Pengembangan Onti Measures Berbasis Web dengan Pengujian Data *Ontology* Virus dan Penyakit

(*Web-based Onti Measures Development with Virus and Disease Ontology Data Testing*)

Nur Alfi Ekowati[1], Ika Indah Lestari[2], Sulistiyasni[3]

***Abstract*—The use of the ontology document in the COVID-19 case is an example where the inconsistency measure for OWL ontologies is undoubtedly needed. Ontology is a knowledge representation of semantic web technology, which is the extension of a website. The role of inconsistency measure is essential in ensuring that all information on an ontology is consistent. This research aims to develop a web-based application program, namely Onti Measures, which is an expansion of the inconsistency measures program prototype that has been created in the previous research. That prototype has multiple weaknesses, such as it is only in the form of program codes with no user interface so that the public cannot access this program. The data collection method in this research was done through literature study, while the waterfall method was used as the system development method. This research's testing sample was ontology files for virus and disease cases served as the input of Onti Measures using 3 types of OWL reasoners. The program's outputs were the information of inconsistency values, running times, and the ontology sizes. The testing was done by employing the whitebox and blackbox testing methods.**

***Intisari*—Penggunaan dokumen *ontology* pada kasus COVID-19 adalah sebuah contoh begitu dibutuhkannya *inconsistency measure* untuk *OWL ontology*. *Ontology* adalah representasi pengetahuan pada teknologi web semantik yang merupakan ekstensi dari *website*. Peran *inconsistency measure* menjadi penting untuk memastikan bahwa seluruh informasi pada sebuah *ontology* konsisten. Penelitian dalam makalah ini bertujuan membangun program aplikasi bernama Onti Measures berbasis web yang merupakan pengembangan dari sebuah prototipe program *inconsistency measures* berbasis *ontology* yang telah dibuat pada penelitian sebelumnya. Prototipe tersebut memiliki beberapa kelemahan, di antaranya prototipe hanya berupa kode program dan tidak ada antarmuka pengguna, sehingga program tidak dapat diakses oleh umum. Pengumpulan data dilakukan melalui studi pustaka, sedangkan metode pengembangan sistemnya adalah metode *waterfall*. Sampel pengujian dalam makalah ini adalah berkas *ontology* kasus virus dan penyakit yang dijadikan sebagai masukan program Onti Measures dengan pemakaian tiga jenis *OWL reasoner*. Keluaran program adalah informasi nilai inkonsistensi, *running time*, beserta ukuran *ontology*. Pengujian dilakukan dengan metode *whitebox testing* dan *blackbox testing*.**

***Kata Kunci*—Onti Measures, Web, *Inconsistency Measures*, *Ontology*, Virus, Penyakit.**

[1,2,3] *STMIK Widya Utama, Jalan Sunan Kalijaga, Berkoh, Kec. Purwokerto Selatan, Kab. Banyumas, Prov. Jawa Tengah 53146 INDONESIA (telp: 0281-6512290; e-mail: [1]nuralfiekowati@swu.ac.id, [2]ikaindah22@swu.ac.id, [3]sulistiyasnipwt@swu.ac.id)*

I. PENDAHULUAN

Ontology adalah kumpulan informasi (*Knowledge Base*, KB) yang terkumpul menjadi satu kesatuan secara terminologi untuk merepresentasikan sebuah kasus pada era kini. Secara khusus, *ontology* adalah representasi pengetahuan pada teknologi web semantik yang merupakan ekstensi dari *website*. Bahasa representasi dari *ontology* biasa disebut *Web Ontology Language* (OWL) yang didasarkan pada *Description Logics* (DL). Referensi [1] menyatakan bahwa *ontology* pada aplikasi sering kali dikompilasi oleh lebih dari satu orang. Hal ini memungkinkan adanya data yang tidak konsisten. *Ontology* yang tidak konsisten dapat merusak informasi yang diakses untuk tujuan tertentu. Inkonsistensi juga dapat terjadi akibat sebuah konstruksi otomatis, misalnya sebuah *ontology* dibangun secara otomatis karena merupakan hasil keluaran sebuah *text mining*. Melalui proses konstruksi tersebut, *ontology* yang inkonsisten bisa saja terjadi.

Salah satu contoh terkait pentingnya konsistensi *ontology* adalah pada kasus virus COVID-19 (*corona virus disease* yang ditemukan pada tahun 2019), yang saat ini tengah menjadi pandemik. Berdasarkan informasi data global dari *World Health Organization* (WHO), hingga tanggal 27 Agustus 2021 COVID-19 telah memakan korban sebanyak 214.468.601 orang di seluruh dunia [2]. Berdasarkan banyaknya kasus COVID-19 yang terjadi dalam rentang waktu lebih dari satu tahun ini, telah jelas bahwa COVID-19 adalah jenis virus sekaligus penyakit yang sangat berbahaya. COVID-19 dapat menyebabkan kematian dan sebarannya menjadi semakin serius dari hari ke hari. Itulah sebabnya jenis virus ini harus segera ditangani.

Jika penanganan masalah pandemi COVID-19 mengandalkan dokumen *ontology* untuk mengecek informasi tertentu terkait COVID-19, maka data tersebut harus benar-benar akurat. Sebagai contoh, informasi yang dimaksud adalah data detail seluruh korban yang terkena kasus COVID-19, tingkat keparahan penyebaran virus COVID-19, dan data detail uji klinis vaksin COVID-19. Jika data *ontology* sebagai andalannya memiliki informasi yang tidak konsisten, akan lebih banyak nyawa manusia yang menjadi taruhannya. Pada kasus seperti inilah *inconsistency measure* menjadi sangat penting dan sangat dibutuhkan. Urgensi adanya *inconsistency measure* menjadi jelas sekali, yaitu untuk memastikan bahwa seluruh informasi pada sebuah *ontology* sudah konsisten.

Beberapa penelitian tentang *inconsistency measure* telah dilakukan [3]-[6]. Hanya saja, berbagai jenis *measure* tersebut bukan untuk mengukur *ontology*. Kemudian, *inconsistency*

measure untuk *ontology* berbasis *axiom* juga telah diciptakan [1], lalu dilanjutkan oleh penelitian yang menyertakan pembuatan prototipe program aplikasi perangkat lunak bernama Axiomi Measures untuk *inconsistency measure* tersebut [7]. Sementara itu, *inconsistency measure* untuk *OWL ontology* berbasis *ontology* telah ditemukan pula [8].

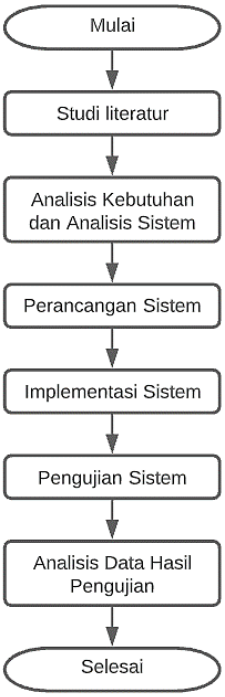
Tujuan penelitian dalam makalah ini adalah membangun program aplikasi berbasis web bernama Onti Measures. Sebelumnya, prototipe dari Onti Measures telah dibangun [8], tetapi masih memiliki kelemahan-kelemahan sebagai berikut.

- Prototipe hanya berupa kode program dan tidak memiliki antarmuka pengguna *(user interface)* sehingga tidak dapat diakses oleh umum.
- Nilai inkonsistensi *(inconsistency value)* yang dihasilkan pada prototipe belum menggunakan pembulatan nilai desimal, sehingga memungkinkan adanya angka yang panjang di belakang koma dari nilai tersebut.
- Prototipe memiliki *running time* yang kurang akurat dari segi presisi detiknya, sehingga memungkinkan adanya data *running time* 0 detik. *Running time* adalah lama waktu yang diperlukan untuk menjalankan Onti Measures dengan masukan yang disediakan.
- Prototipe memiliki kode program yang belum terstruktur, sehingga *maintenance* ataupun pengembangan program di masa depan akan lebih susah.
- Prototipe hanya menggunakan dua jenis *OWL reasoner*, yaitu HermiT dan JFact, sehingga komparasi hasil keluarannya kurang begitu variatif.
- Prototipe perangkat lunak belum memiliki nama, sehingga pemanggilannya akan lebih susah.

