PEMBANGKITAN DATA UJI BERBASIS DIAGRAM AKTIVITAS DAN DIAGRAM *STATECHART*

Arif Rahman Sujatmika[1)], Yanuangga Gala Hartlambang[2)]
Universitas Darul Ulum
Jl. Abdurrahman Wahid No.29 Jombang
Email: arif.sujatmika@undar.ac.id, yanuangga@undar.ac.id

Abstrak

Testing is the stage of software development used to determine whether a software is ready for release or not. In making test cases using reference activity diagrams and statechart diagrams, a help representation was made, ie State-Activity-Diagram (SAD). The generation of test cases using a reference between the statechart diagram and the status diagram is still inadequate because in the case of the test produced there is no test data. The selection of test data for many test cases will be tedious and time consuming. In this paper, it is proposed to generate test data automatically based on existing test cases. Test data created based on class diagrams, and data dictionaries. The test case data consists of inputs and results. First enter information about the functions involved in the test case into the SAD node so that the SAD-S Diagram is obtained. Second, after the process of making the test case is completed, the test data is made by looking at the data dictionary function so that the test data is formed.

Keywords: *UML-based testing, data generation, Activity Diagrams and Statechart Diagrams, OCL.*

1. Pendahuluan

Pengujian merupakan tahapan dari pengembangan perangkat lunak yang digunakan untuk menentukan apakah suatu perangkat lunak telah siap untuk dirilis atau tidak. UML adalah himpunan struktur dan teknik untuk pemodelan rancangan perangkat lunak berorientasi objek. UML merupakan salah satu dari model yang banyak dipakai oleh para pengembang. Hal ini dikarenakan UML dapat dengan mudah dimengerti oleh pengembang. UML yang dibuat berdasar perspektif dari pengguna, sehingga mempermudah proses verifikasinya [1].

Diagram Aktivitas menggambarkan berbagai alir aktivitas dalam perangkat lunak yang sedang dirancang, bagaimana masing – masing alir berawal, keputusan yang mungkin terjadi, dan bagaimana alurnya berakhir. Diagram Aktivitas merupakan Diagram Statechart khusus, dimana sebagian besar transisi dipicu oleh selesainya status sebelumnya (internal processing) [2].

Dalam pembangkitan kasus uji menggunakan diagram Statechart, langkah awal yang dilakukan adalah secara acak membangkitkan kasus uji berdasar Diagram Aktivitas perangkat lunak yang akan diuji. Ketika kasus uji tersebut dijalankan pada perangkat lunak, maka akan didapat rekam jejak jalan sistem. Terakhir akan dilakukan proses reduksi kasus uji dengan membandingkan rekam jejak alur dengan alur sederhana (simple path) dari Diagram Aktivitas. Pendekatan ini juga dapat digunakan untuk memeriksa konsistensi antara rekam jejak jalan sistem dengan perilaku pada Diagram Aktivitas [3].

Diagram Statechart menggambarkan transisi dan perubahan keadaan (dari satu status ke status lainnya) suatu objek pada perangkat lunak sebagai akibat dari stimuli yang diterima. Pada umumnya Diagram Statechart menggambarkan kelas tertentu [2]. Dalam pembangkitan kasus uji menggunakan Diagram Statechart langkah yang dilakukan adalah mentransformasikan Diagram Statechart kedalam bentuk diagram lain, yaitu TFG (Testing Flow Graph). Diagram ini berfungsi untuk mereduksi tingkat kompleksitas dari Diagram Statechart. Pembangkitan kasus uji dilakukan berdasar TFG dari sistem yang ada. Langkah terakhir yang dilakukan adalah melakukan evaluasi dengan metode analisis mutasi untuk mengetahui tingkat efektifitas dari kasus uji [3].

