DINAMIKA REKAYASA
http://dinarek.unsoed.ac.id

Vol. 16 No. 1 (2020) Hal. 11 - 20
p-ISSN 1858-3075 | e-ISSN 2527-6131

TEST CASE GENERATION BERBASIS *STATE MODEL* UNTUK VERIFIKASI SISTEM LAYANAN PERMOHONAN ROHANIWAN

TEST CASE GENERATION BASED ON STATE MODEL TO VERIFY ROHANIWAN APPLICATON SERVICE SYSTEM

Defri Kurniawan[*1], Danang Wahyu Utomo[2], Novita Kurnia Ningrum[3]

*Email: defri.kurniawan@dsn.dinus.ac.id

[123]Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro

Abstrak— Pembuatan kasus uji (*test case generation*) merupakan tahapan yang membutuhkan sumber daya terbesar yang memiliki pengaruh terhadap keefektifan dan efisiensi suatu pengujian perangkat lunak. Pembuatan *test case* menjadi salah satu topik penelitian paling menarik. Pengujian berbasis model (*model based testing*) diusulkan untuk membuat kasus uji pada Sistem Layanan Permohonan Rohaniwan Kementerian Agama Provinsi Jawa Tengah. Model yang diusulkan dalam pembuatan kasus uji dimulai dari kegiatan pengumpulan kebutuhan, menganalisis *use case* dan *class*, mengidentifikasi *state*, melakukan pemodelan perilaku (*behaviour modelling*) menggunakan *state machine diagram* dan membuat daftar kasus uji. Penelitian menunjukkan penggunaan model berbasis *state* mampu mendukung pembuatan kasus uji (*test case*) dan dapat mendeteksi perilaku (*behavior*) dari *response* sistem yang kurang sesuai terhadap inputan atau aksi yang diberikan oleh *user.*

Kata kunci — kasus uji, pembuatan kasus uji, pengujian berbasis model, *state model*, rohaniwan

Abstract— Making test case (test case generation) is a stage that requires the greatest resources that have an influence on the effectiveness and efficiency of a software test. Making a test case is one of the most interesting research topics. Model based testing is proposed to make a test case on the Ministry of Religion's Central Java Provincial Ministry of Religion Request Service System. The model proposed in making test cases starts from the activity of gathering needs, analyzing use cases and classes, identifying states, conducting behavior modeling (behavior modeling) using state machine diagrams and making a list of test cases. Research shows that the use of state-based models can support the making of test cases and can detect behaviors from response systems that are not suitable for input or actions given by the user.

Keywords — test case, test case generation, model based testing, state model, rohaniwan

I. PENDAHULUAN

Pengujian perangkat lunak (*software testing*) merupakan proses melakukan eksplorasi komponen atau sistem secara sistematis dengan tujuan utama untuk menemukan dan melaporkan cacat (*defect*) [1]. Selain itu, pengujian perangkat lunak didefinisikan sebagai suatu proses evaluasi perangkat lunak dalam memenuhi kebutuhan yang telah ditetapkan [2]. Maka pengujian perangkat lunak menjadi proses yang penting dalam penentuan lolos tidaknya suatu perangkat lunak yang telah dibangun. Proses pemeriksaan perangkat lunak apakah telah memenuhi spesifikasi kebutuhan yang ditetapkan atau tidak, biasa disebut dengan *Verification and Validation* (V&V) [3]. Pada praktiknya, dokumen spesifikasi kebutuhan perangkat lunak (*software requirements specification document*) digunakan sebagai acuan terhadap proses evaluasi tersebut.

Black box testing merupakan suatu teknik pengujian yang menguji fungsionalitas dari suatu sistem tanpa memeriksa kedalaman level implementasinya [2]. *Black box testing* juga disebut sebagai *Specification-based techniques* [1], dimana kasus uji (*test case*) diperoleh langsung dari spesifikasi kebutuhan perangkat lunak terkait apa yang harus dilakukan oleh sistem. Pengujian *black*

box juga dapat digunakan dalam mengukur kinerja fungsionalitas suatu sistem [4].

Test case merupakan input yang diperlukan untuk memeriksa apakah perangkat lunak memenuhi persyaratan yang ditentukan dalam kondisi tertentu [5]. Pembuatan *test case* merupakan tahap yang membutuhkan sumber daya terbesar yang berpengaruh terhadap keefektifan dan efisiensi pengujian perangkat lunak [6]. Tantangan tersebut menjadikan pembuatan *test case* menjadi salah satu topik penelitian paling aktif pada pengujian perangkat lunak. Hal tersebut dibuktikan dengan berbagai teknik yang diusulkan selama beberapa dekade ini.

Diagram UML dapat digunakan untuk menghasilkan *test case* seperti penggunaan *Activity Diagram, Sequence Diagram* [7], maupun *Use Case Diagram* dan *State Transition Diagram* [5]. Pengujian berbasis pada model dalam menghasilkan *test case* dikenal dengan *Model Based Testing* (MBT) [8]. Topik MBT menarik dunia industri dan akademisi karena penyediaan pengujian yang sistematis, otomatis, dan komprehensif [9]. Sehingga penelitian pada bidang MBT akan terus berkembang.

State Transition Testing merupakan salah satu teknik pengujian berbasis spesifikasi (*specification-based techniques*) yang memanfaatkan penggunaan *State Transition Diagram* sebagai model [2]. Penggunaan model berorientasi pada *state* (*state model*) merupakan pendekatan yang sering digunakan dalam menghasilkan *test case* [6]. Berkembangnya kajian yang dilakukan para peneliti terhadap pengujian berbasis *state model*, menjadikan pendekatan ini terbukti dapat memberikan solusi dan berkontribusi positif pada pengujian perangkat lunak.

