

### d. Metode Kebutuhan Analisis Inkonsistensi

Penelitian dengan metode persyaratan inkonsistensi (Yikun, Peng, Duwu, & Hui, 2007) menyediakan kemampuan untuk menentukan persyaratan global pada saat melakukan proses penganalisaan inkonsistensi terhadap perangkat lunak, sedangkan logika yang dipilih atas dasar dari analisa inkonsistensi yang terjadi secara otomatis dan memungkinkan untuk menerapkan strategi yang berbeda untuk menganalisis suatu inkonsistensi kebutuhan perangkat lunak.

Beberapa metode persyaratan yang digunakan antara lain :

1) Model kebutuhan mesin – mesin klasik yang biasa digunakan dalam deskripsi sistem terdiri dari beberapa bagian, yaitu alphabet input, fungsi transisi, dan keadaan akhir. Namun beberapa elemen menjadi sangat diperlukan ketika menggambarkan kebutuhan perangkat lunak
2) *State equivalence* – persyaratan menggambarkan proses sistem target yang berbeda untuk mencapai operasi

tertentu. Langkah pertama analisis kebutuhan adalah untuk mengetahui hubungan kesetaraan model persyaratan yang inkonsisten

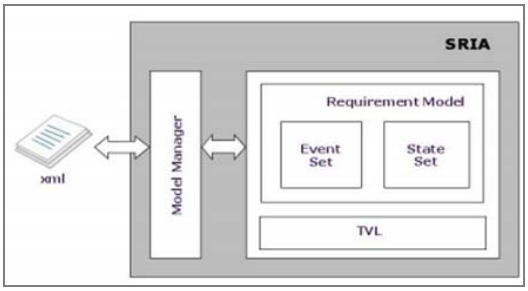
3) Nilai logika – inkonsistensi terjadi di sebagian besar analisis kebutuhan praktis. Sebagai logika klasik tidak dapat menangani informasi yang tidak konsisten, nilai logika telah diusulkan dan dapat berkembang dengan pesat.

Model persyaratan terdiri dari 2 set, yaitu *the state set dan the event set*, sehingga sumber inkonsistensi berasal dari kedua set model persyaratan. Model persyaratan data yang tidak konsisten adalah model target kebutuhan sistem yang sama dengan ataupun berbeda dari kedua set model persyaratan.

Tujuan dari kebutuhan inkonsistensi yang ditulis dalam penelitian ini untuk menentukan model persyaratan yang digunakan sehingga proses analisis kebutuhan inkonsistensi menjadi proses yang dapat membangun model persyaratan terpadu.

Hubungan masing – masing bagian dalam kebutuhan perangkat lunak berbeda dalam model inkonsistensi memiliki deskripsi yang sama. Analisis kebutuhan inkonsistensi didasarkan pada premis bahwa terjadi hubungan yang ekuvalen antara masing – masing dari model inkonsistensi ditentukan. Tujuan analisis yang dilakukan adalah untuk mengetahui hubungan kesetaraan dengan metode analisis yang bervariasi.

*Software Requirement Inconsistency Analys* (SRIA) adalah alat bantu untuk analisis kebutuhan yang dilakukan dalam metode persyaratan.




Gambar 1 Struktur SRIA

Gambar 3 menggambarkan bahwa SRIA memiliki beberapa modul fungsional. SRIA berkomunikasi dengan xml sebagai pemangku kepentingan melalui *Modul Manager* dengan menggunakan *UnitAnalyzer* untuk melakukan analisis inkonsistensi yang bekerja pada 2 model yang inkonsistensi. *UnitAnalyzer* digunakan dalam semua model inkonsistensi, sehingga model kebutuhan perangkat lunak yang dianalisis tercapai. TVL berfungsi untuk mengurangi prosesor logika yang berbasis analisis inkonsistensi.

**e. Inkonsistensi Diukur Menggunakan Perangkat Lunak Otomatis**

Tujuan alat pengukur perangkat lunak untuk mengukur inkonsistensi perangkat lunak (Kamalrudin, 2009) adalah untuk membuat suatu dukungan pengguna yang lebih baik dan mengembangkan perangkat lunak dengan persyaratan informal dan semiformal untuk menjaga agar tetap konsisten. Pendekatan lyang digunakan untuk menghasilkan alat prototipe secara otomatis dengan menyediakan fasilitas untuk kebutuhan tekstual yang digunakan sebagai media pemeriksaan inkonsistensi suatu perangkat lunak.

