Pengujian Aplikasi Pengenalan Tulisan Tangan Menggunakan Model *Behaviour Use Case*

Gita Fadila Fitriana
Institut Teknologi Telkom Purwokerto
Program Studi Rekayasa Perangkat Lunak Fakultas Informatika
e-mail: gita@ittelkom-pwt.ac.id

Abstrak

Pengembangan perangkat lunak yang baik harus melalui proses pengujian tiap fase. Pengujian digunakan untuk mencari kesalahan perangkat lunak yang dikembangkan. Pengujian dilakukan berdasarkan kebutuhan pengembangan perangkat lunak. Pengujian ini bertujuan memvalidasi kondisi awal dan kondisi akhir sesuai dengan kebutuhan. Pada penelitian ini pengujian antarmuka dilakukan dengan model behaviour Unifed Modelling Language (UML) yaitu use case diagram. Berdasarkan hasil uji, terdapat 15 pengujian yang dilakukan terdapat 1 pengujian yang ditolak, sehingga nilai presentase pengujian 93%. Pengujian dilakukan sesuai scenario use case. Hasil pengujian menunjukan bahwa aplikasi dapat berjalan sesuai scenario serta dapat mengetahui kualitas dari aplikasi pengenelan tulisan tangan.

Kata kunci—*Behaviour,* UML, *Use case*, *Use case* Skenario, Tulisan tangan

Abstract

Good software development must go through a testing process every phase. Testing is used to find software problems that are developed. Testing is done based on software development needs. This test validates the initial and final conditions as needed. In this study, testing with the behavior of the Unifed Modelling Language (UML) model, namely the use case diagram. Based on the test results, there were 15 tests conducted there was 1 test that was rejected, then the percentage of testing was 93%. Testing is done according to the use case scenario. The test results show the application can run according to the scenario and can find the quality of the handwriting application.

Keywords— *Behaviour,* UML, *Use case*, *Use case* Scenario, Handwriting

1. PENDAHULUAN

Perangkat lunak yang sesuai dengan standar harus melalui proses pengembangan perangkat lunak atau yang biasa disebut *Software Development Life Cycle* (SDLC). Model siklus hidup pengembangan perangkat lunak (SDLC) mencakup rentang dari prediksi ke adaptif. SDLC prediktif adalah ditandai dengan pengembangan persyaratan perangkat lunak terperinci, perencanaan proyek terperinci, dan perencanaan minimal untuk iterasi di antara fase pengembangan [1]. Pengujian sistem merupakan fase yang wajib diperhatikan serta dibutuhkan untuk meningkatkan kualitas sistem yang sudah ada agar berjalan sesuai fungsinya. Setiap proses dari pengembangan perangkat lunak harus melalui tahap pengujian, sebab proses kebutuhan (*requirement*), analisis, perancangan dan implementasi tidak terlepas dari kesalahan [2] [2]. Adapun beberapa aturan yang digunakan untuk melakukan pengujian perangkat lunak

yaitu pengujian dilakukan sebuah proses eksekusi program dengan tujuan utama mencari kesalahan, kemungkinan penemuan kesalahan yang tinggi dapat dikatakan baik pada sebuah kasus pengujian dan pengujian menemukan kesalahan merupakan pengujian yang berhasil [3].

Jaminan mutu dari sebuah sistem merupakan bagian penting dalam peningkatan sistem, peningkatan sistem didasarkan pada analisa kebutuhan (*requirement*) perangkat lunak, pengujian sistem ini tidak memiliki metode dan kiteria formal. Dikarenakan hal tersebut membuat hasil pengujian bisa menjadi tidak konsisten, maka dari itu diperlukan alat bantu untuk menguji system yang ada berdasarkan spesifikasi model *behaviour* UML [2]. Pengujian perangkat lunak diperlukan skenario pengujian, model *behaviour Unifed Modelling Language* (UML) dapat membangun skenario pengujian. Skenario pengujian terdiri atas pengujian unit, pengujian integrasi dan pengujian sistem [4]. Terdapat 14 diagram di UML digunakan untuk menjelaskan proses pengembangan perangkat lunak yang dikembangkan, ada 2 aspek yaitu aspek prilaku (behaviour aspect) yang bersifat dinamis dan aspek struktural (struktural aspek) yang bersifat statis [5]. Perubahan yang terjadi pada sistem sejalan dengan perubahan waktu dijelaskan pada aspek perilaku, berbeda pada aspek struktural yang menggambarkan struktur elemen-elemen pembentuk sistem yang tidak ada kaitannya dengan waktu, yaitu konsep lebih dari sebuah aplikasi [6].