Penilitian yang dilakukan oleh [5] mengusulkan model pengujian berbasis *state* dan teknik pembuatan *test case* untuk pengujian pada sistem tertanam yang diterapkan pada alat pemutar MP3. Proses pengujian dijalankan dari pemodelan *use case*, *block diagram*, mengidentifikasi *state*, pembentukan *state diagram* dan menghasilkan *test case*. Penelitian menghasilkan interaksi antara komponen sistem tertanam dapat dengan mudah diketahui, sehingga membantu deteksi kesalahan dengan mudah.

Penelitian [10] mengusulkan sekumpulan algoritma yang digunakan untuk mengasilkan *test case* dari *state chart diagram* berdasarkan berbagai *coverage criteria*. Berbagai *coverage criteria* seperti *All Transition* (AT), *Round Trip Path* (RTP) dan *All Transition Pair* (ATP) dipertimbangkan pada penelitian ini. Pengujian dilakukan pada *Stack Operation* dan *Vending Machine Automation System*. Penelitian menghasilkan AT mengkonsumsi sumber daya uji terbanyak, ATP tidak dapat mencapai cakupan transisi 100%, dan *test case* yang dihasilkan berdasarkan RTP adalah efisien.

Penelitian lain dari studi kasus industri dilakukan untuk menguji efektivitas biaya menggunakan pengujian berbasis *state* [11]. Pada penelitian tersebut MBT *Tool* digunakan untuk menghasilkan *test case* secara otomatis, hasilnya terdapat 26 kesalahan nyata yang dapat ditemukan. Penelitian tersebut menunjukkan bahwa model pengujian yang terperinci dan pengujian sistem yang ketat dapat meningkatkan kemampuan deteksi kesalahan pada *System Under Test* (SUT).

Penelitian dilakukan oleh [9] meninjau aplikasi dengan teknik berbasis pencarian / *Search Based Technique* (SBT) yang diterapkan untuk *Model Based Testing* (MBT) [9]. Penelitian dilakukan untuk menganalisis secara komprehensif keadaan terkini dari aplikasi eksperimental SBT untuk MBT dan menyajikan keterbatasan literatur yang mengarah pada penelitian masa depan. Penelitian dilakukan dengan 72 makalah dan enam sumber data. Hasil penelitian menunjukkan bahwa sebagian besar aplikasi SBT fokus pada cakupan fungsional dan struktural, sebagai lawan dari *stress testing, regression testing,* dan *Graphical User Interface* (GUI) *testing.*

Penelitian lain dari pengujian dengan penggunaan *state model* adalah penggunaan *Extended Finite State Machines* (EFSMs) yang dilakukan oleh El-Fakih, dkk [12]. Pada penelitian tersebut menyajikan penilaian jenis kriteria pemilihan tes EFSM yang paling dikenal yaitu *suite* tes yang mencakup *edge-pair, prime path, prime path with side trip,* dan semua kriteria penggunaan yang diturunkan dari *graph* dan representasi *flow-graph.* Para peneliti menemukan *suite* tes acak memiliki kinerja yang baik dibandingkan dengan strategi derivasi tes lainnya.

Sistem layanan permohonan rohaniwan merupakan layanan yang dapat digunakan oleh masyarakat dalam mengajukan permohonan rohaniwan dalam melakukan penyuluhan atau pembimbingan rohani [13]. Pada penelitian ini menyajikan teknik pengujian untuk menghasilkan *test case* yang berbasis pada *state model* dengan menggunakan diagram UML. Pengujian dilakukan untuk melakukan verifikasi terhadap fungsionalitas sistem. Dalam hal ini, *domain* aplikasi yang diuji adalah Sistem Layanan Permohonan Rohaniwan yang terdapat pada kantor Kementerian Agama Provinsi Jawa Tengah.

II. TINJAUAN PUSTAKA

A. Pengujian Berbasis Model

Pengujian berbasis model atau yang dikenal dengan *Model Based Testing* (MBT) merupakan suatu teknik pengujian *black box* yang memanfaatkan informasi yang terdapat pada *requirement model* sebagai dasar dari pembuatan *test case* [8]. Pengujian tersebut memerlukan lima langkah sebagai berikut:

1. Menganalisa model perilaku yang tersedia untuk perangkat lunak atau membuat model perilaku baru.
2. Menelusuri model perilaku dan menentukan *input* yang membuat perangkat lunak melakukan transisi dari *state* ke *state.*
3. Meninjau model perilaku dan mencatat *output* yang diharapkan, saat perangkat lunak melakukan transisi dari *state* ke *state*.
4. Melaksanakan kasus uji (*test case*).
5. Membandingkan hasil aktual dan hasil yang diharapkan, serta melakukan tindakan korektif yang dibutuhkan.

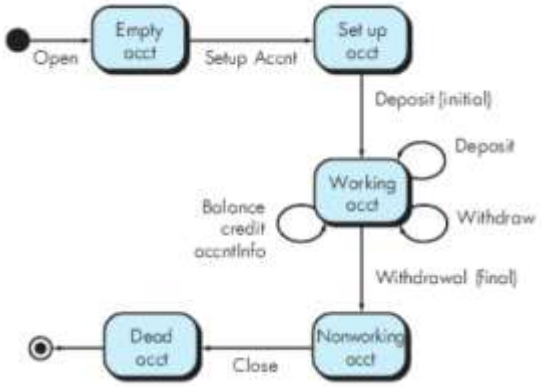
Teknik MBT dengan menggunakan UML *State Diagram* merupakan kasus yang sering ditemui yang membantu mengungkap kesalahan dalam perilaku perangkat lunak.