Metode yang dipaparkan menggunakan pendekatan iteratif untuk menambah manajemen konsistensi setelah evaluasi pada setiap tahap penelitian. Beberapa tahapan yang digunakan adalah : (Kamalrudin, 2009)

1) Melakukan tinjauan literatur konsistensi dan inkonsistensi dengan memeriksa persyaratan dalam persyaratan teknik domain, dibandingkan dan dievaluasi pendekatan mereka dalam memeriksa inkonsistensi persyaratan.
2) Mengidentifikasikan konsep awal bagaimana untuk mendukung persyaratan dalam memeriksa inkonsistensi suatu perangkat lunak, serta aspek kelengkapan dan kebenaran.
3) Mengumpulkan dan mengelompokan terminologi suatu perangkat lunak dengan mengikuti pola penggunaan yang menghasilkan sekenario dari inkonsistensi suatu perangkat lunak.
4) Mengembangkan prototipe untuk mengeksplorasi masalah dan penggalian kasus dan menelusuri antara persyaratan dengan menggunakan interaksi abstrak.
5) Mengembangkan seperangkat aturan konsistensi antara persyaratan tekstual dan penggunaan model kasus dari suatu perangkat lunak.
6) Mengidentifikasi skenario penggunaan yang tepat dan mengevaluasi hasil dari penggunaan manajemen konsistensi dan alat penelusuran jika terjadi perubahan yang dibuat sebagai suatu persyaratan.
7) Mengembangkan prototipe untuk memeriksa inkonsistensi secara otomatis dengan penggunaan desain antarmuka.
8) Mengevaluasi konsistensi secara otomatis dengan menggunakan studi kasus dan contoh skenario.
9) Merencanakan penyempurnaan dengan menambah dukungan lebih lanjut untuk persyaratan kualitas dengan

menggunakan pola studi kasus, serta dukungan agar manajemen konsistensi dapat terwujud.

Alat prototipe yang digunakan dalam melacak kebutuhan perangkat lunak yang inkonsistensi menggunakan Java, yang disimpan dalam database. Database terdiri dari frase yang menggambarkan interaksi abstrak untuk mengidentifikasi dan menguraikan dari kebutuhan bahasa alami.

Frase yang diuraikan dibandingkan dengan interaksi terminologi abstrak dalam database. Setiap perubahan atau modifikasi menggunakan kasus atau editor tekstual yang diharapkan dapat melacak masalah konsistensi kebutuhan perangkat lunak agar dapat dievaluasi.

**f. Perangkat Lunak**

Perangkat lunak adalah instruksi (program komputer) yang bila dieksekusi dapat menjalankan fungsi tertentu, struktur data yang dapat membuat program memanipulasi informasi, dan dokumen yang menjelaskan operasi dan penggunaan program (Pressman, 1997). Perangkat lunak lebih dikenal sebagai elemen logik dari pada fisik, oleh karena itu perangkat lunak memiliki karakteristik yang berbeda dari perangkat keras (Mike, 1995):

a. Perangkat lunak dikembangkan atau direkayasa, jadi tidak diproduksi dalam pengertian klasik.
b. Merupakan produk yang unik (tidak ada seri produksi).
c. Perangkat lunak tidak pernahakan rusak/aus karenaselalu diperbaharui.
d. Tidak terlihat (invisible).
e. Perangkat lunak pada umumnya dibangun sesuaikeinginan, jadi tidak dibentuk dari komponen yang sudah ada.
f. Fleksibel, sehingga mudah dimodifikasi.
g. Dihubungkan (*linked*) dengan sistem komputer

Pemahaman dalam memperoleh pemahaman tentang *software* (dan akhirnya pemahaman tentang software engineering), penting dalam memeriksa karakteristik perangkat lunak yang berbeda dalam pembangunan perangkat lunak. Perangkat lunak dibangun, proses kreatif manusia (analisis, desain , konstruksi, pengujian) akhirnya diterjemahkan ke dalam bentuk fisik. Pembangunan sebuah komputer baru dengan sketsa menggunakan prototipe yang dapat berkembang menjadi produk fisik.