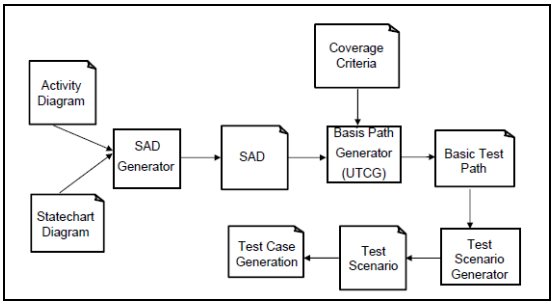
Dalam pembuatan kasus uji dengan menggunakan referensi Diagram Aktivitas dan Diagram Statechart, sebuah representasi bantuan dibuat, yaitu State-Activity-Diagram (SAD). Dalam SAD, kendali arus informasi selama eksekusi kasus penggunaan direpresentasikan menggunakan kombinasi dari Diagram Statechart dan Diagram Aktivitas. Kendali status dapat diperoleh dari ekstraksi Diagram Statechart, sedangkan kendali alur diekstraksi dari Diagram Aktivitas. Untuk menangani eksekusi yang bersamaan, beberapa simpul baru telah diperkenalkan [4]. Pembangkitan kasus uji dengan menggunakan referensi antara Diagram Statechart dan diagram Aktivitas masih terdapat kekurangan karena dalam kasus uji yang dihasilkan belum terdapat data uji. Pemilihan data uji untuk kasus uji yang banyak akan membosankan dan memakan banyak waktu. Oleh karena itu perlu adanya pembuatan data uji secara otomatis untuk kasus uji tersebut.

2. Metode Penelitian

SATEC (State Activity Test Case Generation) merupakan metode yang pakai untuk membangkitkan kasus
penggunaan berdasar Diagram Statechart dan Diagram Aktivitas seperti diagambarkan pada Gambar 1. Sebagian besar SATEC terdiri dari dua

fase penting, diantaranya sintesis SAD dan pembuatan kasus uji.
Komponen dalam SAD terdiri dari:

- simpul State-Activity;
- simpul AND-OR;
- busur.



Gambar 1. Gambaran Umum Pembangkitan Kasus uji Menggunakan Diagram Statechart dan Activity Diagram [4]

Setiap simpul terdapat sebuah aktivitas. Sebuah aktivitas dijalankan ketika memasuki sebuah simpul ketika fase eksekusi. Ketika status tidak sedang melakukan suatu aktivitas, hal tersebut diasumsikan dalam keadaan menunggu kejadian dari luar untuk memicu transisi statusnya.

Untuk membuat Diagram SAD digunakan dua patokan diagram, diantaranya Diagram Aktivitas dan Diagram Statechart. Simpul dari diagram ini dibuat berdasar simpul pada Diagram Statechart, sedangkan alur dari diagram dibuat berdasar alur dari Diagram Aktivitas

Untuk menghasilkan alur pengujian dari SAD digunakan depth-first search. Dimulai dari status awal dan mengeksplorasi status tetangga dengan menjelajahi semua busur yang tersambung. Selanjutnya menyimpan status sistem, busur transisi dan kondisi penjaga pada tumpukan. Ketika terdapat perulangan, seperti yang disebutkan dalam definisi alur basis, perulangan akan dilalui sekali. Oleh karena itu, saat berhadapan dengan perulangan, akan dilakukan pengecekan apakah status yang baru dilewati merupakan anggota dari alur basis yang ada dalam tumpukan. Jika status tersebut merupakan anggota sebuah alur, maka akan dibuat alur basis dengan mengabungkan antara alur bagian dari alur basis dengan alur yang mencakup status tersebut. Jika tidak terdapat alur basis yang telah ada, akan dilakukan penelusuran pada tumpukan dan memperbaharui tumpukan sehingga selanjutnya terdapat status saat mengeksplorasi busur yang belum tereksplorasi pada kedalaman maksimum [4].

Kamus data adalah suatu daftar data elemen yang terorganisir dengan definisi yang tetap dan sesuai dengan sistem, sehingga user dan analis sistem mempunyai pengertian yang sama tentang masukan, hasil dan komponen data. Pembentukan kamus data dilaksanakan dalam tahap analisis dan perancangan suatu sistem [5].

Ekspresi OCL dapat didefinisikan menjadi sebuah precondition atau postcondition yang berhubungan dengan operasi atau perilaku pada sistem. Deklarasi konteks dalam OCL menggunakan kata kunci context, yang diikuti oleh deklarasi tipe dan operasi. Batasan dari suatu stereotype ditunjukkan dengan mengambil label ‘pre’ dan ‘post’ sebagaimana precondicition dan postcondition[6]. Contoh:

contextTypename::operationName(param1 : Type1, ...): ReturnType
pre : param1 > ...
post: result = ..