B. State Diagram

State diagram digunakan untuk memodelkan perilaku (*behaviour modeling*) dan termasuk salah satu elemen perilaku pada model analisis [8]. Elemen perilaku pada model analisis, digunakan untuk menggambarkan bagaimana *event external* mengubah keadaan (*state*) sistem atau kelas yang berada di dalamnya. Pada Gambar-1, menunjukkan contoh *state diagram* untuk kelas *account*.

State diagram menunjukkan keadaan sistem (*state*) dan peristiwa (*event*) yang menyebabkan transisi dari satu keadaan ke keadaan lainnya [3]. Pada UML 2.0 *state diagram* dikenal sebagai *state machine diagram* yang termasuk pada jenis diagram perilaku (*behaviour diagram*) [14]. Pembuatan model menggunakan *behaviour diagram* dilakukan dengan beberapa langkah berikut [8]:

1. Mengevaluasi semua *use case* untuk dapat memahami sepenuhnya urutan interaksi pada sistem.
2. Mengidentifikasi *events* yang memicu urutan interaksi dan memahami bagaimana *events* tersebut berhubungan dengan *object* tertentu.
3. Membuat urutan (*sequence*) untuk setiap *use case.*
4. Membangun *state diagram* untuk sistem.
5. Melakukan *review* model perilaku untuk memverifikasi akurasi dan konsistensi.



Gambar-1. *State diagram* untuk kelas *account* [8].

C. Verifikasi dan Validasi

Proses verifikasi dan validasi berkaitan dengan proses pemeriksaan terhadap perangkat lunak yang dikembangkan, apakah telah memenuhi spesifikasi yang ditentukan dan memberikan fungsionalitas yang diharapkan oleh orang-orang yang membiayai perangkat lunak tersebut [3]. Proses pemeriksaan tersebut, dimulai setelah persyaratan tersedia dan berlanjut selama proses pengembangan.

Verifikasi merujuk pada serangkaian tugas yang dilakukan untuk memastikan bahwa perangkat lunak telah diimplemenasikan dengan benar sesuai fungsi yang spesifik [8]. Verifikasi dilakukan untuk memeriksa apakah produk kerja telah memenuhi persyaratan yang telah ditentukan [1]. Verifikasi membantu memastikan bahwa pengembang telah membangun produk dengan cara yang benar.

Validasi mengacu pada serangkaian tugas berbeda untuk memastikan bahwa perangkat lunak yang telah dibangun dapat dilacak ke persyaratan pelanggan (*customer requirements*) [8]. Validasi dilakukan untuk mengevaluasi produk kerja apakah telah sesuai terhadap kebutuhan pengguna [1]. Validasi memastikan bahwa perilaku produk kerja sesuai dengan kebutuhan pelanggan seperti yang didefinisikan.

D. Sistem Layanan Permohonan Rohaniwan

Rohaniwan adalah orang yang memiliki kompetensi untuk membimbing keagamaan orang lain sebagai landasan dalam berperilaku [15]. Secara khusus peran Rohaniwan di Kementerian Agama RI, dilakukan oleh penyuluh agama yang merupakan pejabat berwenang yang diberikan amanah untuk melakukan pembinaan atau penyuluhan keagamaan pada kelompok masyarakat tertentu [16]. Jika pembinaan atau penyuluhan bagi masyarakat beragama Islam maka pekerjaan rohaniwan di bawah naungan Bidang Urusan Agama Islam dan Pembinaan Syariah (Binsyar). Selain itu peran rohaniwan juga dilakukan pada pengambilan sumpah jabatan pada acara pelantikan pejabat di bawah koordinasi Biro Humas Kementerian Agama.

Pada Kementerian Agama Provinsi Jawa Tengah, layanan permohonan rohaniwan untuk kelompok masyarakat maupun instansi dilakukan melalui aplikasi berbasis web. Aplikasi tersebut dibangun untuk mengatasi *redundancy* data dari sistem lama yang masih menggunakan permohonan melalui pengisian formulir berbasis kertas [13]. Sehingga layanan permohonan rohaniwan dapat dilakukan secara terkomputerisasi dan *online*.

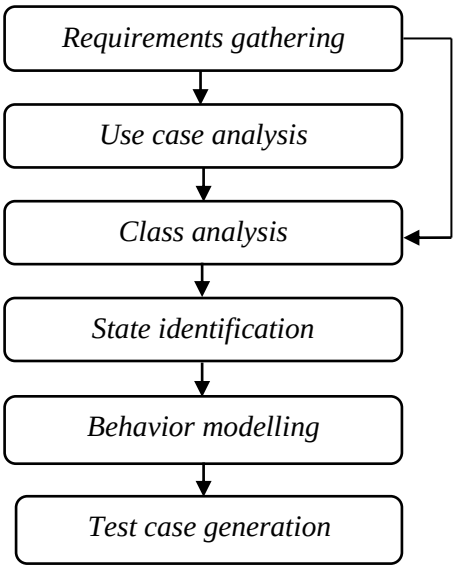
III. METODE PENELITIAN

Penelitian dilakukan untuk menghasilkan kasus uji (*test case*) terhadap sistem layanan permohonan rohaniwan di Kementerian Agama Provinsi Jawa Tengah, berdasarkan spesifikasi yang dibangun menggunakan diagram UML. Penelitian dimaksud untuk memberikan gambaran (deskripsi) terhadap objek yang diteliti dengan menguraikan pendekatan atau teknik yang dilakukan dalam menghasilkan *test case.*

Metode pengumpulan data yang digunakan adalah dengan melakukan wawancara dengan staf Biro Humas Kementerian Agama Provinsi Jawa Tengah dan studi dokumen digunakan untuk memodelkan sistem layanan permohonan rohaniwan. Jenis informasi yang dihasilkan pada penelitian ini bersifat kualitatif.