Sistem perangkat lunak merupakan sistem yang abstrak dan tidak terwujud. Perangkat lunak tidak dibatasi bahan dengan proses manufaktur, sehingga menyederhanakan perangkat lunak itu sendiri, karena tidak ada batas untuk potensi perangkat lunak. Namun karena kurangnya kendala fisik, sistem perangkat lunak dapat cepat menjadi sangat kompleks, sulit dipahami, dan biaya yang mahal untuk mengubah suatu perangkat lunak tersebut. (Sommerville, 2011)

Sistem perangkat lunak memiliki berbagai macam jenis dari sistem yang sederhana sampai sistem yang komplek. Penelitian yang dilakukan bermanfaat untuk mencari notasi yang universal, metode, atau teknik untuk perangkat lunak karena berbagai jenis perangkat lunak memerlukan pendekatan yang berbeda. (Sommerville, 2011)

Proyek sistem perangkat lunak masih banyak yang mengalami kesalahan sistem maupun kegagalan dari perangkat lunak itu sendiri. Kegagalan suatu perangkat lunak adalah konsekuensi dari dua faktor, yaitu:

Harapan yang terlalu tinggi untuk meningkatkan tuntutan sebagai teknik perangkat lunak baru yang dapat membantu untuk membangun suatu perangkat lunak yang lebih besar, sistem yang lebih kompleks, dan tuntutan yang seringkali berubah. Sistem perangkat lunak harus dibangun dan disampaikan lebih cepat, lebih besar, bahkan sistem yang lebih komplek yang diperlukan. Sistem perangkat lunak juga harus memiliki kemampuan baru yang sebelumnya dianggap mustahil.

Harapan yang sangat rendah yang relatif mudah untuk menulis program komputer tanpa menggunakan metode rekayasa perangkat lunak. Banyak perusahaan yang mengolah perangkat lunak sebagai produk dan layanan mereka, namun perangkat lunak tersebut tidak digunakan dalam pekerjaan mereka sehari – hari, akibatnya perangkat lunak yang dikembangkan tidak dapat diandalkan dari yang seharusnya.

Perangkat lunak yang baik harus dapat memberikan fungsi dan kinerja yang diperlukan pengguna dan harus dipertahankan, diandalkan, serta dapat diandalkan. Sementara proyek perangkat lunak harus dikelola secara professional dan dikembangkan. Misalnya peragkat lunak dapat dikembangkan dengan menggunakan serangkaian prototipe.

**g. Kebutuhan Perangkat Lunak**

Menurut kamus Webster seperti dikutip oleh Davis (1993), kebutuhan adalah sesuatu yang diisyaratkan, sesuatu yang diinginkan

atau diperlukan. Sedangkan menurut Sommerville (2011) kebutuhan perangkat lunak adalah kondisi, kriteria, syarat atau kemampuan yang harus dimiliki oleh perangkat lunak untuk memenuhi apa yang diisyaratkan atau diinginkan pemakai.

Pengindentifikasian kebutuhan pemakai dilakukan dengan menghasilkan informasi yang diperoleh masih belum terstruktur. Biasanya pemakan akan mengungkapkan apa yang diinginkan dengan bahasa sehari – hari yang biasa digunakan. Kemudian pada tahap ini, kebutuhan pemakai yang belum terstruktur tersebut akan dianalisis, diklasifikasikan dan diterjemahkan menjadi kebutuhan fungsional, antarmuka dan unjuk kerja perangkat lunak. Sebagai contoh, kebutuhan data yang dimasukan oleh bagian penjualan bisa langsung diolah.

**h. Konsistensi**

Konsistensi manajemen antara artefak yang berbeda dalam rekayasa perangkat lunak telah diakui sebagai suatu hal yang penting selama bertahun-tahun. Dalam kebutuhan perangkat lunak, manajemen konsistensi antara spesifikasi kebutuhan perangkat lunak, arsitektur dan desain model telah diinvestigasi. Demikian pula, beberapa pendekatan telah dikembangkan untuk mencoba dan menentukan inkonsistensi antara deskripsi dalam *natural language* pada kebutuhan perangkat lunak. Beberapa teknik telah dikembangkan untuk mendukung koreksi inkonsistensi seperti penggunaan operasi perbaikan. (Black, 2000)

Sebuah model konsistensi termasuk antarmuka, bagian kontrol dan representasi untuk pendeskripsian dalam suatu sistem perangkat lunak dan spesifikasi dari suatu konsistensi yang relevan untuk sebuah sistem perangkat lunak. Sehingga memberikan gambaran tentang hubungan konsistensi dalam sistem perangkat lunak yang dirancang.