Tabel 1 merupakan perbandingan model referensi pembangkitan kasus uji dari Diagram Aktivitas, Diagram Statechart, dan Diagram Urutan. Dari tabel tersebut dapat disimpulkan bahwa diagram gabungan antara Diagram Statechart dan Diagram Aktivitas lebih baik dibanding ketiga pembandingnya. Hal ini dikarenakan antara Diagram Statechart dan Diagram Aktivitas saling melengkapi sehingga kelemahan pada tiap diagram dapat ditutupi. Akan tetapi masih terdapat kekurangan, yaitu tidak terdapat data uji. Oleh karena itu pembangkitan data uji dibutuhkan untuk melengkapi kerangka kerjanya, dan metode itu dapat diperoleh dengan memberi implan sebagian kerangka kerja pembangkitan kasus uji dengan diagram urutan.

Tabel 1 Perbandingan Model Referensi Pembangkitan Kasus uji

Pembandingan	**Kasus uji dengan Diagram Aktivitas[3]**	**Kasus uji dengan Diagram Statechart [7]**	**Kasus uji dengan Diagram Urutan [8]**	**Kasus uji dengan Activity dan Statechart [4]**	**Usulan**
Diagram	Diagram Aktivitas	Diagram Statechart	Diagram urutan	Diagram Activity dan Diagram Statechart	Diagram Activity dan Diagram Statechart
Cakup-an	Cakupan alur	Cakupan transisi	Cakupan alur	Cakupan basis alur (gabungan antara alur dan transisi)	Cakupan basis alur (gabungan antara alur dan transisi)
Model bantu	Simple path	TFG (Testing Flow Diagram)	SDG (Sequence Diagram Graph)	SAD	SAD + SDG
Kelebihan	Merepresenta-sikan dengan baik perilak	Merepresentasikan status informasi dalam sistem	Terdapat data uji	Merepresentasikan status informasi dan	Merepresentasikan status

	u sistem secara garis besar			perilaku sistem	informasi, perilaku sistem, serta data uji
Kelemahan	Tidak merepresenta-sikan detail informasi pada objek sistem	Tidak dapat merepresentasikan perilaku sistem	Tidak dapat mengetahui perilaku keseluruhan sistem	Tidak adanya data uji	-

3. Hasil dan Pembahasan

Untuk mencapai tujuan yang diharapkan dari penelitian ini, maka terdapat beberapa langkah yang dilaksanakan, diantaranya:

a. Analisis sistem
b. Perancangan sistem
c. Impelementasi sistem
d. Pengujian sistem

3.1. Analisis Sistem

Dalam tahap analisis, terdapat tiga bagian diantaranya:

a. analisis model dalam UML,
b. analisis pembuatan kasus uji,
c. analisis pembuatan data uji.

Dalam analisis model UML, hal yang perlu diperhatikan adalah diagram apa saja yang digunakan dalam membuat kasus uji. Dalam penelitian ini diagram yang digunakan adalah Diagram Aktivitas, Diagram Statechart, dan diagram kelas. Sedangkan informasi pendukung yang dipakai adalah OCL pada simpul diagram kelas. Diagram Aktivitas digunakan untuk melihat alur dari sebuah sub sistem secara garisbesar. Diagram Aktivitas terdiri dari swimlane, simpul, dan busur. Swimlane berfungsi untuk mendefinisikan kelompok dari pelaku yang terlibat dalam aktivitas tersebut.

Diagram Statechart digunakan untuk melihat secara lebih mendetail alur status pada tiap transisi dalam satu objek dalam suatu aktivitas. Diagram Statechart terdiri dari simpul dan busur. Perpindahan simpul dalam diagram dipicu oleh aktivitas dalam busurnya. Oleh karena itu, Diagram Aktivitas dan Diagram Statechart dapat dikatakan diagram yang saling melengkapi.