Langkah-langkah pembuatan *test case* berbasis *state model* didasari oleh model pengujian (*test model*) yang diterapkan pada *embedded system* [5]. *Test model* tersebut dilakukan perubahan berdasarkan pendekatan *Object Oriented Analyst and Design (OOAD)* dengan menggunakan diagram UML. Sehingga penulis mengganti *block diagram* dengan *class diagram* untuk menganalisis kelas yang disajikan pada Gambar-2. Adapun langkah-langkah pembuatan *test case* yang diusulkan meliputi:

1. Pengumpulan kebutuhan (*requirements gathering*)
2. Menganalisis *use case* (*use case analysis*)
3. Menganalisis kelas (*class analysis*)
4. Identifikasi keadaan (*state identification*)
5. Pemodelan perilaku (*behavior modelling*)
6. Pembuatan kasus uji (*test case generation*)



Gambar-2. Pembuatan *test case* yang diusulkan.

A. Pengumpulan Kebutuhan (*Requirements Gathering*)

Sistem layanan permohonan rohaniwan memiliki empat aktor yang berinteraksi dengan sistem, yaitu Admin Rohaniwan (Kemenag Prov. Jateng) yang kemampuannya dapat diturunkan kepada Admin Kemenag Kota/Kab dengan akses yang sama, SKPD/Instansi sebagai pengguna dan petugas Rohaniwan sebagai penerima informasi penugasan. Kebutuhan sistem secara fungsional meliputi kemampuan untuk melakukan *login*, melakukan *logout*, dan mengubah *password* dapat dilakukan oleh Admin Rohaniwan dan SKPD/Instansi. SKPD/Instansi dapat mendaftar sebagai pengguna (*member*) pada layanan rohaniwan untuk dapat mengajukan permohonan petugas rohaniwan, mengetahui daftar permohonan dan dapat mengunduh surat tugas rohaniwan yang telah ditandatangi oleh pejabat terkait. Admin rohaniwan dapat memasukkan data rohaniwan, memasukkan data SKPD/Instansi, melakukan penugasan kepada rohaniwan, dan mencetak surat tugas dan

mengunggah surat tugas. Petugas rohaniwan setelah ditugaskan dapat menerima pemberitahuan penugasan yang ditujukan melalui email.

B. Menganalisis *Use Case* (*Use Case Analysis*)

Kebutuhan-kebutuhan perangkat lunak yang telah didapat dari *requirement gathering* selanjutnya dilakukan analisis *use case* yang dilihat berdasarkan kebutuhan fungsional sistem.

Pekerjaan selanjutnya tingkat kepentingan (*importance level*) ditambahkan untuk mengukur tingkat kepentingan (prioritas) pada sistem. *Importance level* merupakan *element of a use case* yang menggambarkan aktivitas yang kritis (*high*), sedang (*medium*), atau rendah (*low*) [17]. Maka *use cases* yang telah teridentifikasi dilengkapi dengan *importance level* yang disajikan pada Tabel-1.

Selain *importance level*, *primary actor* juga diidentifikasi untuk melihat aktor yang memicu peristiwa yang akan ditanggapi oleh sistem.

Tabel-1. Daftar *use case* sistem layanan permohonan rohaniawan.

Use Case	*Importance Level*	*Primary Actor*
Melakukan l*ogin*	*Low*	*User*
Mengubah u*ser password*	*Low*	*User*
Melakukan *logout*	*Low*	*User*
Entry data rohaniwan	*High*	Admin
Entry data instansi/ SKPD	*High*	Admin
Menugaskan rohaniwan	*Medium*	Admin
Mencetak surat tugas	*Medium*	Admin
Upload surat tugas	*Medium*	Admin
Mendaftar sebagai pengguna (*Member*)	*Medium*	Instansi
Mengajukan permohonan rohaniwan	*High*	Instansi
Menampilkan daftar permohonan	*Low*	Instansi
Download surat tugas	*Medium*	Instansi
Menerima e*mail* penugasan	*Medium*	Rohaniwan

Tabel-1 menunjukkan bahwa *use case* mengajukan permohonan rohaniwan merupakan bagian terpenting dari sistem layanan permohonan rohaniwan sehingga dinyatakan memiliki tingkat kepentingan yang kritis (*high*). Mendata instansi/SKPD (*use case entry* data instansi/SKPD) diperlukan sebagai master data pemohon dan *entry* data rohaniwan digunakan untuk memilih rohaniwan, sehingga mereka juga memiliki tingkat kepentingan yang sama (*high*).

C. Menganalisis Kelas (*Class Analysis*)

Pada tahap analisis kelas, hal-hal yang berpotensi sebagai kelas (*class*) dianalisis berdasarkan *requirement gathering* yang telah dilakukan. Hasil dari analisis *use case* juga dipertimbangkan untuk menentukan kelas (*class*). Pada umumnya kelas (*class*) muncul dari perwujudan/manifestasi dari beberapa hal berikut seperti *external entities, things, occurrences or events, roles, organizational units, places, structures* [8]. Sehingga hasil kelas-kelas analisis sistem layanan permohonan rohaniwan dapat disajikan pada Tabel-2. *Basic operations* pada tiap *class* disesuaikan pada tahap *requirement gathering*.