Konsistensi mengacu pada situasi dimana spesifikasi dari suatu perangkat lunak. Beberapa jenis konsistensi berlaku untuk semua spesifikasi sebagai suatu fungsi yang dapat dihasilkan.

Beberapa teknik dalam menggambarkan penanganan konsistensi dalam sistem perangkat lunak yang diberikan antara lain : (Kamalrudin, 2010)

1) *Grammatical Consistency*

Menulis kebutuhan perangkat lunak tidak seperti bentuk lain dari sebuah tulisan biasa, dokumen kebutuhan peragnkat lunak tidak dapat di variasikan dengan sembarang baahasa pemrograman. Pembelajaran menulis di sekolah mengajarkan agar tidak berulang kali menggunakan kata yang sama, tetapi untuk menggunakan sinonim untuk kata itu seperti : berjalan, melenggang, melangkah. Sinonim tersebut memungkinkan untuk membuat citra dan membantu pembaca agar terhindar dari kebosanan. Pembelajar di sekolah mendorong untuk menggunakan metafora yang menciptakan citra yang lebih kuat seperti: terbang, mampat, kacau, menari-nari. Metafora yang memungkinkan untuk menanamkan emosi kedalam cerita dan membantu menulis yang lebih hidup dalam membaca.

Kebutuhan perangkat lunak bukan merupakan sebuah narasi, tujuan utama yang dibangun tidak sekedar hiburan, tetapi penjelasan yang benar-benar jelas. Narasi kebutuhan perangkat lunak tidak diperkenankan menggunakan berbagai istilah untuk menggambarkan objek atau tindakan yang sama atau menggunakan istilah-istilah yang sama secara berulang-ulang. Penggambaran objek perangkat lunak tidak diperkenankan melakukan variasi dalam perubahan istilah kecuali ketika secara eksplisit membagi konteks dan item yang direferensikan dan kemudian memastikan bahwa penggunaan perubahan yang sama secara konsisten untuk konteks dan referensi yang sama. Kebutuhan perangkat lunak berhadapan dengan kepemilikan beberapa kompleksitas, menyediakan gambar atau diagram (mungkin dalam *Apendiks* atau daftar istilah) yang menjelaskan makna eksplisit setiap istilah dan istilah pengubah. Kebutuhan perangkat lunak untuk sistem klasifikasi untuk karya-karya tertentu, lebih spesifik tentang definisi penulis, editor, penerbit dan resensi. Pendefinisikan sebuah istilah yang digunakan secara kolektif yang bekerja secara konsisten menggunakan kata-kata yang sama dan idealnya menggunakan kalimat aktif.

2) *Logical Consistency*

*Logical inconsistensy* umumnya berasal dari mengekspresikan kebutuhan perangkat lunak yang sama 2 kali. Kata – kata yang tepat untuk mengekspresikan persyaratan 2 kali, seringkali kata yang digunakan akan berlebihan dan berulang-ulang dalam pengungkapannya. Ekspresi kedua dari kebutuhan perangkat lunak tidak disengaja dan terlihat berbeda dalam pengambilan risiko bahwa itu akan ditafsirkan berbeda sehingga menjadi tidak konsisten.

Konsistensi kebutuhan perangkat lunak tidak sesuai dengan logika jika terjadi kesalahan dalam pembuatan sistem yang ada. Konsisten kebutuhan tidak sesuai dapat langsung terjadi seperti menulis "sistem harus menyimpulkan nilai..." dan "pengguna harus memberikan nilai..." atau kebutuhan perangkat lunak dapat langsung tidak konsisten seperti "hewan itu menggonggong" dan "hewan adalah kucing" yang merupakan secara logika tidak konsisten karena kucing tidak menggonggong (sehingga membuat setidaknya salah satu kebutuhan perangkat lunak inkonsisten).