Berdasarkan kelemahan-kelemahan tersebut, Onti Measures berbasis web dibangun dengan mengembangkan prototipe yang sudah ada. Nama Onti Measures terinspirasi oleh kependekan kata dari *ontology-based inconsistency measures.* Program aplikasi berbasis web ini merupakan sebuah paket *measure* yang berisi beberapa jenis perhitungan *inconsistency measure* untuk *OWL ontology* berbasis *ontology* yang menghitung nilai inkonsistensi seluruh bagian KB (*ontology*). Ini berkebalikan dengan *inconsistency measure* untuk *ontology* berbasis *axiom* yang didefinisikan sebagai pengukur inkonsistensi *ontology* yang menghitung nilai inkonsistensi setiap *axiom* pada KB (*ontology*) tersebut. Onti Measures dibatasi hanya untuk pengukuran *OWL ontology* dengan bahasa OWL 2 atau subbahasanya. OWL 2 adalah bahasa *ontology* yang didasarkan pada DL SROIQ.

Setelah program aplikasi tersebut dibangun, program diuji dengan tiga *OWL reasoner*, yaitu HermiT, JFact, dan Pellet, serta menggunakan *test cases* eksperimen berupa data *ontology* dengan bahasa OWL (*OWL ontology*). Khusus dalam hal ini, domain virus dan penyakit menjadi *test cases* pengujian tersebut. Kemudian, dilakukan juga analisis hasil uji program Onti Measures tersebut.

Manfaat inti makalah ini adalah menyajikan sebuah media berupa kumpulan *inconsistency measure* untuk *ontology* yang dapat dipakai secara terbuka guna mengukur tingkat inkonsistensi suatu *ontology*. Secara khusus, makalah ini menyajikan data hasil pengujian inkonsistensi *ontology* beberapa jenis virus dan penyakit, sehingga hasil penelitian dapat dirasakan manfaatnya pula oleh lintas bidang ilmu, seperti bidang *biomedical informatics* yang menggabungkan ilmu kedokteran dan informatika.



Gbr. 1 Diagram alir penelitian.

II. METODOLOGI

Penelitian dalam makalah ini dilakukan dengan beberapa langkah, seperti tampak pada Gbr. 1. Hal pertama yang dilakukan adalah studi literatur (pustaka) yang termasuk dalam langkah pengumpulan data. Analisis kebutuhan dan analisis sistem, perancangan sistem, implementasi sistem, serta pengujian sistem sudah termasuk dalam langkah-langkah pengembangan sistem, sehingga hal tersebut otomatis harus dilakukan. Analisis data hasil pengujian perlu dilaksanakan setelah pengujian sistem selesai.

A. Metode Pengumpulan Data

Pengumpulan data dilakukan melalui teknik penelitian utama, yaitu studi pustaka. Studi pustaka merupakan metode pengumpulan data yang diambil dari kearsipan, dilakukan dengan mempelajari dan mengamati serta menganalisis berkas-berkas yang sudah ada dari berbagai sumber lainnya guna mendukung penelitian. Pada penelitian dalam makalah ini, pengumpulan data utamanya dilakukan dengan cara mempelajari tesis, artikel ilmiah, dan sumber-sumber data *OWL ontology* yang terkait dengan topik makalah.

B. Metode Pengembangan Sistem

Pengembangan sistem dalam makalah ini menggunakan metode *waterfall*, dengan tahapan-tahapan yang ada pada *Software Development Life Cycle* (SDLC) untuk membangun sebuah perangkat lunak. Tahapan-tahapan tersebut meliputi

TABEL I
JENIS *INCONSISTENCY MEASURE* PADA ONTI MEASURES

No.	Kategori	Nama *Inconsistency Measure*
1.	*Drastic inconsistency measure*	*Drastic inconsistency measure*
2.	*Minimal inconsistency-based measures*	*MI-inconsistency measure*
		MI^c-inconsistency measure
		D_f-inconsistency measure
		Problematic inconsistency measure
		Incompatibility ratio inconsistency measure
3.	*Maximal consistency-based measures*	*MC-inconsistency measure*
		The nc-inconsistency measure
4.	*Variable-based measures*	*The mv-inconsistency measure*
		ID_{MCS} inconsistency measure

analisis kebutuhan, analisis sistem, perancangan, implementasi, pengujian, dan perawatan.

C. Metode Analisis Data

Metode analisis data yang digunakan adalah kuantitatif dengan pendekatan analisis deskriptif. Metode ini digunakan karena data yang diolah adalah data yang berhubungan dengan angka atau kuantitas. Selain itu, metode ini juga bergantung pada kemampuan untuk menghitung secara akurat. Sementara itu, pendekatannya adalah pendekatan analisis deskriptif karena data pada makalah ini memerlukan analisis yang bekerja dengan menggambarkan distribusi data. Hasil diperoleh dari perhitungan variabel-variabel penelitian lalu dipaparkan secara tertulis.

III. ANALISIS KEBUTUHAN DAN ANALISIS SISTEM

A. Analisis Kebutuhan

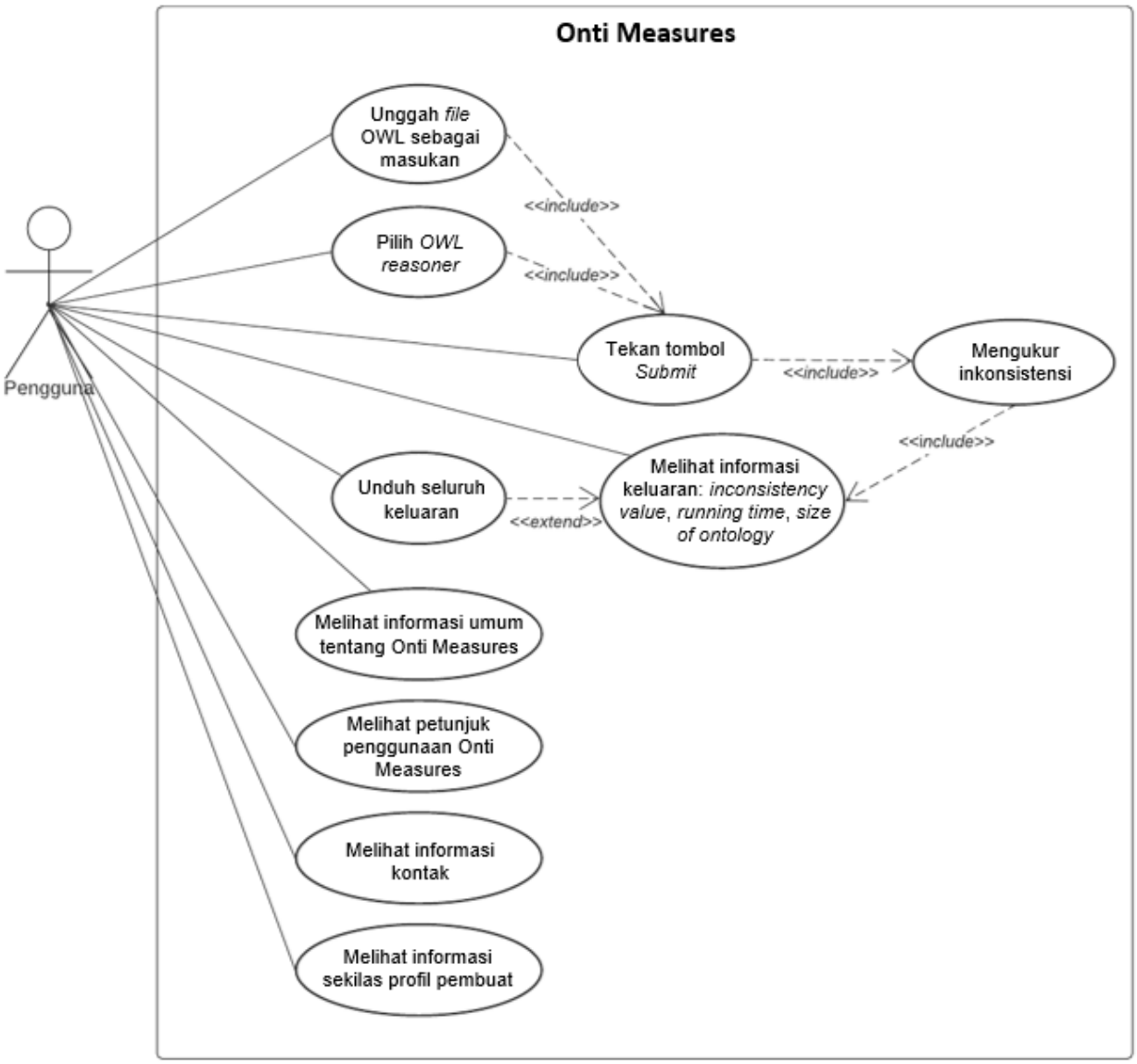
Analisis kebutuhan perangkat lunak dilakukan untuk mendapatkan informasi dan spesifikasi yang diperlukan untuk membangun aplikasi. Tahap ini dilakukan dengan mencari informasi untuk pemilihan teknologi yang tepat guna pengembangan aplikasi web Onti Measures. Selain itu, spesifikasi perangkat keras yang akan digunakan juga ditentukan pada tahap ini.