Tabel-2. Daftar kelas analisis sistem layanan permohonan rohaniwan

Class	*Basic Operations*	*Class Category*
AdminProvinsi	- LoginAdmin() - LogoutAdmin()	*Roles, External Entities*
AdminKotaKab	- LoginAdmin() - LogoutAdmin()	*Roles, External Entities*
InstansiSKPD	- Registrasi() - LoginUser() - LogoutUser()	*Organizational Units, External Entities*
Rohaniwan	- Create()	*Roles, External Entities*
Permohonan	- Create() - Show()	*Things*
Penugasan	- Create() - SendtoEmail()	*Things*
SuratTugas	- Create() - Upload() - Download() - Print()	*Things*

D. Identifikasi Keadaan (*State Identification*)

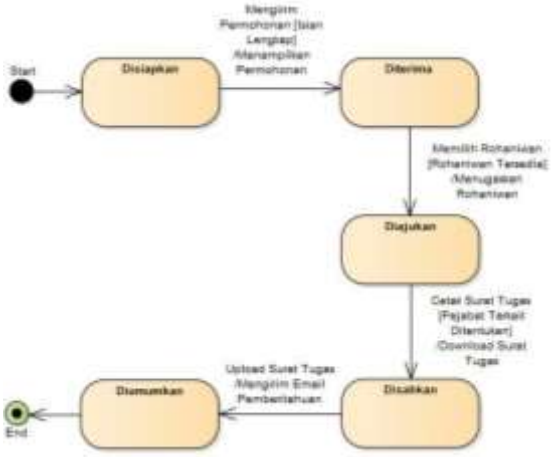
Pada sistem layanan permohonan rohaniwan, kebutuhan fungsional mengajukan permohonan rohaniwan merupakan kebutuhan fungsional utama yang harus dibangun, sehingga *use case* mengajukan permohonan rohaniwan memiliki tingkat kepentingan (*importance level*) yang tinggi (*high*). *Class* permohonan memiliki operasi membuat permohonan rohaniwan (*create*) yang mendukung sukses tidaknya suatu permohonan yang diajukan oleh Instansi/SKPD.

Identifikasi keadaan (*state*) dimulai dengan memilih salah satu *class* dari kelas-kelas analisis yang telah didapat [8]. *Class* permohonan digunakan untuk memodelkan perilaku sistem terhadap layanan permohonan rohaniwan. Adapun hasil identifikasi keadaan (*state*) dari Sistem Layanan Permohonan Rohaniwan meliputi:

1. Disiapkan, selanjutnya disingkat S1.
2. Diterima, selanjutnya disingkat S2.
3. Diajukan, selanjutnya disingkat S3.
4. Disahkan, selanjutnya disingkat S4.
5. Diumumkan, selanjutnya disingkat S5.

E. Pemodelan Perilaku (*Behavior Modelling*)

Diagram berbasis s*tate* digunakan untuk memodelkan perilaku (*behavior modelling*) dari suatu objek pada kelas permohonan. Berdasarkan hasil identifikasi keadaan (*state*) yang telah didapatkan yaitu disiapkan, diterima, diajukan, disahkan, dan diumumkan. Maka dapat dimodelkan perilaku suatu objek permohonan pada sistem layanan permohonan rohaniwan dengan menggunakan *state machine diagram* seperti pada Gambar-3.



Gambar-3. *State machine diagram* sistem layanan permohonan rohaniwan.

Pada *state machine diagram* pada Gambar-3, keadaan awal *object* permohonan berada pada keadaan Disiapkan. Setelah pengguna (instansi/SKPD) mengirimkan permohonan yang telah diisi lengkap, maka sistem dapat menampilkan permohonan yang dapat dilihat oleh admin. Keadaan berubah menjadi "Diajukan" setelah admin memproses permohonan dengan memilih rohaniwan pada daftar rohaniwan yang telah tersedia. Pemilihan rohaniwan tersebut mengakibatkan rohaniwan mendapatkan suatu penugasan yang tercantum pada permohonan tersebut. Permohonan berubah menjadi "Disahkan" setelah surat tugas dicetak dengan mengisi data pejabat terkait yang akan menandatangani surat tugas tersebut. Surat tugas yang telah dicetak tersebut dapat di-*download* untuk dimintakan tanda tangan kepada pejabat terkait. Setelah surat tugas ditandatangi oleh pejabat terkait, surat tugas dapat diunggah sehingga keadaan berubah menjadi "Diumumkan". Petugas rohaniwan yang ditugaskan akan mendapat email pemberitahuan penugasan dan pengguna (instansi/SKPD) dapat melihat *progress* dari permohonan yang diajukan.

F. Pembuatan Kasus Uji (*Test Case Generation*)

State model sistem layanan permohonan rohaniwan pada Gambar-3, digunakan untuk menghasilkan tabel transisi keadaan (*state transition table*) yang dapat disajikan pada Tabel-3. Penggunaan alias atau singkatan pada *state* yang telah diinisialisasi pada identifikasi *state* digunakan untuk mempermudah penyajian kasus uji.