3) *Strategic Consistency*

Salah satu elemen yang mendefinisikan kebutuhan perangkat lunak adalah bahwa elemen kebutuhan perangkat lunak "mendukung strategi bisnis." Dekomposisi model dalam konteks nilai telah diciptakan untuk memberikan nilai dan mewujudkan strategi atau komponen dari sebuah strategi.

Sebuah strategi tidak selalu didefinisikan oleh satu ukuran. Strategi dapat mencakup metrik keuangan dan sasaran, sebagai pernyataan misi mengenai tingkat pelayanan pelanggan dan investasi jangka panjang yang diharapkan untuk yang pada akhirnya adalah penghargaan jangka panjang. Kebutuhan perangkat lunak sementara tidak hanya benar-benar mendefinisikan mekanisme untuk mencapai sebuah nilai, juga harus konsisten dengan unsur-unsur lain dari strategi yang dimiliki.

Kurangnya suatu informasi tidak harus diambil keuntungannya ketika integritas dimiliki, terlepas dari seberapa efektif yang mungkin pada rapat keuangan. Kebutuhan perangkat lunak yang diartikulasikan bertujuan untuk memaksimalkan pendapatan akan konsisten dengan tujuan beroperasi pada integritas.

**i. Inkonsistensi**

Inkonsistensi adalah masalah yang dapat terjadi dalam suatu perangkat lunak. Aturan umum pengelolaan inkonsistensi dan pendekatan berbasis kasus khusus untuk menangani inkonsistensi dengan menyajikan model proses untuk menangani persyaratan inkonsistensi dalam kerangka sudut pandang. Dalam model proses ini, ketika sebuah inkonsistensi antara sudut pandang terdeteksi, penanganan inkonsistensi akan dihasilkan dengan menggunakan teknik multi-agent. (Nuseibeh, 2003)

Manajemen persyaratan inkonsistensi adalah kunci untuk pengembangan sistem perangkat lunak yang digunakan., dengan pengukuran yang tepat sebagai suatu persyaratan inkonsistensi yang benar.

Inkonsistensi menunjukan situasi dimana 2 deskripsi tidak mematuhi beberapa hubungan dari suatu perangkat lunak. Hubungan antara deskripsi dapat dinyatakan sebagai aturan konsistensi, terhadap deskripsi yang dapat diperiksa. Dalam kenyataannya, suatu perangkat lunak mempunyai aturan konsistensi dalam berbagai dokumen proyek, baik yang tertanam dalam perangkat maupun yang tidak tertanam dimanapun.

Inkonsistensi merupakan masalah dalam rekayasa perangkat lunak jika mengarah ke kesalahpahaman dan kesalahan. Namun, masalahnya bukan dengan inkonsistensi, tetapi kadangkala penyebab beberapa inkonsistensi terjadi karena didasarkan pada struktur modul yang menyebabkan perubahan dalam penggunaan manual. Inkonsistensi terjadi akibat melakukan perubahan dan semua perubahan konsekuen sebagai tindakan atom tunggal, sehingga ketika mendeskripsikan suatu perangkat lunak dikatakan inkonsisten.

Beberapa perkembangan perangkat lunak inkonsistensi yang dapat ditoleransi, diantaranya adalah :

1) Inkonsistensi dapat menunjukan penyimpangan dari model proses. Alasannya karena untuk pemodelan proses adalah bahwa hal inkonsistensi dapat memfasilitasi proses perbaikan. Cugola (1996) berpendapat bahwa hal inkonsistensi dapat dicapai dengan memungkinkan penyimpangan dari proses yang ditentukan dan dengan memberikan dukungan untuk berurusan dengan inkonsistensi yang dihasilkan.
2) Inkonsistensi dapat memfasilitasi fleksible kerja kolaboratif. Schwanke dan Kaiser (1998) mempertimbangkan masalah pembangunan inkremental dengan adanya inkonsistensi. Schwanke dan Kaiser berpendapat bahwa mencoba untuk menegakan konsistensi total yang sulit dan karena hal tersebut memungkinkan inkonsistensi terjadi, dan untuk menyelesaikannya secara berkala daripada mencegah terjadinya inkonsistensi.