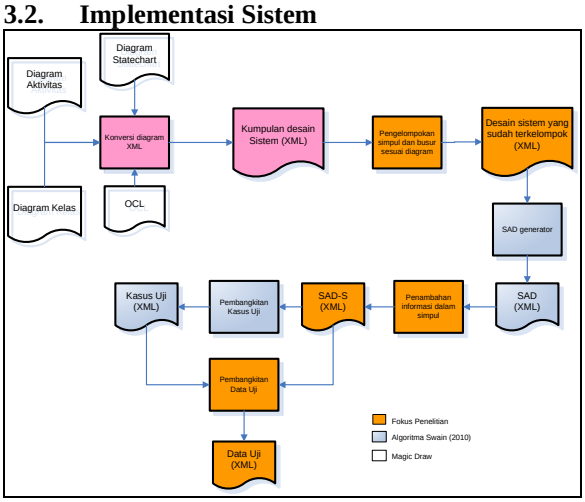
Diagram kelas memberikan pandangan secara luas dari suatu sistem dengan menunjukkan kelas – kelasnya dan hubungan mereka. Diagram kelas merupakan diagram yang menggambarkan keseluruhan fungsi dan parameter fungsi pada sistem tersebut. OCL digunakan untuk mendefinisikan perilaku fungsi, seperti kondisi yang harus dicapai sebelum masuk fungsi, serta kondisi setelah melewati fungsi.

Setelah pembuatan diagram UML selesai, akan dilakukan proses validasi diagram UML yang ada. Proses ini digunakan untuk memastikan bahwa diagram yang dibuat telah sesuai dengan standart yang ada. Untuk proses validasi model yang dimasukkan kedalam kerangka kerja, digunakan validasi secara manual. Jadi sebelum model dimasukkan pada kerangka kerja model sudah dalam keadaan tervalidasi.

Analisis pembuatan kasus uji, sebagaimana telah dijelaskan oleh Swain[4] yang telah dibahas pada bab sebelumnya, menggunakan metode depth-first search. Disamping itu metode yang digunakan menggunakan basis alur, dimana hanya akan mengeksekusi perulangan pada iterasi satu kali saja.

Tahap pembangkitan data uji dilakukan setelah terbentuk alur dalam SAD-S yang akan menjadi kasus uji. Algoritma untuk proses pembangkitan data uji secara otomatis mengadopsi dari algoritma TestSetGeneration [8]. Data uji dibuat berdasar alur kasus uji yang terbentuk sebagai acuan untuk menentukan input, serta output pada status yang akan dikunjungi.

3.2. Implementasi Sistem



Gambar 2. Gambaran umum pembuatan data uji

Sistem yang akan dibangun dalam penelitian ini adalah sistem yang mampu membangkitkan kasus uji berdasar Diagram Statechart dan Diagram Aktivitas. Untuk proses pembangkitan kasus uji akan dilakukan berdasar dari referensi Diagram Statechart dan Diagram Aktivitas. Sedangkan untuk membangkitkan data kasus uji menggunakan referensi dari diagram kelas dan OCL. Secara umum alur sistem dapat dilihat pada Gambar 2.

Pembangkitan kasus dan data uji terbagi menjadi beberapa tahapan, diantaranya pembangkitan alur gabungan antara Diagram Statechart dan Diagram Aktivitas yang dinamakan SAD-S. Status pada graph ini diekstraksi dari Diagram Statechart, sedangkan alur kendali akan diambil dari alur Diagram Aktivitas. Dalam status berisi beberapa informasi mengenai atribut, kondisi sebelum masuk status, kondisi setelah keluar dari status dan jangkauan nilai atribut fungsi status pada sistem. Status dalam alur juga menyimpan hasil

yang akan diperoleh pada fungsi yang terdapat dalam status tersebut yang diperoleh dengan mengkolaborasikan antara masukan dari diagram kelas, standar kasus uji, dan kamus data OCL. Selain itu juga terdapat beberapa simpul bantuan agar alur sistem dapat dibuat, diantaranya simpul AND dan simpul OR.

Kriteria cakupan dalam pembangkitan kasus uji yang digunakan adalah cakupan alur basis. Cakupan alur basis digunakan karena basis path dinilai lebih kuat dari cakupan transisi dan cakupan status [4]. Secara garis besar kriteria cakupan ini adalah memastikan bahwa keseluruhan status dan busur telah dikunjungi minimal sekali. Pembangkitan kasus uji dilakukan dengan referensi alur basis dan kondisi dari penjaga pada transisinya. Algoritma yang digunakan untuk membangkitkan alur basis adalah depth first-search. Langkah awal berasal dari status awal, dilanjutkan dengan mencari status yang memiliki transisi dengan status awal. Keseluruhan status, busur, dan penjaga yang ada didekat status akan disimpan pada tumpukan. Ketika terdapat perulangan, akan dilakukan pengecekan dalam tumpukan apakah status tersebut sudah terdapat dalam alur basis yang sudah ada. Apabila status tersebut tidak ditemukan, maka status tersebut akan dimasukkan kedalam tumpukan dan akan dipakai ketika menemui status yang cocok yang merupakan alur basisnya. Apabila status tersebut termasuk bagian dari alur basis yang sudah ada, maka alur basis tersebut akan ditambahkan status yang telah dikunjungi.