Tabel-3. *State transition table* sistem layanan permohonan rohaniwan

Initial State	*Event*	*Guard*	*Action*	*Final State*
S1	Mengirim permohonan	Isian lengkap	Menampilkan permohonan	S2
S2	Memilih rohaniwan	Rohani-wan tersedia	Menugaskan rohaniwan	S3
S3	Cetak surat tugas	Pejabat terkait ditentu-kan	Download surat tugas	S4
S4	Upload surat tugas	-	Mengirim email pemberitahuan	S5

Berdasarkan tabel transisi keadaan, selanjutnya akan dibuatkan tabel pengujian dengan menyajikan kasus uji (*test case*), skenario pengujian (*test scenario*), dan hasil yang diharapkan (*expected result*).

Terdapat delapan kasus uji (test case) yang dihasilkan berdasarkan state model pada Sistem Layanan Permohonan Rohaniwan. Delapan kasus uji tersebut dapat dikembangkan lebih lanjut misalkan dengan memberikan jenis inputan data yang valid dan tidak valid.

Tabel-4. Tabel pengujian berbasis *state model* sistem layanan permohonan rohaniwan

Test case Id	*Initial state*	*Final state*	*Test scenario*	*Expected result*
TC.S1.01	S1	S2	*User* mengisi form permohonan rohaniwan dengan isian lengkap dan mengirimkannya.	Sistem dapat menyimpan permohonan dan menampilkannya ke *user*.
TC.S1.02	S1	S2	*User* mengisi form permohonan rohaniwan dengan tidak lengkap dan mengirimkannya.	Sistem dapat menunjukkan isian data yang tidak lengkap.
TC.S2.01	S2	S3	*User* memilih rohaniwan yang tersedia dan menyimpannya.	Sistem dapat menyimpan rohaniwan yang telah terpilih dan menugaskannya.
TC.S2.02	S2	S3	*User* tidak memilih rohaniwan yang tersedia dan menyimpannya.	Sistem menginfokan rohaniwan harus dipilih.
TC.S3.01	S3	S4	*User* mengisi data pejabat terkait yang menandatangani surat tugas dan mencetaknya.	Sistem dapat menyimpan data surat tugas dan dapat diunduh.
TC.S3.02	S3	S4	*User* tidak mengisi data pejabat terkait yang menandatangani surat tugas dan mencetaknya.	Sistem menginfokan data pejabat terkait harus diisi.
TC.S4.01	S4	S5	*User* melampirkan surat tugas dan mengunggahnya.	Sistem mengirimkan email pemberitahuan ke email rohaniwan yang ditugaskan.
TC.S4.02	S4	S5	*User* tidak melampirkan surat tugas dan mengunggahnya.	Sistem menginfokan harus ada lampiran surat tugas.

Test scenario dibangun berdasarkan *event* dan kondisi (*guard*), sedangkan *expected result* disesuaikan dengan *action* atau *response* sistem dari *event* yang diterapkan.

IV. HASIL DAN PEMBAHASAN

Pada bagian ini menyajikan pengujian yang dilakukan berdasarkan tabel pengujian yang telah dihasilkan pada Tabel-4. Pengujian dimulai dari transisi *state* S1 ke S2 dengan dua skenario pengujian yang berbeda yaitu mengisi formulir permohonan secara lengkap (TC.S1.01) dan yang kedua secara tidak lengkap (TC.S1.02). Jenis pengisian yang berbeda tersebut, diterapkan berdasarkan kondisi (*guard*) yang dihasilkan pada *state model*. Data pengujian dari isian permohonan yang lengkap meliputi pengisian nomor permohonan, perihal permohonan, tanggal surat, nama acara, tanggal acara, waktu acara, alamat acara, tempat acara, keterangan, dan mencentang pilihan keagamaan petugas rohaniwan yang diminta serta melampirkan surat permohonan rohaniwan dalam bentuk pdf atau *images* yang ditunjukkan pada Gambar-4. Sedangkan pengisian secara tidak lengkap, tidak diisinya tanggal dan waktu acara serta tidak memilih keagamaan petugas rohaniwan. Pengujian dimulai dari pengisian data form permohonan yang tidak lengkap, lalu pada pengisian lengkap untuk efisiensi waktu pengujian.

Gambar-4. Pengisian secara tidak lengkap pada form permohonan rohaniwan.

Pengujian terhadap pengisian data permohonan rohaniwan secara tidak lengkap disajikan pada Gambar-5. Hasil pengujian, sistem dapat mengawal isian data yang tidak lengkap dari pengguna. Pengujian dengan data yang diisi secara lengkap

memiliki hasil yang sesuai harapan yang disajikan pada Gambar-6.

Gambar-5. Peringatan pengisian data tidak lengkap pada form permohonan rohaniwan.

Gambar-6. Pemberitahuan permohonan rohaniwan berhasil disimpan.

Pengujian selanjutnya beralih pada transisi *state* S2 ke S3. Pengujian dilakukan dengan dua skenario pengujian yaitu memilih rohaniwan dari daftar petugas rohaniwan yang tersedia (TC.S2.01) dan skenario pengujian yang kedua yaitu tidak memilih rohaniwan yang tersedia (TC.S2.02). Pengujian dilakukan dari tidak dipilihnya petugas rohaniwan dilanjutkan dengan memilih petugas rohaniwan.

Gambar-7. Peringatan petugas rohaniwan belum dipilih.

Pada pengujian TC.S2.02 admin tidak memilih petugas rohaniwan, maka sistem dapat memberikan peringatan bahwa petugas rohaniwan harus dipilih seperti pada Gambar-7. Sedangkan pada kasus uji TC.S2.01, dengan skenario pengujian admin memilih petugas rohaniwan. Sistem dapat menyimpan penugasan rohaniwan yang dipilih seperti yang ditunjukkan pada Gambar-8. Namun sistem tidak dapat memberitahukan kepada petugas rohaniwan yang bersangkutan.