Perangkat lunak yang digunakan adalah sistem operasi Windows, Eclipse Jee Oxygen sebagai *Integrated Development Environment* (IDE), *cloud platform* Heroku, Command Prompt, dan New Relic. Secara garis besar, *framework* dan *library* yang digunakan meliputi Spring Boot, OWL API, HermiT, JFact, Pellet, Thymeleaf, dan Bootstrap. Berbagai *dependency* pendukung *library*-nya meliputi Commons Collections, Telemetry, Trove4j, Aterm-java, Icu4j, Iri, Jena, Jgrapht, Jjtraveler, Pellet-common, RelaxngDataType, Shared-objects, XercesImpl, Xml-apis, Xsdlib, dan Slf4j-log4j12, sedangkan perangkat keras utama yang digunakan adalah *notebook* (laptop).

Sebagai bahan penelitian, rumus atau formula dari sepuluh *measure* menjadi bahan utama untuk pembuatan program aplikasi Onti Measures yang diterjemahkan ke dalam penulisan kode program dengan bahasa pemrograman Java. Tabel I menyebutkan nama masing-masing *measure* tersebut. Referensi [8] menjabarkan masing-masing rumusnya. Sementara itu, bahan untuk pengujiannya adalah sampel penelitian berupa berkas/dokumen (*file*) *OWL ontology*.

TABEL II
ANALISIS PIECES SISTEM

No.	Kebutuhan	Sistem Baru
1.	*Performance* (kinerja)	Sistem yang baru memiliki antarmuka pengguna sehingga sistem dapat diakses oleh umum dan dirasakan manfaatnya.
2.	*Information* (informasi)	Nilai inkonsistensi serta *running time* yang dihasilkan oleh sistem adalah informasi yang sangat penting. Adanya pembulatan desimal pada setiap nilai inkonsistensi dan presisi detik pada *running time* akan membuat informasi keluaran sistem tersebut lebih valid dan akurat.
3.	*Economy* (ekonomis)	Dari segi ekonomi, sistem akan terasa ekonomis karena bersifat *free*, yang artinya pengguna dapat mengakses sistem secara bebas (gratis).
4.	*Control* (pengendalian)	Pengembangan sistem dilakukan menggunakan perangkat lunak ataupun *tool* yang bersifat *open source* dan sumber daya manusia yang menguasai pengembangan tersebut, sehingga sistem mudah dikendalikan kapan pun dan di mana pun. Sistem juga mempunyai kode program yang lebih rapi dan terstruktur sehingga *best practice* dari implementasi kode program dapat tercapai. Pengembangan selanjutnya pun akan lebih mudah dilakukan serta tidak membingungkan.
5.	*Efficiency* (efisiensi)	Sistem terasa efisien dengan adanya antarmuka pengguna yang *friendly*, mudah diakses, dan cukup sederhana. Sistem juga dapat memberikan respons dan waktu pengolahan data yang lebih cepat sesuai fungsinya, sehingga keluaran yang dihasilkan lebih cepat pula untuk didapat.
6.	*Services* (pelayanan)	Sistem memberikan keluaran berupa informasi ukuran *ontology*, nilai inkonsistensi, *running time* dari sepuluh jenis *measure* dengan menggunakan tiga jenis *OWL reasoner*, sehingga layanan yang disajikan akan terasa lebih lengkap dengan hasil keluaran yang lebih variatif. Berbagai menu yang disajikan juga dirancang sesuai kebutuhan pengguna sehingga sistem yang diberi nama Onti Measures ini akan terasa tepat guna, tidak berlebihan, dan tidak rumit. Selain itu, seluruh keluaran program beserta keterangan *log*-nya juga dapat diunduh secara gratis.



Gbr. 2 Diagram *use case* Onti Measures.

B. Analisis Sistem

Referensi [9] menyebutkan bahwa James Wetherbe membangun sebuah *framework* yang berguna untuk mengklasifikasikan masalah. *Framework* tersebut disebut PIECES, yaitu singkatan dari *performances, information, economy, control, efficiency, services*. Analisis PIECES dapat membantu perbaikan ataupun pengembangan sistem lama dengan diadakannya sistem baru untuk Onti Measures, seperti yang dijabarkan pada Tabel II. Penjabaran tersebut diharapkan dapat menjawab kelemahan-kelemahan sistem lama yang telah diuraikan pada bagian sebelumnya.

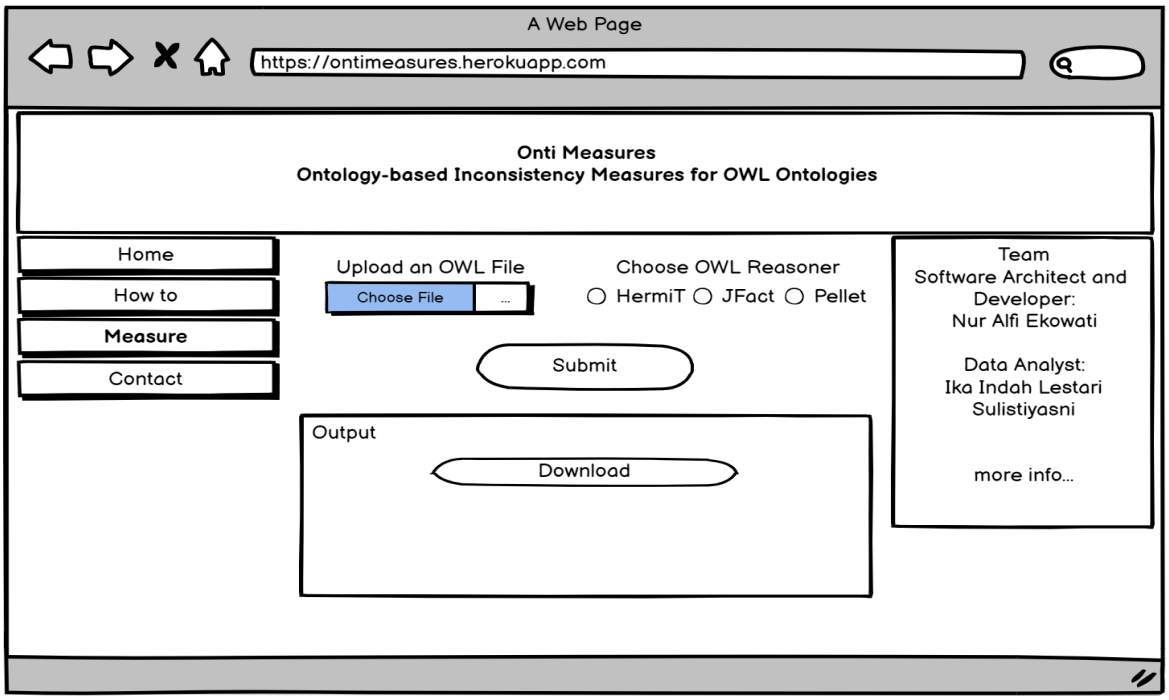
IV. PERANCANGAN DAN IMPLEMENTASI SISTEM

Perancangan sistem dilakukan dengan beberapa tahap, yaitu perancangan diagram, perancangan algoritme, dan perancangan antarmuka. Selanjutnya, tahap implementasi meliputi implementasi kode program dan implementasi antarmuka.