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>

<root>

<simpul>

<ID>19</ID>

<Activity>GetPIN()</Activity>

<State>PIN Entry</State>

<in>39</in>

<out>26</out>

<Fungsi>GetPIN</Fungsi>

<Parameter>String f</Parameter>
```

Gambar 3. Pembuatan data uji pada simpul SAD-S

Tahap pembangkitan data uji dilakukan dengan cara berdasar alur kasus uji yang terbentuk sebagai acuan untuk menentukan input, serta output pada status yang akan dikunjungi sebagaimana telah dijelaskan pada subbab sebelumnya. Dalam hal ini dibutuhkan dua file, diantaranya file output proses penambahan informasi status dan file alur pengujian. File hasil penambahan proses dipakai sebagai bahan membuat data uji yang memanfaatkan OCL nya, sedangkan file alur pengujian sebagai acuan alur dari data ujinya seperti digambarkan dalam gambar 3.

Langkah pengujian pada penelitian ini dibagi menjadi beberapa tahapan diantaranya

- pembuatan artefak sistem,
- menjalankan kerangka kerja,
- menganalisa kasus uji dan data uji hasil pengujian.

Pembuatan artefak sistem dilakukan dengan cara melakukan konsultasi pada perusahaan penyedia perangkat lunak. Artefak sistem yang dibutuhkan antara lain Diagram Aktivitas, Diagram Statechart, Diagram kelas, serta kamus data berformat OCL. OCL digunakan untuk merepresentasikan perilaku fungsi terkait. Fungsi OCL yang dipakai adalah fungsi “pre condition” atau biasa disebut kondisi sebelum eksekusi fungsi dan fungsi “post condition” atau biasa disebut kondisi setelah eksekusi fungsi.

Kerangka kerja dijalankan pada lingkungan computer dengan spesifikasi processor AMD dual core 1,6 Ghz serta memori 4 Gb. Artefak sistem dibuat dengan menggunakan alat bantu Magicdraw yang dapat diakses pada website http://www.magicdraw.com. Artefak sistem tersebut akan dirubah kedalam file dengan format XML agar dapat dibaca oleh alat bantu yang dibuat peneliti. Hasil dari alat bantu yang dibuat peneliti berupa tiga file diantaranya file berisi diagram yang diperlukan untuk membangkitkan kasus uji, file berisi daftar simpul dan busur diagram SAD-S, serta file berisi alur pengujian dan data uji yang dihasilkan pada masing-masing kasus uji.

Setelah dilakukan proses pengujian dari beberapa skenario diatas dilakukan pengecekan kesesuaian/kebenaran kasus uji serta data uji. Kasus uji dikatakan sesuai/benar apabila alur dari kasus uji telah sesuai dengan alur pada diagram aktivitas yang ada, sedangkan data uji dikatakan sesuai/benar apabila data uji yang dihasilkan telah sesuai dengan deskripsi perilaku sistem. Semua proses penentuan kesesuaian kasus uji dan data uji dilakukan oleh peneliti secara manual dengan melihat hasil pengujian dan dokumen artefak sistem dari masing-masing scenario

Pada sub proses kedatangan pasien, pasien akan memilih akan masuk pada poli mana mereka menuju. Pada poliklinik terdapat dua buah poli, diantaranya poli umum dan poli gigi. Setelah memilih poli pasien akan menerima informasi menuju ruangan mana pasien harus ada. Berikut Diagram Aktivitas, Diagram Statechart, Diagram Kelas, serta OCL pada sub proses tersebut.

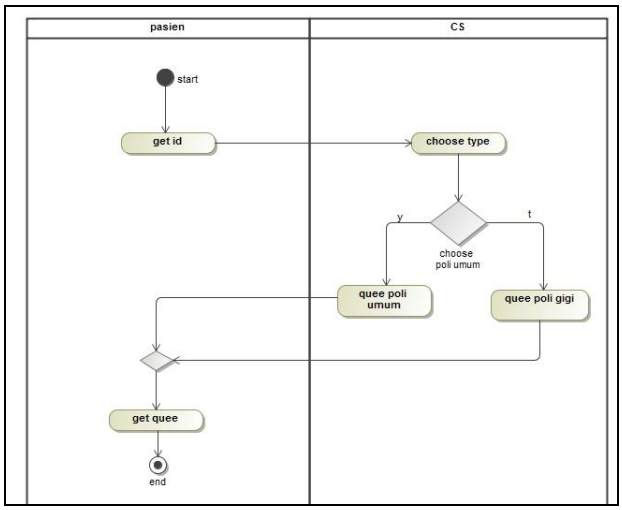
Dari beberapa diagram diatas dilakukan proses seperti berikut:

- proses filtrasi diagram file xml hasil dari magicdraw,
- proses pembuatan diagram gabungan Diagram Statechart dan Diagram
- Aktivitas,
- proses penambahan informasi dalam simpul SAD-S,