Gambar-8. Pemberitahuan penugasan terhadap rohaniwan berhasil dilakukan.

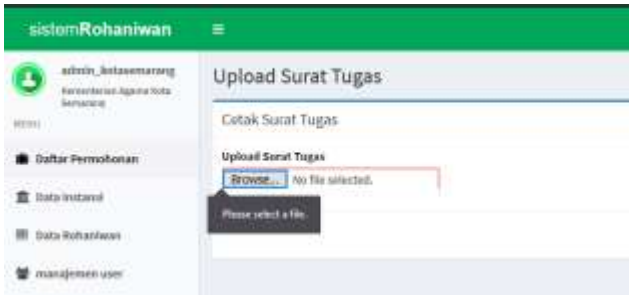
Selanjutnya pengujian dilakukan pada transisi S3 ke S4. Setelah petugas rohaniwan berhasil ditetapkan, surat tugas perlu dibuat sebagai tanda pengesahan penunjukkan petugas rohaniwan dari pejabat terkait. Skenario pengujian yang pertama, data pejabat terkait diisi dan skenario pengujian yang kedua, data pejabat terkait tidak diisi. Hasil pengujian dari kasus uji tidak diisinya data pejabat terkait, sistem dapat memberikan peringatan bahwa data pejabat terkait wajib diisi yang ditunjukkan pada Gambar-9.

Gambar-9. Peringatan data pejabat terkait wajib diisi.

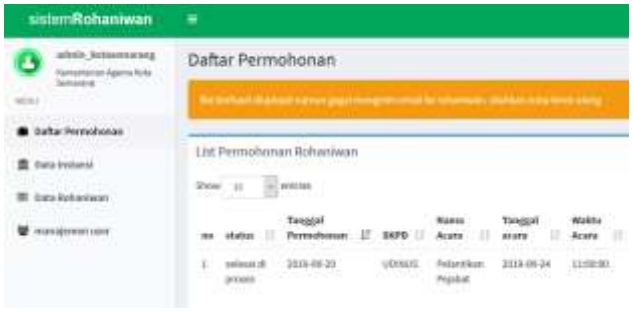
Sedangkan pada kasus uji data pejabat terkait diisi, sistem mampu membuat surat tugas dengan format pdf yang dapat diunduh untuk dimintakan tanda tangan kepada pejabat terkait. Keberhasilan sistem dalam membuat surat tugas ditunjukkan dengan menampilkan nofikasi atau pemberitahuan yang disajikan pada Gambar-10.

Gambar-10. Pemberitahuan surat tugas berhasil dibuat.

Pada pengujian transisi terakhir state S4 ke S5, pengujian dilakukan untuk menguji fungsi *upload* hasil *scan* surat tugas yang telah ditandatangani oleh pejabat terkait. Terdapat dua skenario pengujian yang dilakukan, pertama surat tugas tidak dilampirkan, kedua surat tugas dilampirkan. Hasil pengujian untuk skenario pengujian pertama, sistem dapat memberikan info bahwa surat tugas wajib dilampirkan yang ditunjukkan pada Gambar-11. Sedangkan hasil pengujian skenario kedua, sistem gagal mengirimkan info penugasan ke alamat email petugas rohaniwan yang ditunjukkan pada Gambar-12.



Gambar-11. Peringatan surat tugas wajib dilampirkan.



Gambar 12. Pemberitahuan *upload* surat tugas berhasil namun mengirim email gagal.

Sebanyak delapan kasus uji telah diuji sesuai dengan tabel pengujian (Tabel-4). Selanjutnya, hasil pengujian yang didapatkan (*actual result*) tersebut dibandingkan dengan hasil yang diharapkan (*expected result*) pada tabel pengujian (Tabel-4). Hasil pengujian yang didapatkan, dicatat dan ditentukan status pengujiannya yaitu "Lolos" atau "Gagal", seperti yang disajikan pada Tabel-5. Jika hasil pengujian sesuai dengan hasil yang diharapkan maka status pengujian adalah "Lolos". Sedangkan jika hasil pengujian tidak sama dengan hasil yang diharapkan, maka status pengujian adalah "Gagal".

Berdasarkan hasil pengujian yang dilakukan sebanyak tujuh kasus uji dinyatakan "Lolos". Sedangkan satu kasus uji dinyatakan "Gagal" yaitu *Test Case* Id TC.S4.01. Pada kasus uji tersebut, hasil yang diharapkan, sistem dapat mengirimkan email pemberitahuan ke email rohaniwan yang ditugaskan, namun hasil pengujian mendapatkan sistem tidak dapat mengirimkan pemberitahuan melalui email. Selengkapnya hasil pengujian (*actual result*) beserta status pengujiannya (status) pada setiap kasus uji dapat dilihat pada Tabel-5 di bawah ini.