A. Perancangan Sistem

1) Perancangan Diagram: Pemodelan sistem yang menjabarkan aksi pada sistem dengan aksi pengguna dapat direpresentasikan melalui sebuah diagram. Secara khusus, hubungan interaksi antara sistem Onti Measures dan pengguna digambarkan melalui diagram *use case* yang ditunjukkan pada Gbr. 2.

2) Perancangan Algoritme: Pada perancangan Onti Measures terdapat minimal sebelas algoritme yang dapat dibentuk karena di dalam Onti Measures terdapat sepuluh perhitungan dari sepuluh *measure*. Satu algoritme lainnya dibuat untuk menentukan instruksi-instruksi sistem induk dari Onti Measures itu sendiri. Algoritme narasi adalah salah satu bentuk penulisan algoritme yang paling mudah dibuat. Berikut ini adalah salah satu contoh algoritme narasi untuk cara kerja dari perhitungan *drastic inconsistency measure*.



Gbr. 3 Desain antarmuka menu *Measure* di aplikasi Onti Measures.

Algoritme *Drastic Inconsistency Measure*
{Menentukan nilai inkonsistensi dari sebuah *OWL ontology*. Algoritme menerima masukan berupa berkas *ontology* dan *OWL reasoner*, kemudian mengukur nilai inkonsistensinya, *running time*, ukuran *ontology*, serta mencetak hasil tersebut.}

Deklarasi:
Inconsistency value = float
Start time, end time, total running time = long
Deskripsi:

a) Tentukan *start time*.
b) *Input* berkas *ontology*, jenis OWL *reasoner* yang dipilih.
c) Hitung ukuran/kardinal *ontology (size of ontology)*. Hitung nilai inkonsistensi *(inconsistency value)*. Jika *ontology*-nya konsisten maka nilai inkonsistensi = 0, jika tidak maka nilai inkonsistensi = 1.
d) Tampilkan nilai inkonsistensi dan ukuran *ontology*.
e) Tentukan *end time*.
f) Total *running time = end time – start time*.
g) Tampilkan *running time.*

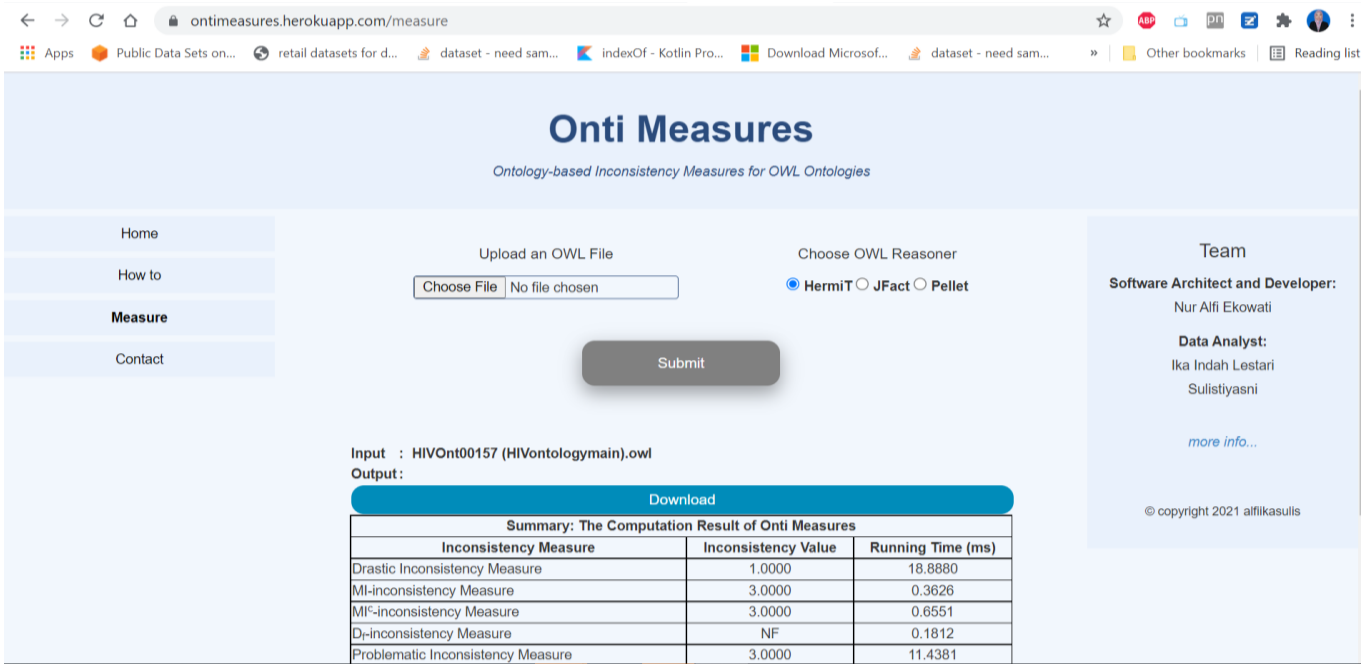
3) Perancangan Antarmuka: Antarmuka aplikasi Onti Measures dibuat sederhana agar terlihat lebih formal, sehingga desain antarmuka dalam perancangannya dibuat tidak terlalu rumit. Desain dirancang dengan mengadakan lima macam menu, yaitu *Home*, *How to*, *Measure*, *Contact*, dan *More Info*. *Home* adalah tampilan halaman awal ketika pengguna membuka aplikasi Onti Measures; *How to* adalah halaman berisi petunjuk penggunaan; serta *Measure* adalah halaman inti dari program aplikasi tempat masukan, proses perhitungan, dan keluaran Onti Measures berada. Desain tampilan menu *Measure* terlihat seperti pada Gbr. 3. Selanjutnya, *Contact* adalah halaman berisi informasi alamat kontak untuk berkomunikasi dengan pembuat aplikasi jika diperlukan dan ada satu tambahan menu pada program, yaitu *More Info*, yang berisi sekilas profil pembuat program.

B. Implementasi Sistem

1) Implementasi Kode Program: Sistem yang telah dirancang dikonversi ke dalam bahasa yang dimengerti oleh komputer. Kode program untuk membuat Onti Measures ditulis menggunakan bahasa pemrograman Java, sedangkan tampilan antarmuka pengguna Onti Measures menggunakan *HyperText Markup Language* (HTML). Kode program mengonversi formula atau rumus perhitungan sepuluh jenis *inconsistency measure* pada implementasi perhitungan Onti Measures. Masing-masing nama *measure* tersebut ditunjukkan pada Tabel I. Referensi [8] menjabarkan satu per satu rumus setiap *measure* tersebut.

Dari beberapa *measure* tersebut, terdapat empat *measure* yang mengimplementasikan metode *power set* pada kode programnya, yaitu D_f*-inconsistency measure*, *the nc-inconsistency measure*, *MC-inconsistency measure*, dan ID_{MCS} *inconsistency measure*. Secara umum, *power set* merupakan himpunan berisi seluruh himpunan bagian dari sebuah himpunan. Khusus dalam hal ini, *power set* adalah sebuah himpunan yang berisi seluruh himpunan bagian yang memungkinkan untuk dibentuk dari sebuah KB milik *ontology* yang merupakan himpunan berisi beberapa *axiom*.