Pengujian dengan model *behaviour* dengan objek aplikasi pengenalan tulisan tangan, tulisan tangan merupakan pengenalan pola karakter, dikarenakan cara penulisan tiap orang yang berbeda atau bervariasi sehingga menyebabkn bentuk tulisan tangan yang tidak konsisten seperti tulisan tangan yang mengalami rotasi dan ukuran tulisan berbeda menjadi permasalahan [7]. Aplikasi pengenalan tulisan tangan ini telah dilakukan implementasi perangkat lunak. Pengembangan perangkat lunak aplikasi ini menggunakan SDLC. Pada aplikasi pengenalan tulisan tangan terdapat dua form menu yaitu pelatihan dan pengujian. Tiap menu tersebut dilakukan pengujian dengan *use case* diagram.

Use case diagram merupakan diagram yang masuk ke dalam aspek prilaku. Diagram ini menceritakan hubungan antara sistem terhadap pengguna. Hal ini digunakan untuk menguji fungsionalitas dari aplikasi pengenalan tulisan tangan. Aplikasi tersebut tidak terlepas dari beberapa kesalahan pada tiap proses pengembangan perangkat lunak. Untuk mengharapkan fungsionalitas sistem berjalan dengan baik maka harus melalui proses pengujian yang bertujuann untuk mencari beberapa kesalahan yang terjadi pada aplikasi ini. Pengujian ini menggunakan model *behaviour use case*. Sebelum pengujian diimplementasikan, ada beberapa rencana pengujian menggunakan skenario untuk menguji bagian-bagian fungsionalitas aplikasi ini.

2. METODE PENELITIAN

2.1 Pengujian

Konstruksi melibatkan dua bentuk pengujian, yang sering dilakukan oleh pengembang perangkat lunak yang menulis kode yaitu pengujian unit dan pengujian integrasi. Tujuan dari pengujian konstruksi adalah untuk mengurangi kesenjangan antara waktu ketika kesalahan dimasukkan ke dalam kode dan waktu ketika kesalahan terdeteksi, mengurangi biaya yang dikeluarkan untuk memperbaikinya. Dalam beberapa kasus, pengujian dilakukan setelah kode ditulis. Dalam kasus lain, pengujian dapat dibuat sebelum kode ditulis. Dua standar telah diterbitkan pada topik pengujian konstruksi: Standar IEEE 829-1998, Standar IEEE untuk Dokumentasi Uji Perangkat Lunak, dan Standar IEEE 1008-1987, Standar IEEE untuk Pengujian Unit Perangkat Lunak [8].

Pengujian Unit adalah pengujian yang difokuskan pada unit terkecil pada program, pengujian ini berdasarkan informasi dari deksripsi perancangan detil perangkat lunak. Pengujian ini terbagi atas dua yaitu *white box* dan *black box*, pengujian *white box* digunakan untuk mendapatkan kesalahan pada persayaratan fungsional tanpa mengabaikan kerja bagian dalam dari suatu perangkat lunak [9]. Pengujian *white box* dapat digunakan juga untuk mengetahui kompleksitas pada kode program [10]. Pengujian *black box* merupakan pengujian yang yang memperhatikan detail sistem dari funsginya, medeteksi kesalahan pada fungsi antarmuka, kesalahan pada model data serta kesalahan pada alur data yang disimpan pada basis data [11].

Pada penelitian ini, pengujian antarmuka dilakukan dengn model *behaviour* yaitu *use case* diagram. Pengujian ini memastikan bahwa informasi masukan (*input*) dan keluaran (*output*) dari modul-modul perangkat lunak mengalir dengan benar.

2.2 Unifed Modelling Language (UML)

Pengujian system ini dilakukan dengan model behaviour UML, beberaapa diagram UML diantaranya *use case* diagram, *class* diagram, *activity* diagram, *class* diagram, dan *sequence* diagram [12]. Penelitian ini hanya menggunakan *use case* diagram untuk menguji antarmuka pada aplikasi pengenalan tulisan tangan.