3) Inkonsistensi dapat digunakan untuk mengidentifikasi area ketidakpastian. Jika tim pengembang dapat bekerja sama dalam suatu deskripsi, maka inkonsistensi dapat ditunjukan dalam suatu area perangkat lunak. Easterbrook (1996) menjelaskan bagaimana sebuah kasus dapat digunakan untuk meningkatkan pemahaman dalam tim.

Berikut adalah pembagian ukuran dari inkonsistensi kebutuhan perangkat lunak :

1. *Formula-centric measures*

   Langkah-langkah *Formula-centric measures* diperlukan memperhitungkan jumlah inkonsistensi : lebih sedikit formula berarti lebih tinggi tingkat inkonsistensi.
2. *Atom-centric measures*

   Langkah-langkah *Atom-centric measures* memperhitungkan porsi dari bahasa yang dipengaruhi oleh inkonsistensi : lebih proposisional variabel yang terlibat dalam inkonsistensi berarti lebih tinggi tingkat inkonsistensi.

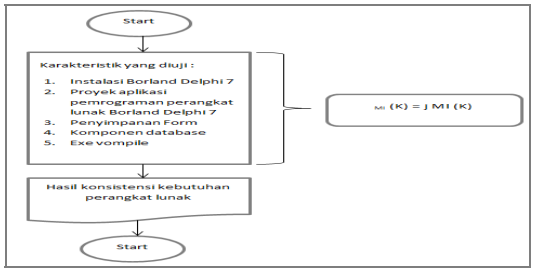
Pengembangan sistem perangkat lunak merupakan kegiatan yang kompleks dan memerlukan waktu yang lama dan melibatkan kolaborasi dari banyak pihak. Pengembangan sistem menyebabkan banyak model parsial dari perkembangan siste. Model parsial dapat menjadi inkonsistensi satu sama lain karena menggambarkan sistem dari perspektif yang berbeda dan mencerminkan pandangan dari pemangku kepentingan yang terlibat dalam model parsial. Model perangkat lunak yang inkonsistensi dapat memiliki efek negatif maupun positif dalam pengembangan siklus perangkat lunak. Inkonsistensi dapat memfasilitasi identifikasi beberapa aspek dari sistem yang dianalisis, membantu dengan spesifikasi alternatif untuk pengembangan sistem dan mendukung elisitasi informasi.

### 2.1. Implementasi Sistem dan Hasil

#### a. Analisis Inkonsistensi Menggunakan *Formula Centric Measure*

Hasil penelitian mengetahui inkonsistensi pada perangkat lunak dengan langkah-langkah yang telah dilakukan dengan mengamati ukuran inkonsistensi kedasar keyakinan secara keseluruhan. Inkonsistensi pada perangkat lunak yang individual namun dibandingkan pada perangkat keras dengan persamaan spesifikasi dan dengan sistem operasi yang berbeda.

Hasis analisis dengan menggunakan *formula centric measure* seperti yang digambarkan pada gambar 4 :



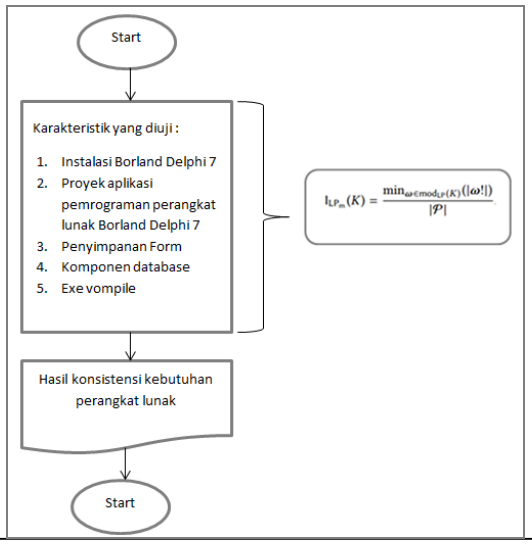

Gambar 4 Hasil analisis *formula centric measure*

Gambar 4 menggambarkan bahwa |MI| merupakan dasar tingkat inkonsistensi ukuran untuk nilai K, sehingga dapat mengetahui penyebab inkonsistensi dari perspektif yang dilakukan.