Cara mendapatkan seluruh kemungkinan himpunan bagian yang bisa dibentuk adalah dengan menghitung hasil dari 2^n. sebagai contoh, ukuran sebuah *ontology* adalah 3, yang berarti KB pada *ontology* tersebut memiliki tiga *axiom*. Maka, himpunan bagian yang bisa dibentuk adalah $2^3 = 8$, sehingga nilai *power set* dari KB milik *ontology* tersebut adalah 8. Ini berarti *power set* tersebut adalah sebuah himpunan berisi delapan himpunan bagian. *Power set* yang telah dihasilkan kemudian diolah untuk menyeleksi himpunan-himpunan pada *power set* tersebut yang konsisten. Kemudian, himpunan yang



Gbr. 4 Tangkapan layar antarmuka menu *Measure* di program aplikasi Onti Measures.

konsisten tersebut diolah tergantung pada kebutuhan perhitungan masing-masing *measure* yang telah disebutkan di atas.

Selain itu, ada satu jenis *measure* pada Onti Measures yang menerapkan metode eliminasi himpunan bagian pada kode programnya. Metode tersebut dimaksudkan untuk mengeliminasi himpunan bagian dari KB *ontology* yang tidak termasuk *consistent subset*, sehingga hasil metode tersebut adalah himpunan berisi himpunan bagian yang konsisten pada KB. *Measure* tersebut adalah *MC-inconsistency measure*.

ID_{MCS} inconsistency measure merupakan sebuah *measure* yang menerapkan metode kode program khusus pula, yaitu untuk mengecek himpunan bagian KB yang masuk dalam golongan *Minimal Correction Subset* (MCS). Metode khusus lainnya yang diterapkan adalah perhitungan ukuran dari KB *ontology* dan total *running time* yang menunjukkan lama waktu yang diperlukan sebuah *measure* untuk menjalankan sebuah berkas *OWL ontology*.

Perhitungan *minimal inconsistency* diambil dari nilai *explanation* yang dihasilkan oleh masing-masing *OWL reasoner*. Karena *reasoner* yang digunakan pada Onti Measures berjumlah tiga (HermiT, JFact, dan Pellet), maka metode kode program untuk perhitungan *inconsistency* dengan masing-masing *reasoner* tersebut dibuat terpisah dan berjumlah tiga metode.

Seluruh metode yang telah disebutkan di atas masing-masing dibuat dalam *class* program tersendiri, termasuk dalam hal ini adalah implementasi tiap-tiap rumus dari sepuluh *measure*, *service* untuk mengunduh berkas keluaran program, serta *controller* utama program. *Controller* digunakan untuk memproses *request* dari *Representational State Transfer – Application Programming Interface* (REST API), menyiapkan model, dan memanggil *view* dari web yang akan dikenai proses *render* sebagai bentuk responsnya. *View* tersebut dibuat menggunakan HTML dengan jumlah berkas HTML mengikuti jumlah menu pada web.

2) Implementasi Antarmuka: Implementasi antarmuka dilakukan dengan mengonversi rancangan antarmuka ke dalam kode-kode program menggunakan HTML, *Cascading Style Sheets* (CSS), Thymeleaf, dan Bootstrap, sehingga terbentuk beberapa *view* sesuai rancangan. Salah satu menu sebagai bentuk dari implementasi ini tergambar pada tangkapan layar, seperti diperlihatkan pada Gbr. 4. Referensi [10] menampilkan menu yang dimaksud tersebut. Onti Measures dengan seluruh implementasi yang telah dilakukan di-*deploy* ke server milik *cloud platform* Heroku dan menghasilkan sebuah *website* yang halamannya dapat diakses secara bebas (gratis) pada alamat situs web *https://ontimeasures.herokuapp.com*.

V. Pengujian dan Perawatan

A. Pengujian

Sampel uji dalam makalah ini adalah *ontology* kasus virus dan penyakit yang berkas *ontology*-nya didapatkan dengan memilah berkas-berkas *ontology* dari berbagai *website* terkait bidang kedokteran ataupun biologi. *Ontology* yang sudah terkumpul kemudian dikelompokkan lagi berdasarkan jenis bahasa *ontology*-nya, termasuk OWL 2 atau bukan. Jika termasuk jenis bahasa *ontology* OWL 2, sampel terpilih tersebut dijadikan sampel pengujian program aplikasi Onti Measures. Pengambilan sampel didasarkan pada kelompok-kelompok subjek, sehingga teknik pengambilan sampel ini disebut *cluster sampling*. Tabel III menunjukkan sampel penelitian berjumlah tiga puluh buah. Referensi [11] menyebutkan bahwa jumlah batas minimal sampel yang harus diambil adalah tiga puluh sampel. Referensi [12] dijadikan

TABEL III
SAMPEL UJI PENELITIAN

No.	Nama Berkas *Ontology* (**.owl*)	Deskripsi *Ontology*
1	*abd (anthology of biosurveillance diseases)*	Kumpulan berbagai macam penyakit
2	*Alzheimer Ontology v15R-xml_merged*	Penyakit Alzheimer
3	*ao (asthma ontology)*	Penyakit asma
4	*asdpto* (*autism spectrum disorder phenotype ontology)*	Penyakit autis
5	*cvdo (cardiovascular disease ontology)*	Penyakit kardiovaskular
6	*ido covid-19 (the covid-19 infectious disease ontology)*	Penyakit/virus COVID-19
7	*ido functional updated (infectious disease ontology)*	Penyakit menular
8	*idomal (malaria ontology)*	Penyakit malaria
9	*melo (melanoma_ontology)*	Penyakit kanker kulit
10	*ocimido (ocular immune-mediated inflammatory diseases ontology)*	Penyakit inflamasi mata
11	*pdon (parkinson disease ontology)*	Penyakit Parkinson
12	*scdo (sickle cell disease ontology)*	Penyakit disorder sel darah
13	*vido (virus infectious disease ontology)*	Penyakit/virus menular
14	*Prostate Cancer Ontology*	Penyakit kanker prostat
15	*Thyroid Cancer Ontology 20200703*	Penyakit kanker tiroid
16	*Neomark Ontology*	Penyakit kanker mulut *neomark*
17	*Pneumonia Diagnosis Ontology*	Penyakit pneumonia
18	*Brain Tumour Ontology*	Penyakit tumor otak
19	*Pathogenic Disease ontology*	Patogen (virus) penyebab penyakit
10	*Nanbyo disease ontology*	Penyakit-penyakit yang susah disembuhkan dan langka
21	*Chronic Kidney disease ontology*	Penyakit ginjal kronis
22	*Partumdo (postpartum depression ontology)*	Penyakit depresi setelah melahirkan
23	*ontology of Glucose Metabolism Disorder*	Penyakit gangguan metabolisme glukosa
24	*COPD ontology*	Penyakit paru-paru *Chronic Obstructive Pulmonary Disease* (COPD)
25	*HP_O (hypersensitivity pneumonitis ontology) 2018*	Penyakit pneumonitis hipersensitif
26	*Inherited Retinal Dystrophies*	Penyakit distrofi retina turunan
27	*Inherited Retinal Dystrophy*	Penyakit distrofi retina turunan
28	*Uganda Disease*	Kumpulan berbagai macam penyakit di negara Uganda
29	*covid-19 updated*	Penyakit/virus COVID-19
30	*HIVont00157*	Virus *Human Immunodeficiency Virus* (HIV)

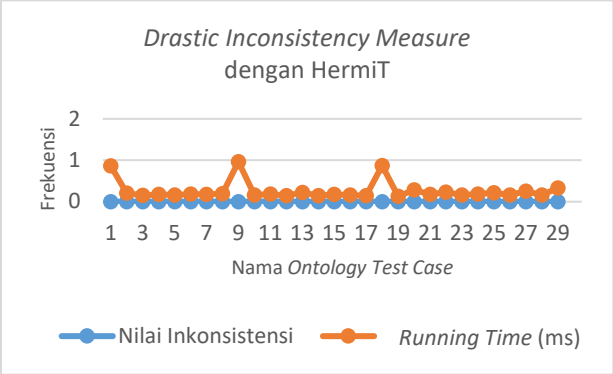
sebagai sumber didapatkannya seluruh sampel tersebut, kecuali sampel nomor 3 dan 4 [13], serta nomor 29 [14].