- proses pembuatan kasus uji,
- proses pembuatan data uji.

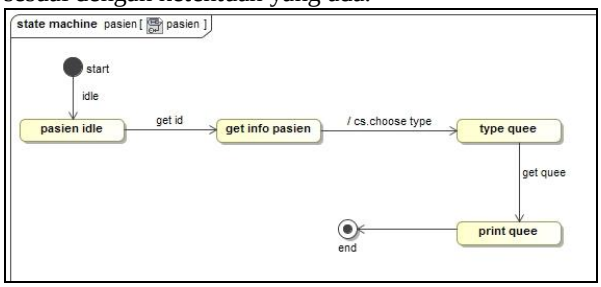
File desain sistem berupa diagram UML yang terdiri dari beberapa diagram diatas, akan dirubah terlebih dahulu dalam format xml. Selanjutnya dilakukan proses pengelompokan simpul, busur, serta fungsi OCL berdasar tipe diagram yang ada. Pada Diagram Aktivitas dan Diagram Statechart diambil busur dan simpul. Sedangkan pada Diagram Kelas diambil nama fungsi dan parameter fungsi tersebut. Untuk fungsi OCL diambil bagian ”pre” dan “post” nya. Berikut gambar sub sistem poliklinik.

Diagram Aktivitas sub sistem kedatangan pasien menceritakan tentang alur pasien pada sebuah poliklinik. Poliklinik tersebut mempunyai dua buah poli, diantaranya poli umum dan poli gigi. Apabila pasien akan menuju poli umum maka pasien harus mengantri pada ruangan 1, sedangkan pasien yang menuju poli gigi akan mengantri pada ruangan 2 sebagaimana dapat dilihat pada Gambar 4.



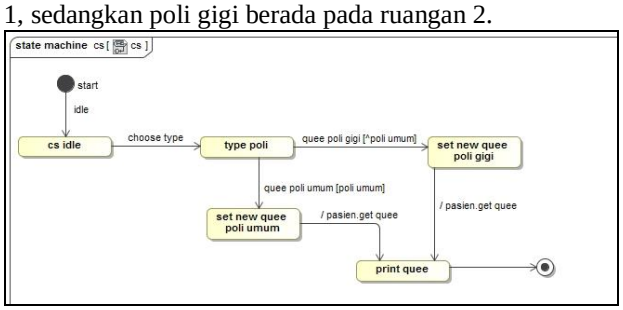
Gambar 4 Diagram Aktivitas Poliklinik

Gambar 5 menunjukkan transisi status pada pasien, dimulai dengan pasien datang kepada CS. Selanjutnya pasien akan memilih untuk menuju poli umum atau tidak.Selanjutnya CS akan mengarahkan ruangan yang harus dituju oleh pasien sesuai dengan ketentuan yang ada.



Gambar 5 Diagram *Statechart* pasien

Gambar 6 menunjukkan transisi status pada cs, dimulai dengan signal dari pasien yang memilih tipe poli yang dituju. Apabila pasien memilih poli umum maka dilakukan pengesetan antrian pada poli umum. Sebaliknya, apabila pasien tidak memilih poli umum, akan dilakukan pengesetan antrian pada poli gigi.Setelah semua selesai maka sistem akan meminta pasien untuk menuju ruangan yang telah disediakan, dimana poli umum berada pada ruangan 1, sedangkan poli gigi berada pada ruangan 2.



Gambar 6 Diagram *Statechart* cs

Fungsi OCL pada sub sistem ini terdapat pada fungsi tertentu, diantaranya:

- choose poli, memiliki fungsi OCL
  - Pre ”poli=umum”, Post “quee=poli umum”,
  - Pre ”poli=gigi”, Post “quee=poli gigi”,
- quee poli umum, memiliki fungsi OCL
  - Pre “quee=poli umum”,
  - Post “message silahkan menuju ruangan 1",
- quee poli gigi, memiliki fungsi OCL
  - Pre “quee=poli gigi”,

  Post “message silahkan menuju ruangan 2";

Dari diagram tersebut selanjutnya dibuat diagram SAD-S. Hasilnya Kasus uji dengan referensi id=2 dibaca alurnya. Setiap alur yang dilewati dilakukan pengecekan apakah ada fungsi OCL yang mendefinisikan simpul tersebut. Ketika simpul mengacu pada fungsi choose poli dilakukan pengecekan apakah simpul tersebut bertipe percabangan. Apabila memiliki percabangan maka dilakukan pengecekan simpul setelah simpul choole poli. Simpul selanjutnya adalah simpul dengan fungsi OCL pre: quee=poli gigi. Simpul choose poli sendiri memiliki dua buah fungsi OCL.

Untuk memilih fungsi OCL mana yang dieksekusi dilakukan pencarian fungsi OCL yang memiliki kondisi post: quee=poli gigi, oleh karena itu dipilih fungsi OCL Pre: poli=gigi, Post: quee=poli gigi. Dari fungsi OCL yang diperoleh maka variabel yang dimasukkan adalah poli=gigi. Karena simpul quee poli gigi bukan bertipe percabangan, maka dipilih fungsi OCL Pre: quee=poli gigi Post “message silahkan menuju ruangan 2", sehingga diset input quee=poli gigi, dan output “message silahkan menuju ruangan 2”.