#### b. Analisis Inkonsistensi Menggunakan *Atom Centric Measure*

Berdasarkan hasil pengamatan yang dilakukan untuk mengidentifikasi model perangkat lunak Borland Delphi 7 yang dibandingkan pada perangkat keras dengan spesifikasi yang sama tetapi menggunakan sistem operasi yang berbeda maka keseluruhan pendeskripsian proses dilakuakn kebijakan yang sama pada sistem operasi yang berbeda.

Hasil analisis dengan menggunakan *atom centric measure* seperti yang digambarkan pada gambar 5 :



Gambar 5 Hasil analisis *atom centric measure*

Penelitian yang dilakukan untuk memeriksa kesalahan terhadap aturan konsistensi dengan model perangkat lunak. Hasil yang didapatkan ketika perangkat lunak Borland Delphi 7 diinstalasi pada sistem operasi Windows 8 dan windows 10 dengan ketentuan pengisntalasian yang sudah benar adalah terdapat kesalahan instalasi, hal ini mengakibatkan perangkat lunak Borland Delphi 7 tidak dapat dijalankan pada sistem operasi tersebut.

Variable yang terlibat dalam penambahan proyek aplikasi, penyimpanan form, komponen database, dan exe compile tidak dapat dilanjutkan karena proses instalasi Borland Delphi 7 tidak dapat dijalankan pada sistem operasi Windows 8 dan windows 10, sehingga menyebabkan perangkat lunak Borland Delphi 7 inkonsisten terhadap kebutuhan perangkat lunak.

Proses instalasi perangkat lunak Borland Delphi 7 juga dilakukan pada sistem operasi windows 7. Hasil yang didapatkan adalah tidak terjadi kesalahan pada saat proses instalasi, namun ada beberapa variable yang harus diganti penamaannya sehingga perangkat lunak Borland Delphi 7 ini dapat digunakan pada sistem operasi windows 7.

Instalasi perangkat lunak Borland Delphi 7 pada sistem operasi windows 98, windows xp dan windows 2000 berjalan dengan lancar dan sesuai dengan prosedur penginstalasian kebutuhan perangkat lunak. Proyek aplikasi yang dibuat dapat ditambahkan dengan menggunakan komponen yang ada pada perangkat lunak Borland Delphi 7. Proses penyimpanan form sehingga menghasilkan file exe compile tidak mengalami masalah dan ukuran dari penyimpanan tersebut tidak terlalu besar.

Proses pengidentifikasian inkonsistensi perangkat lunak Borland Delphi 7 juga dilakukan pada windows vista, instalasi perangkat lunak dapat dilakukan, namun diperlukan penambahan beberapa komponen dari command prompt, karena perangkat lunak Borland Delphi 7 tidak resmi pada windows vista.

Sumber inkonsistensi adalah set elemen model perangkat lunak yang telah digunakan dalam penelitian yang dilakukan menunjukan pada perangakat keras tertentu telah terjadi inkonsisten kebutuhan perangkat lunak yang digunakan.

Penanganan inkonsistensi telah dianggap sebagai kegiatan pusat dalam inkonsistensi manajemen. Kegiatan penanganan inkonsistensi berkaitan dengan :

1) identifikasi dari tindakan mungkin untuk berurusan dengan suatu inkonsistensi
2) evaluasi biaya dan akibat yang akan timbul dari aplikasi dari setiap tindakan ini
3) evaluasi risiko yang akan muncul dari tidak memecahkan ketidakkonsistenan.

Penelitian yang dilakukan berkaitan dengan rekaman adalah pendukung deteksi inkonsistensi, sumber, penyebab dan dampak inkonsistensi, menangani tindakan yang dianggap sehubungan dengan inkonsistensi dan argumen yang mendasari keputusan untuk memilih salah satu dari pilihan kebutuhan perangkat lunak yang diteliti konsisten ataupun inkonsistensi.

### 3.1. Kesimpulan

Menentukan inkonsistensi pada kebutuhan lunak dapat dilihat dari membandingkan perangkat lunak yang digunakan ke dalam beberapa sistem operasi. Berdasarkan perbandingan perangkat lunak Borland Delphi 7 pada sistem operasi yang berbeda didapatkan tidak konsistennya kebutuhan perangkat lunak dikarenakan terlalu tinggi spesifikasi yang muncul dalam sistem operasi, sehingga mengakibatkan tidak sesuainya sistem dengan keinginan pengguna sehingga permasalahan dari pengguna tidak dapat diselesaikan. Oleh sebab itu, perangkat lunak Borland Delphi yang digunakan untuk windows 7 ke atas sebaiknya adalah versi terbaru yang konsisten terhadap kebutuhan dari segi sistem operasi.