2.2.1 Use Case

Use case diagram merupakan diagram yang menjelaskan secara rinci menggunakan dokumen yaitu *use case* skenario. *Use case* scenario ini mendeskripsikan secara tekstual antara aktor dengan sistem, berbeda hal dengan *use case* diagram yang merupakan gambaran representasi dari sistem yang dikembangkan. *Use case* scenario dijelaskan dalam bentuk tekstual yang bergantung pada format kebutuhannya, yaitu informal, singkat, dan lengkap. *Use case* scenario juga menceritakan kondisi awal dan kondisi akhir pada sistem. Pada scenario terdapat bagian-bagian penting yaitu aktor, kondisi awal, kondisi akhir, skenario utama, dan skenario alternatif [13].

Use case diagram digambarkan dalam visualisas hubungan antara aktor dengan sistem. Elemen-elemen pada *use case* diagram yaitu aktor, *use case*, asosiasi, include, extend dan hubungan generalisasi. Aktor merupakan orang atau sistem yang memperoleh manfaat dari dan bersifat eksternal terhadap subjek. Aktor dinotasikan dengan symbol gambar orang-orangan (*stickman*). *Use case* dinotasikan dengan bentuk elips dengan nama kata kerja yang aktif pada bagian dalamnya yang menceritakan aktivitas dari perspektif aktor. Setiap *use case* diperbolehkan berinteraksi dengan sistem dan banyak aktor dapat menjalankan satu *use case*. Extend merupakan hubungan tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan dapat berdiri sendiri tanpa *use case* tambahan dan arah panah mengarah pada *use case* yang ditambahkan. Sedangkan include merupakan tambahan *use case* dimana membuthkan *use case* ini untuk menjalakan fungsinya atau *use case* ini sebagai syarat untuk menjalankan *use case* ini, arah panah mengarah pada *use case* yang dibutuhkan. Hubungan generalisasi merupakan hubungan umum ke khusus, arah panah mengarah pada *use case* yang menjadi umum [14].

2.3 Metode Pengembangan Perangkat Lunak

Metode pengembangan perangkat lunak Aplikasi Pengenalan Tulisan Tangan menggunakan SDLC, Secara umum memiliki 4 fase yaitu *planning* (perencanaan), analisis, desain, dan implementasi. Berikut ini penjelasan tiap fase SDLC yaitu:

A. *Planning* (perencanaan
1. Pengumpulan kebutuhan
   - Bahan-bahan berupa jurnal, buku serta dari internet yang berkaitan dengan penelitian ini. Kebutuhan pada perangkat lunak ini berupa dua buah data tulisan tangan angka yaitu data primer dan data sekunder.
   - Implementasi aplikasi ini menggunakan Netbeans JDK 7.2.
   - Microsoft Word 2010 untuk dokumentasi aplikasi pengenalan tulisan tangan.
   - Perangkat keras yang digunakan pada sistem ini yaitu Prosessor Intel(R) Core (TM) i5-520M CPU @ 2.40 GHz, RAM 4.00 GB, dan *Hard Disk* 500 GB.

B. Analisis
1. Analisis kebutuhan
   Kebutuhan yang telah dikumpulkan dari *user* dan *stakeholder* kemudian di analisis untuk menentukan prioritas implementasinya.
2. Analisis perangkat lunak: perangkat lunak yang digunakan adalah bahasa pemrograman java, IDE Netbeans, serta Sistem Operasi Windows 7 (seven).
3. Analisis perangkat keras: Analisa kebutuhan perangkat keras terhadap sistem seperti kecepatan processor, kapasitas memori utama dan memori sekunder.

C. Desain
1. Rancangan Antarmuka
   Merancang tampilan masukan dan keluaran yang berbasis GUI menggunakan IDE Netbeans 7.2.
2. Rancangan Objek
   Merancang modul-modul program dalam bentuk objek yang digunakan pada saat pengkodean sistem. Rancangan modul dapat berbentuk algoritma, UML dan pseudo-code.