```
ID kasus uji: 2
input: poli=gigi
input: quee=poli gigi
output: message"silahkan menuju ruangan 2"
```

Gambar 7 Data Uji Pertama Skenario 1

Kasus uji dikatakan sesuai dengan sarat kasus uji melalui simpul dan busur yang dikehendaki atau sesuai dengan alur pada diagram aktivitasnya.

Pada Skenario 1 jumlah keseluruhan kasus uji sebanyak empat buah, sedangkan kasus uji yang sesuai sebanyak tiga buah. Apabila diambil semuanya data uji yang sesuai sebanyak tiga buah. Data uji yang dihasilkan pada skenario 1 antara lain:

Tabel 2. Hasil Pengujian Skenario 1

Kasus uji	Input	Output
choose poli umum=y	poli= umum, quee= poli umum	Message "silahkan menuju ruangan 1"
choose poli umum=y	poli=umum, quee= poli umum	Message "silahkan menuju ruangan 1"
choose poli umum=t	poli= gigi, quee= poli gigi	Message "silahkan menuju ruangan 2"

Tabel 3. Daftar Alur Kasus Uji Skenario 1

Kasus Uji	Alur
choose poli umum=y	1, 3, 2, 5, 4, 44, 8, 15, 12, 20, 19, 46, 18, 26, 25, 47, 27, 32, 31, 48, 33, 38, 37
choose poli umum=y	1, 3, 2, 7, 6, 24, 19, 46, 18, 26, 25, 47, 27, 32, 31, 48, 33, 38, 37
choose poli umum=t	1, 3, 2, 5, 4, 44, 8, 15, 12, 20, 19, 46, 18, 26, 25, 49, 39, 43, 31,48, 33, 38, 37
-	1, 3, 2, 5, 4, 44, 8, 15, 12, 45, 16, 30, 31, 48, 33, 38, 37

Data uji yang dihasilkan dari skenario 1 memiliki kesesuaian sebanyak 75% dikarenakan kasus uji yang dibuat tidak melalui suatu perulangan, sehingga memperkecil peluang kesalahan alur uji yang diinginkan. Pada skenario ini juga memiliki sedikit busur bertipe sinyal, sehingga memperkecil percabangan yang ada pada diagram SAD-S.

Kasus uji keempat dinilai tidak sesuai karena tidak menghasilkan data uji dari kasus uji yang ada. Kasus uji keempat ini tidak melalui simpul percabangan seperti yang diharapkan sehingga dapat dikatakan kasus uji tersebut tidak sesuai. Dikarenakan kasus uji yang tidak sesuai tersebut, maka dapat dipastikan bahwa data uji yang dihasilkan juga tidak sesuai.

Untuk mengitung prosentase kesesuian suatu kasus uji menggunakan rumus sebagai berikut:

$$s = \frac{\sum kb}{\sum k} * 100\% \quad \text{.....1}$$

Dimana s = prosentase kesesuaian,

$\sum kb$ = total data uji yang seusai,

dan $\sum k$ = total data uji yang dihasilkan.

Berikut tabel kesesuaian pada masing – masing skenario.

Tabel 4. Perbandingan Hasil Pengujian

Kriteria	Skenario 1	Skenario 2	Skenario 3
Keseluruhan kasus uji	75%	60%	50%
Kasus uji dengan pemilihan kasus uji yang sesuai	100%	100%	100%

4. Kesimpulan