Pengujian terhadap Onti Measures dilakukan dengan metode *whitebox testing* dan *blackbox testing*. *Whitebox testing* disebut juga *structural testing* atau *glass box testing* [15]. Jenis pengujian ini dilakukan dengan memeriksa dan menganalisis kode program Onti Measures untuk memastikan seluruh *module* yang dibuat tidak ada yang salah. Teknik yang digunakan pada pengujian ini adalah teknik *basis path* dengan pendekatan *bottom-up*. Teknik ini dilakukan agar pengujian dapat fokus pada logika program. Sementara itu, pendekatan *bottom-up* dilakukan pada *module* yang lebih kecil terlebih dahulu, yaitu *module* pada setiap jenis *measure*, kemudian ke *module* yang lebih besar, yaitu *module* di level paling atas yang digunakan oleh Onti Measures.

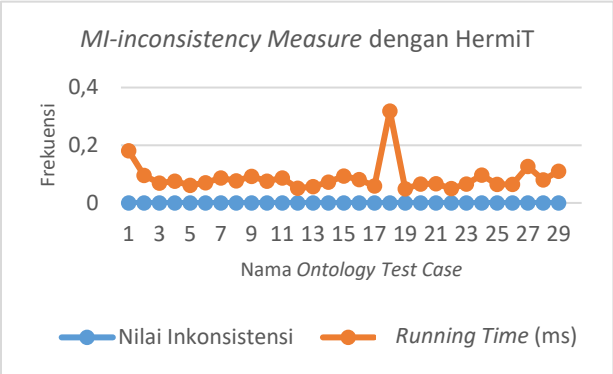
Kebenaran suatu *module* pada kode program dicek dari ketiadaan *error* saat dipakai untuk menjalankan sebuah masukan program serta dicek dari kebenaran keluarannya. Khusus dalam hal ini, empat berkas *OWL ontology* berisi informasi sederhana sengaja dibuat. Salah satu dari empat berkas tersebut merepresentasikan *ontology* yang konsisten dan sisanya inkonsisten. Seluruh berkas tersebut dijalankan tanpa menggunakan antarmuka web Onti Measures, tetapi menggunakan IDE Eclipse. Hasilnya adalah ketika berkas dijalankan, *console* IDE Eclipse tidak menunjukkan adanya *error*. Selanjutnya, untuk melihat kebenaran keluarannya, *log* perhitungan dan nilai inkonsistensi dicek kebenarannya. *Log* perhitungan didapat dari hasil proses perhitungan yang dicetak oleh program. Berdasarkan pengecekan data keluaran yang dihasilkan Onti Measures, *log* perhitungan dan nilai inkonsistensi dinyatakan benar.

Blackbox testing dilakukan untuk memastikan bahwa tampilan aplikasi beserta fungsi-fungsinya dapat berjalan sesuai dengan rancangan. Referensi [15] menyatakan bahwa *blackblox testing* disebut juga sebagai *functional testing*. Pengujian fungsi utama Onti Measures dilakukan dengan memanfaatkan sampel penelitian berupa berkas *OWL ontology* berjumlah tiga puluh buah seperti yang disebutkan satu per satu pada Tabel III. Sampel tersebut adalah *ontology test case* yang dijadikan sebagai bahan masukan program Onti Measures.

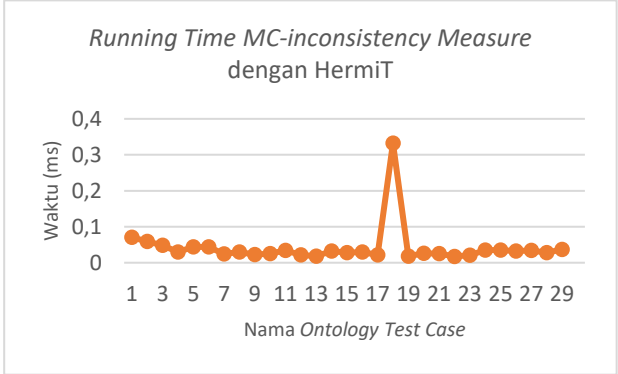
Masing-masing *ontology* dijalankan dengan sebuah *OWL reasoner* dalam satu waktu. Hal tersebut berarti banyaknya kemungkinan untuk menjalankan tiga puluh berkas adalah 30 berkas × 3 *OWL reasoner* = 90 kali *running*. Masing-masing *running* menghasilkan data sepuluh nilai inkonsistensi dan sepuluh *running time* dari sepuluh jenis *measures*. Maka, dengan menjalankan 90 kali *running* pada Onti Measures akan dihasilkan data keluaran berupa 900 nilai inkonsistensi, 900 *running time*, dan tiga puluh ukuran *ontology*. Jumlah ukuran



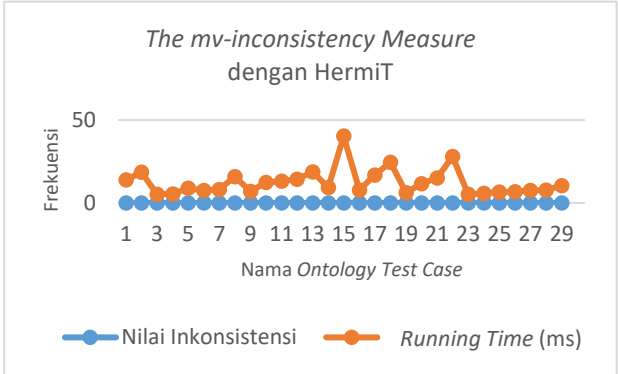
Gbr. 5 Nilai inkonsistensi *ontology test case* dan *running time* dari *drastic inconsistency measure* menggunakan *reasoner* HermiT.



Gbr. 6 Nilai inkonsistensi *ontology test case* dan *running time* dari *MI-inconsistency measure* menggunakan *reasoner* HermiT.



Gbr. 7 *Running time* untuk menjalankan *ontology test case* diukur dengan *MC-inconsistency measure* dan *reasoner* HermiT.



Gbr. 8 Nilai inkonsistensi *ontology test case* dan *running time* dari *the mv-inconsistency measure* menggunakan *reasoner* HermiT.

ontology tersebut tergantung pada jumlah berkas *OWL ontology* yang diuji. Hasil pengujian Onti Measures selengkapnya dijabarkan pada penjelasan berikut. Nama *ontology test case* 1-30 merujuk pada daftar urutan *ontology* di Tabel III.

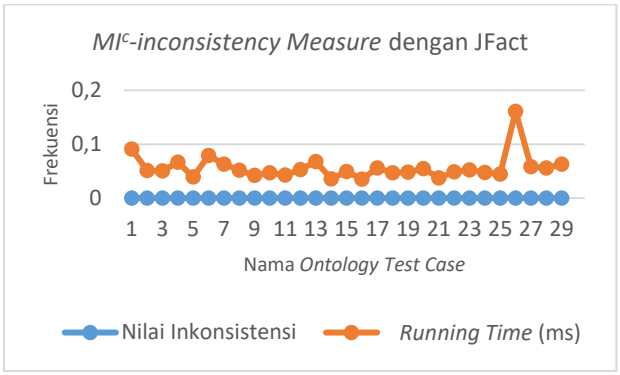
1) Nilai Inkonsistensi: Berdasarkan hasil pengujian tiga puluh *ontology* menggunakan *OWL reasoner* HermiT, diperoleh hasil bahwa 29 *ontology* adalah konsisten atau dengan kata lain mempunyai nilai inkonsistensi 0, sedangkan satu *ontology* lainnya, yaitu nomor 30, dinyatakan *not compatible*, sehingga tidak ada nilai inkonsistensi yang bisa dihasilkan, seperti tampak pada Gbr. 5 dan Gbr. 6. Status *not compatible* tersebut terjadi karena adanya sebuah tipe data atau elemen pada *OWL ontology* yang tidak dapat didukung oleh *reasoner* tersebut. Sementara itu, pada penggunaan *OWL reasoner* Pellet, terdapat satu *ontology* yang *not compatible* pula. Hanya saja, hal tersebut terjadi pada *ontology* yang berbeda, yaitu pada nomor 29. *Ontology* lain dapat menghasilkan nilai inkonsistensi, yaitu nilai 0.