### Daftar Pustaka

[1] Acharya, S., & George, C. (2005). Domain Consistency in Requirements Specification, *414*(9).

[2] A. Russo, B. A. Nuseibeh, and J. Kramer, “Restructuring requirements specifications for inconsistency analysis: A case study”, Proceedings of 3rd International Conference on

Requirements Engineering (ICRE98), 51-60, Colorado Springs, USA, IEEE Computer Society Press

[3] BSSC (1995), Guide to the user requirements definition phase, Perancis.

[4] Demarco, Sharon (1993), Identifying Important Issues.

[5] F. Schneider, S. M. Easterbrook, J. R. Callahan, and G. J. Holzmann, "Validating Requirements for Fault Tolerant Systems using Model Checking",Proceedings of 3rd International Conference on Requirements Engineering (ICRE-98), 4-13, Colorado Springs, USA, IEEE Computer Society Press, 6-10 April 1998

[6] G. Cugola, E. Di Nitto, A. Fuggetta, and C. Ghezzi, "A Framework for Formalizing Inconsistencies and Deviations in

[7] Human-Centered Systems",Transactions on Software Engineering and Methodology, 5(3):191-230, ACM Press, July 1996

[8] Hunter, Anthony (2010), On the measure of conflicts: Shapley Inconsistency Values, London.

[9] Kadir, Abdul, 2002, Pemrograman Database dengan Delphi 7 Menggunakan Access dan ADO, penerbit : Andi Publisher

[10] Kamalrudin, M. (2009). Automated Software Tool Support for Checking the Inconsistency of Requirements. 2009 IEEE/ACM International Conference on Automated Software Engineering, 693–697. http://doi.org/10.1109/ASE.2009.38

[11] Kamalrudin, Massila (2010), Managing Consistency between Textual Requirements, Abstract Interactions and Essential Use Cases, Selandia Baru.

[12] Kroha, P., Janetzko, R., & Labra, J. E. (2009). Ontologies in Checking for Inconsistency of Requirements Specification. 2009 Third International Conference on Advances in Semantic Processing, 32–37. http://doi.org/10.1109/SEMAPRO.2009.11

[13] Martina, Inge, 2002, 36 jam belajar komputer pemrograman internet dengan delphi, Penerbit : Elex Media Komputindo

[14] Mu, K., Jin, Z., & Zowghi, D. (2008). A Measurement-Driven Process Model for Managing Inconsistent Software Requirements. 2008 15th Asia-Pacific Software Engineering Conference, 291–298. http://doi.org/10.1109/APSEC.2008.24

[15] Pressman, Roger, S, 1997, Rekayasa Perangkat Lunak : Pendekatan Praktisi (Edisi Satu), Penerbit : Andi, Yogyakarta

[16] R. W. Schwanke and G. E. Kaiser, "Living With Inconsistency in Large Systems", Proceedings of Proceedings of the International Workshop on Software Version and Configuration Control, 98-118, Grassau, Germany, B. G. Teubner,

[17] Stuttgart, 27-29 January 1988

[18] Sommerville (2007), Software Engineering, 8th edition, Pearson Education Limited, England

[19] Spanoudakis, George (2000), Inconsistency Management in Software Engineering: Survey and Open Research Issues, Inggris

[20] Wahana Komputer. 2002. Pemrograman Borland Delphi 7.0. Yogyakarta. C.V. Andi Offset

[21] Yikun, Z., Peng, Y., Duwu, C., & Hui, X. (2007). A Method of Requirement Inconsistency Analysis. 31st Annual International Computer Software and Applications Conference - Vol. 1- (COMPSAC 2007), (Compsac), 211–214. http://doi.org/10.1109/COMPSAC.2007.28

[22] Yu, Lian (2008), Completeness and Consistency Analysis on Requirements of Distributed Event-Driven Systems, China.

[23] Zhu, Xuefeng (2012), Generating Software Requirements Consistency Checking Patterns: An Environment Ontology*-driven Approach*. China.