Kesimpulan yang dapat diambil berdasarkan hasil rangkaian uji coba dan analisa penelitian yang dilakukan terhadap metode yang diusulkan. Kesimpulan tersebut adalah sebagai berikut.

a. Pembangkitan data uji secara otomatis dapat dibuat dengan memanfaatkan Diagram tambahan, diantaranya Diagram Kelas serta kamus data dalam bentuk OCL yang mendefinisikan fungsi yang ada pada Diagram Kelas.
b. Data uji dapat dibuat dengan menggunakan kamus data dalam format OCL. Fungsi OCL yang digunakan adalah tag "pre" dan "post.
c. Data uji dapat dibentuk dengan menyesuaikan kasus uji yang sudah ada dengan membaca alur dari setiap simpul yang dilewati oleh kasus uji.
d. Data uji yang dihasilkan tidak hanya berupa pernyataan salah atau benar saja, tetapi dapat berupa nilai dari variabel tertentu yang telah didefinisikan pada OCL.
e. Tingkat rata – rata kesesuaian data uji pada keseluruhan skenario pengujian mencapai 62%. Hasil tersebut dikarenakan kasus uji yang tidak sesuai/tidak benar tetap diproses untuk pembangkitan data ujinya. Apabila pada keseluruhan skenario dilakukan pemilihan kasus uji yang sesuai/benar saja diperoleh nilai rata – rata kesesuaian data uji sebanyak 100%.

Dari hasil penelitian yang telah dilakukan dapat dilihat suatu poin pada metode ini perlu adanya pengembangan fungsi OCL yang dapat dimanfaatkan untuk membuat data uji dari kasus uji yang sudah ada. fungsi OCL digunakan pada penelitian ini adalah fungsi "pre" dan "post" yang mendifinisikan fungsi pada suatu Diagram Kelas. Meskipun data uji yang dihasilkan dapat bervariasi, tetapi variabel yang didefinisikan pada funsi ini bertipe benar dan salah. Dari Fungsi tersebut dapat dibuat sematiknya untuk membuat data yang memiliki jangkauan lebih luas.

Referensi

[1] Hasling, Bill, "Model Based testing of system requirements using UML Use Case Models", ICST, 2008

[2] Dharwiyanti Sri, Wahono Romi Satria, "Pengantar Unified Modelling Language", 2003;http://ikc.dinus.ac.id/umum/yanti-uml.php, diakses pada 2 Maret 2018

[3] Mingsong, Chen, "Automatic Test Case Generation for UML Activity Diagrams", ACM, 2006

[4] Swain, Kumar Santos, "Test Case Generation Based on State and Activity Models", JOT, 2010

[5] Setia, "Kamus Data", 2009; http://setia.staff.gunadarma.ac.id/Downloads/files/../ Modul Kamus Data.pdf, diakses pada tanggal 2 Maret 2018

[6] Anonymous, "Object Constraint Language", http://www.omg.org ,diakses pada 2 Maret 2018

[7] Kansomkeat, Supaporn, "Automated-Generating Test Case Using UML *Statechart* Diagram", SAICST, 2003

[8] Sarma Monalisa, "Automatic Test Case Generation from UML Sequence Diagram", ADCOM, 2007

Halaman Sengaja Dikosongkan