Tabel-5. Hasil pengujian berbasis *state model* sistem layanan permohonan rohaniwan

Test Case Id	*Actual Result*	*Status*
TC.S1.01	Sistem dapat menyimpan permohonan dan menampilkannya ke *user*	Lolos
TC.S1.02	Sistem dapat menunjukkan isian data yang tidak lengkap	Lolos
TC.S2.01	Sistem dapat menyimpan rohaniwan yang telah terpilih dan menugaskannya	Lolos
TC.S2.02	Sistem menginfokan rohaniwan harus dipilih	Lolos
TC.S3.01	Sistem dapat menyimpan data surat tugas dan dapat diunduh	Lolos
TC.S3.02	Sistem menginfokan data pejabat terkait harus diisi	Lolos
TC.S4.01	Sistem dapat mengunggah surat tugas, namun tidak dapat mengirimkan pemberitahuan melalui email	Gagal
TC.S4.02	Sistem menginfokan harus ada lampiran surat tugas	Lolos

V. PENUTUP

A. Kesimpulan

Penggunaan model berbasis *state* mempu mendukung pembuatan kasus uji (*test case*) yang diterapkan pada sistem layanan permohonan rohaniwan. Elemen-elemen *state*, seperti *event* yang memicu terjadinya perubahan *state*, memberikan

kontribusi terhadap skenario pengujian dari aksi pengguna (*system input*). *Guard* sebagai kondisi yang mengawal *event* memberikan batasan terhadap persyaratan input dari skenario pengujian. Dan *action* sebagai *effect* dari transisi dapat menggambarkan *response* yang perlu dilakukan oleh sistem (*system output*).

Pengujian berbasis *state model* mampu mendeteksi perilaku (*behavior*) dari *response* sistem yang kurang sesuai terhadap inputan atau aksi yang diberikan oleh *user*. Dan memberikan saran perbaikan dari layanan yang harus sistem terapkan.

B. Saran

Berdasarkan pengujian yang dilakukan dengan *state model*, saran perbaikan untuk sistem layanan permohonan rohaniwan yaitu petugas rohaniwan perlu fasilitas untuk mengakses sistem. Sesuai dengan *state model*, penugasan rohaniwan didapatkan setelah admin memilih rohaniwan tersebut. Selain itu, setelah admin mencetak surat tugas, sistem otomatis melakukan *download* sehingga dapat segera dimintakan tanda tangan oleh pejabat terkait.

Penelitian selanjutnya, model yang telah diusulkan pada pengujian ini dapat diterapkan pada domain aplikasi untuk layanan permohonan dari masyarakat atau sistem dengan menggunakan disposisi surat tugas terhadap staf/petugas terkait.

ACKNOWLEDGEMENT

Terima kasih untuk Kementerian Agama Provinsi Jawa Tengah atas kepercayaan yang telah diberikan untuk dapat melakukan penelitian ini. Terima kasih juga kepada Universitas Dian Nuswantoro, progam studi Teknik Informatika S-1 Fakultas Ilmu Komputer dan Lembaga Penelitian dan Pengabdian Masyarakat (LPPM), sehingga kami dapat menyelesaikan jurnal ini.

DAFTAR PUSTAKA

[1] P. Hambling, Brian; Morgan, Peter; Samaroo, Angelina; Thompson, Geoff; Williams, *Software Testing An ISTQB-ISEB Foundation Guide Second Edition*. Swindon: British Informatics Society Limited, 2010.

[2] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software testing techniques: A literature review," in *Proceedings - 6th International Conference on Information and Communication Technology for the Muslim World, ICT4M 2016*; 2017;177–182.

[3] I. Sommerville, Software Engineering, Ninth Edition. Addison-Wesley, 2010.

[4] E. R. Subhiyakto and Y. P. Astuti, "Pengembangan Aplikasi Penjadwalan Rapat Menggunakan Metode Phased Development," Dinamika Rekayasa, 2019; 15 (1); 32–42.

[5] S. Y. Jeong, C. J. Yoo, and H. M. Noh, "State transition based test model and test case generation technique for embedded system: An empirical approach," Int. J. Softw. Eng. its Appl., 2016; 10(11); 233–254.

[6] N. Setiani, R. Ferdiana, P. I. Santosa, and R. Hartanto, "Literature Review on Test Case Generation Approach," Proc. 2nd Int. Conf. Softw. Eng. Inf. Manag., 2019; 91–95.

[7] Meiliana, I. Septian, R. S. Alianto, Daniel, and F. L. Gaol, "Automated Test Case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm," Procedia Comput. Sci., 2017; 116; 629–637.

[8] M. Roger S, Pressman; Bruce R, Software Engineering: A Practitioner's Approach 8th Edition. McGraw-Hill Education, 2015.

[9] A. Saeed, S. H. Ab Hamid, and M. B. Mustafa, "The experimental applications of search-based techniques for model-based testing: Taxonomy and systematic literature review," Appl. Soft Comput. J., 2016; 49; 1094–1117.

[10] S. Pradhan, M. Ray, and S. Kumar, "Transition coverage based test case generation from state chart diagram," J. King Saud Univ. - Comput. Inf. Sci., 2019.

[11] N. E. Holt, L. C. Briand, and R. Torkar, "Empirical evaluations on the cost-effectiveness of state-based testing: An industrial case study," Inf. Softw. Technol., 2014; 56(8); 890–910.

[12] K. El-Fakih, A. Simao, N. Jadoon, and J. C. Maldonado, "An assessment of extended finite state machine test selection criteria," J. Syst. Softw., 2017; 123; 106–118.

[13] D. Kurniawan and W. Utomo, Danang, "Pengembangan Model-View-Controller Pada Sistem Permohonan Rohaniwan Kementerian Agama Provinsi Jawa Tengah," 2019; 27–34.

[14] D. Brahma, Object-Oriented Analysis and Design. Springer, 2010.

[15] Kamus Ilmiah Populer. Surabaya: Arkola, 1994.

[16] M. Muslihudin, D. Kurniawan, and I. Widyaningrum, "Implementasi Model Fuzzy SAW Dalam Penilaian Kinerja Penyuluh Agama," J. TAM (Technol. Accept. Mod*el)*, 2017: 8(1); 39–44.

[17] A. Dennis, *System Analysis and Design 4th Edition*. John Wiley and Sons, 2009.