Gbr. 7 menunjukkan nilai inkonsistensi tidak dapat dihasilkan. Hal tersebut terjadi karena *MC*, D_f, *the nc*, dan ID_{MCS} *inconsistency measure* pada Onti Measures hanya mendukung pemakaian *OWL ontology* dengan ukuran *ontology* maksimal 20. Hal ini disebabkan pada perhitungan empat *measure* tersebut terdapat pemakaian metode *power set* pada kode programnya. Jika *power set* dipakai untuk ukuran *ontology* yang bernilai lebih dari 20, akan dibutuhkan waktu yang sangat lama (relatif tidak dapat diprediksi secara pasti) untuk menyelesaikannya. Oleh karena itu, Onti Measures terpaksa menggagalkan proses tersebut, yang berdampak pada gagalnya pemakaian empat *measure* tersebut dan nilai inkonsistensinya pun tidak dapat ditemukan.

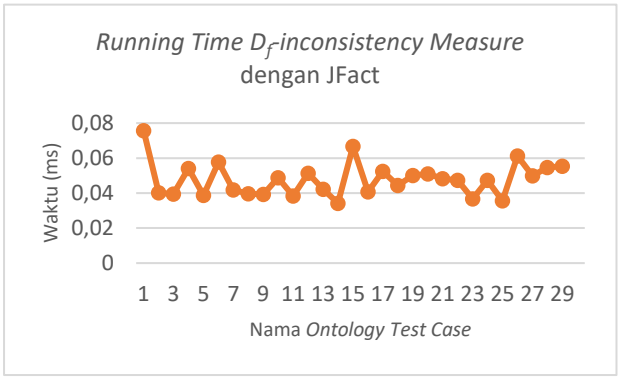
Gbr. 8 dan Gbr. 9 juga menunjukkan keadaan nomor 30 yang sama dengan yang tampak pada Gbr. 5 dan Gbr. 6, hanya saja Gbr. 9 menggunakan OWL *reasoner* yang berbeda, yaitu JFact. Sedangkan Gbr. 10 dan Gbr. 11 menunjukkan kondisi yang sama dengan yang tampak pada Gbr. 7.

Satu-satunya *ontology* yang dapat menghasilkan nilai inkonsistensi lebih dari 0 adalah *ontology* nomor 30 (*HIVont00157*) dengan pemakaian *reasoner* Pellet. Ini berarti *ontology* tentang penyakit HIV tersebut inkonsisten. Gbr. 12 menunjukkan keadaan tersebut dengan nilainya adalah 3. Gbr. 13 juga menunjukkan keadaan yang sama dengan nilainya yang mendekati 0, yaitu 0,0011. Keadaan inkonsisten ini terjadi bukan hanya pada *problematic* dan *incompatibility ratio inconsistency measure* saja, melainkan pada seluruh jenis *measure* yang ada pada Onti Measures dengan pemakaian *reasoner* Pellet. Maka, dapat dikatakan bahwa dari tiga puluh *OWL ontology* yang digunakan pada pengujian ini, hanya satu *ontology* yang inkonsisten, yaitu *ontology HIVont00157*.

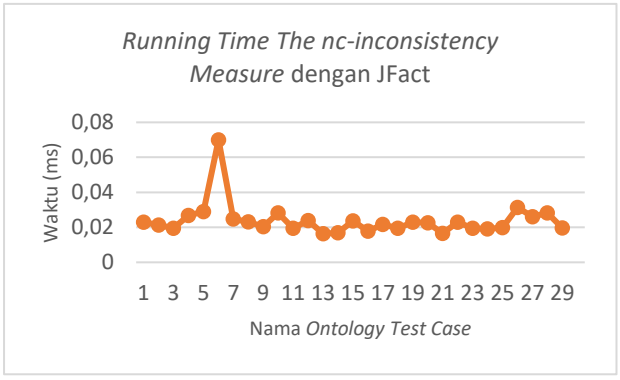
2) Running Time: Gbr. 5 menunjukkan bahwa secara keseluruhan, *running time* untuk menjalankan masing-masing *ontology* tidak ada yang melebihi 1 milidetik. Hal yang sama



Gbr. 9 Nilai inkonsistensi *ontology test case* dan *running time* dari *MI*^{C}-*inconsistency measure* menggunakan *reasoner* JFact.



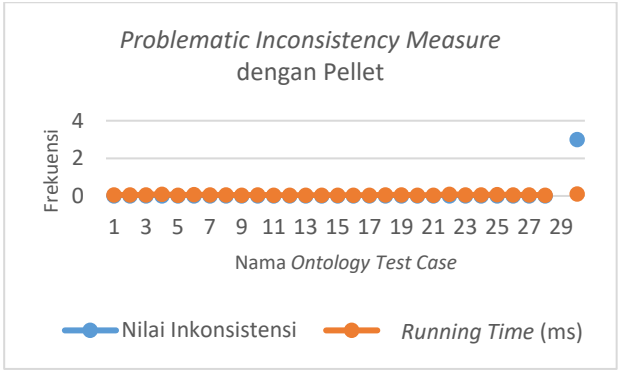
Gbr. 10 *Running time* untuk menjalankan *ontology test case* diukur dengan D_f *inconsistency measure* dan *reasoner* JFact.



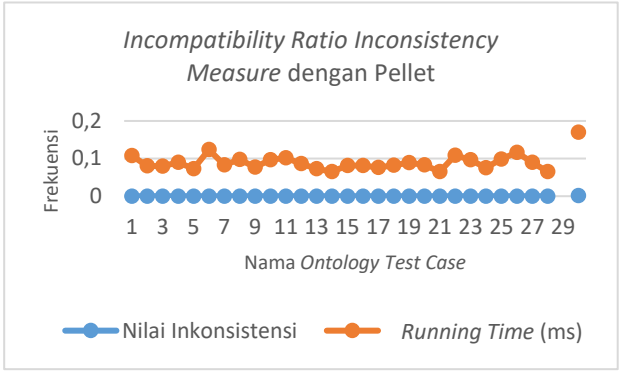
Gbr. 11 *Running time* untuk menjalankan *ontology test case* diukur dengan *the nc-inconsistency measure* dan *reasoner* JFact.

juga terjadi pada *measure* yang lain, seperti yang tampak pada Gbr. 6, Gbr. 7, dan Gbr. 9 sampai Gbr. 13. Di sisi lain, Gbr. 8 dan Gbr. 14 menunjukkan bahwa terdapat beberapa *ontology* yang membutuhkan *running time* hingga puluhan milidetik untuk dijalankan dengan *the mv* dan ID_{MCS} *inconsistency measure* beserta pilihan *reasoner* yang tersedia. Dapat dikatakan bahwa *the mv* dan ID_{MCS} *inconsistency measure* pada Onti Measures adalah dua jenis *measure* yang membutuhkan waktu paling lama untuk menjalankan sebuah *ontology*.

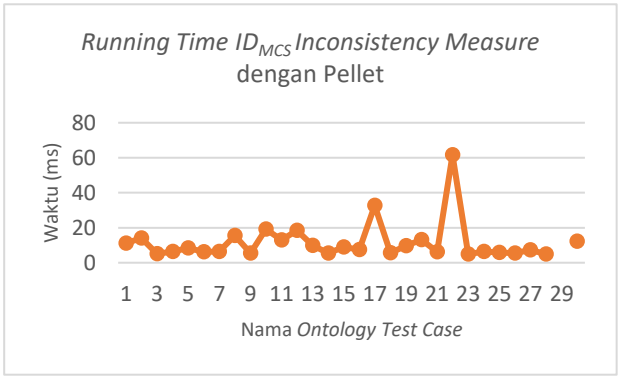
3) Ukuran Ontology: Masing-masing *ontology* pada makalah ini memiliki ukuran yang berbeda. Ukuran *ontology* merepresentasikan banyaknya jumlah *axiom* yang ada pada



Gbr. 12 Nilai inkonsistensi ontology *test case* dan *running time* dari *problematic inconsistency measure* menggunakan *reasoner* Pellet.



Gbr. 13 Nilai inkonsistensi *ontology test case* dan *running time* dari *incompatibility ratio inconsistency measure* menggunakan reasoner Pellet.



Gbr. 14 Nilai inkonsistensi *ontology test case* dan *running time* dari ID_{MCS} *inconsistency measure* menggunakan *reasoner* Pellet.

ontology tersebut. Berdasarkan hasil pengujian, ditemukan informasi bahwa ukuran *ontology* terbesar adalah milik *ontology* nomor 22, yaitu Partumdo (*postpartum depression ontology*) sebesar 15.002. Hal itu berarti terdapat 15.002 *axiom* yang berada di dalam *ontology* Partumdo. Sementara itu, *ontology* nomor 24 (*COPD ontology*) memiliki ukuran *ontology* terkecil, yaitu 108. Informasi-informasi ini tercantum pada Tabel IV. Nomor *ontology* pada tabel tersebut mengacu pada nomor di Tabel III.

B. Perawatan

Penelitian ini telah merapikan dan memperbaiki struktur kode program prototipe sebelumnya sehingga *best practice*

TABEL IV
UKURAN *ONTOLOGY* PADA SAMPEL PENELITIAN

Nomor *Ontology*	**Ukuran *Ontology***	**Nomor *Ontology***	**Ukuran *Ontology***
1	2.637	16	399
2	2.401	17	1.552
3	283	18	362
4	283	19	425
5	579	20	2726
6	836	21	272
7	701	22	15.002
8	4.086	23	135
9	689	24	108
10	1.661	25	313
11	1.252	26	540
12	6.206	27	409
13	801	28	307
14	637	29	871
15	1.163	30	2.698

dari implementasi kode program untuk Onti Measures telah tercapai. Selain itu, penggunaan New Relic pada sistem dapat dijadikan salah satu alat untuk melihat kinerja sistem dan mempermudah pemantauannya kapan pun jika diperlukan. Hal ini sangat berguna untuk perawatan sistem, terutama jika ada pengembangan selanjutnya dengan pemakaian Heroku.

VI. KESIMPULAN

Makalah ini telah menghasilkan sebuah program aplikasi berbasis web bernama Onti Measures yang telah berhasil dibangun. Aplikasi ini merupakan sebuah media berupa kumpulan beberapa *inconsistency measure* untuk *ontology* yang dapat digunakan secara bebas (gratis) dan terbuka untuk umum guna mengukur tingkat inkonsistensi suatu *ontology*. Secara khusus, makalah ini menyajikan data hasil pengujian inkonsistensi *ontology* untuk beberapa kasus virus dan penyakit.

Berdasarkan pengujian yang telah dilakukan, Onti Measures telah lulus pengujian dari segi kode program maupun tampilan antarmuka. Ini berarti fungsi-fungsi pada Onti Measures dapat berjalan dengan baik. Masukan Onti Measures yang berupa tiga puluh *OWL ontology* telah dijalankan dengan Onti Measures beserta tiga pilihan *OWL reasoner*, yaitu HermiT, JFact, dan Pellet. Hasilnya adalah hanya satu dari tiga puluh *ontology* tersebut yang dinyatakan inkonsisten, yaitu *ontology HIVont00157*. *Ontology* tersebut berisi informasi tentang penyakit HIV. Sebanyak 29 *ontology* lainnya dinyatakan konsisten.

Berdasarkan informasi hasil pengujian dari sisi *running time*, *the mv* dan ID_{MCS} *inconsistency measure* adalah dua jenis *measure* pada Onti Measures yang membutuhkan waktu paling lama untuk menjalankan sebuah *ontology*. Sementara itu, dari sisi ukuran *ontology*, Partumdo (untuk penyakit depresi setelah melahirkan) memiliki ukuran *ontology* terbesar dan *COPD* (untuk penyakit paru-paru) memiliki ukuran *ontology* terkecil dari seluruh *ontology* yang ada. Besar atau kecilnya ukuran sebuah *ontology* tidak menentukan besarnya nilai inkonsistensi *ontology* tersebut.

Pengembangan selanjutnya yang dapat dilakukan dari penelitian ini adalah analisis lebih detail dari hasil pengujian Onti Measures dengan perhitungan statistika yang lebih mendalam. Selain itu, uji manfaat dari Onti Measures juga dapat dilakukan dengan melibatkan beberapa pengguna sebagai responden.

UCAPAN TERIMA KASIH

Ucapan terima kasih disampaikan kepada Kemenristekdikti/BRIN (Kementerian Riset, Teknologi, dan Perguruan Tinggi/Badan Riset dan Inovasi Nasional) yang telah mendukung jalannya penelitian ini melalui program pendanaan penelitian kategori penelitian kompetitif nasional. Penelitian ini termasuk dalam skema PDP (Penelitian Dosen Pemula) tahun anggaran 2021 dengan ketua peneliti Nur Alfi Ekowati.

REFERENSI

[1] N.A. Ekowati, Sunaryono, dan D. Prasetyo, "Inconsistency Measures OWL Ontology Berbasis Axiom dengan MinInc Inconsistency Value," *Teknikom*, Vol. 3, No. 1, hal. 1-4, 2019.

[2] (2021) Covid-19 WHO (World Health Organizaion) website, [Online], https://covid19.who.int/, tanggal akses: 29-Agu-2021.

[3] K. McAreavey, W. Liu, dan P. Miller, "Computational Approaches to Finding and Measuring Inconsistency in Arbitrary Knowledge Bases," *Int. J. Approx. Reasoning*, Vol. 55, No. 8, hal. 1659-1693, Nov. 2014.

[4] M. Thimm, "On the Expressivity of Inconsistency Measures," *IJCAI-17: Proc. the 26th Int. Joint Conf. Artif. Intell*., 2017, hal. 5070-5074.

[5] S. Akama dan N.C.A. da Costa, "Why Paraconsistent Logics?" dalam *Towards Paraconsistent Engineering*, *Intelligent Systems Reference Library*, Vol 110, S. Akama, Ed., Cham, Swiss: Springer, 2016, hal. 7-24.

[6] M. Thimm, "On the Evaluation of Inconsistency Measures," dalam *Measuring Inconsistency in Information*, *Studies in Logic*, Vol. 73, J. Grant dan M.V. Martinez, Eds., Rickmansworth, Inggris: College Publication, 2018, Ch. 2.

[7] N.A. Ekowati dan Sunaryono, "Axiom-based Inconsistency Measures for OWL Ontology," *Int. J. Adv. Sci. Technol. (IJAST)*, Vol. 29, No. 7, hal. 10812-10822, Jul. 2020.

[8] N.A. Ekowati, "Inconsistency Measures for OWL Ontologies," M.Sc. thesis, Technische Universitat Dresden, Dresden, Jerman, Mar. 2017.

[9] L.D. Bentley dan J.L. Whitten, *Systems Analysis and Design for the Global Enterprise*, 7th ed., New York, AS: McGrawHill, 2007.

[10] (2021) Onti Measures website, [Online], https://ontimeasures.herokuapp.com/, tanggal akses: 3-Sep-2021.

[11] L. Cohen, L. Manion, dan K. Morrison, *Research Methods in Education*, 6th ed., London, Inggris: Routledge, 2007.

[12] (2021) NCBO BioPortal website, [Online], https://bioportal.bioontology.org/, tanggal akses: 5-Agu-2021.

[13] (2021) AberOWL Repository website, [Online], http://aber-owl.net/, tanggal akses: 3-Agu-2021.

[14] (2021) Bio Ontology website, [Online], https://data.bioontology.org/, tanggal akses: 4-Agu-2021.

[15] S. Nidhra dan J. Dondeti, "Blackbox and Whitebox Testing Techniques – A Literature Review," *Int. J. Embedded Syst. Appl. (IJESA)*, Vol. 2, No. 2, hal. 29-50, Jun. 2012.