D. Implementasi
1. Construction
   Bahasa pemrograman yang digunakan adalah bahasa Java sesuai sesuai dengan diagram UML untuk melihat hubungan antar kelas pada aplikasi. Rancangan UML yang telah dibuat, desain antarmuka menggunakan Microsoft Visio 2013.
2. Pengujian
   - Mempersiapkan pengujian terhadap aplikasi berupa data masukan.
   - Melakukan implementasi rencana pengujian terhadap aplikasi pengenalan tulisan tangan berupa masukan data selain format .BMP, input cluster, tidak input nama cluster
   - Menguji peringatan pada aplikasi kondisi tidak sesuai dengan keluaran yang diharapkan.

2.3 Rencana Pengujian

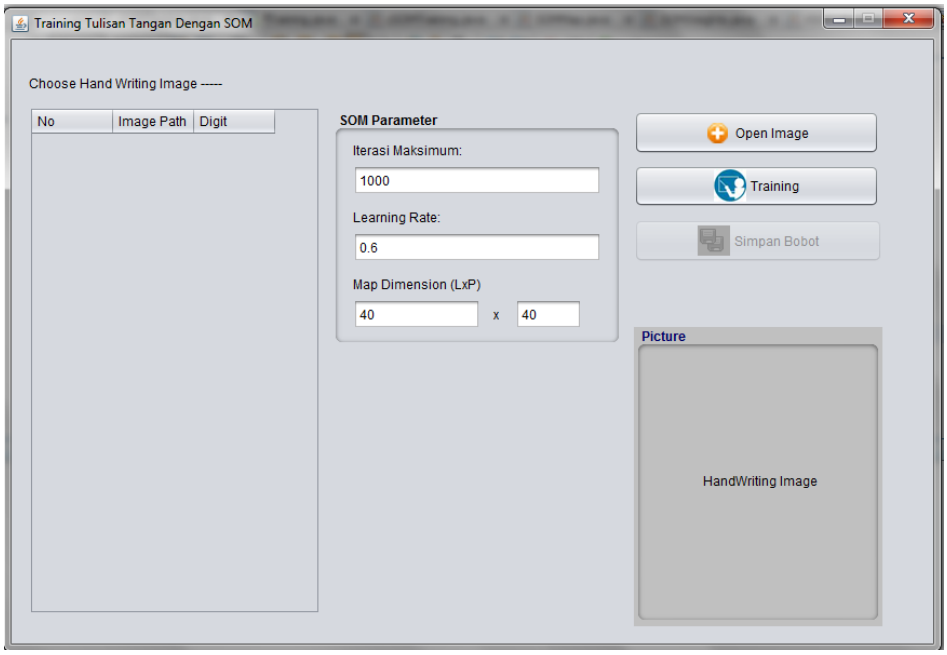
Salah satu teknik pengujian melalui model *behaviour* diagram *use case*, ada 2 pengujian dilakukan yaitu antarmuka dan unit. Pengujian ini bertujuan untuk mengetahui banyaknya kesalahan yang terjadi pada aplikasi pengenalan tulisan tangan. Aplikasi ini terdiri atas 2 menu yaitu pelatihan dan pengujian, aplikasi ini bertujuan untuk mengenali tulisan tangan seseorang dengan mengidentifikasi dari citra tulisan tangan. Citra tulisan tangan ini kemudian dilakukan proses ekstraksi ciri dan selanjutnya dilakukan klaster berdasarkan metode Jaringan Syaraf Tiruan (JST).

Pengujian perangkat lunak ini dilakukan pada setiap proses menu pelatihan dan pengujian melalui scenario *use case* yang terlampir di Tabel 1, Tabel 2, Tabel 3 dan Tabel 4. Setelah skenario *use case*, identifikasi pengujian dilakukan agar fokus pada pengujian tiap menu yang

telah diuraikan dari skenario *use case*. Proses pengujian dimulai dari mengikuti proses skenario *use case* dengan memberikan masukan (*input*) yang sesuai kemudian keluaran (*output*) yang diharapkan serta hasil yang diterima, apakah sesuai dengan keluaran yang diharapkan atau tidak. Jika hasil yang dikeluarkan sesuai dengan yang diharapkan maka kesimpulan diterima dan jika hasil yang dikeluarkan tidak sesuai harapan maka hasilnya ditolak. Berikut gambar antarmuka aplikasi pengenalan tulisan tangan, Gambar 1 menunjukkan halaman utama.

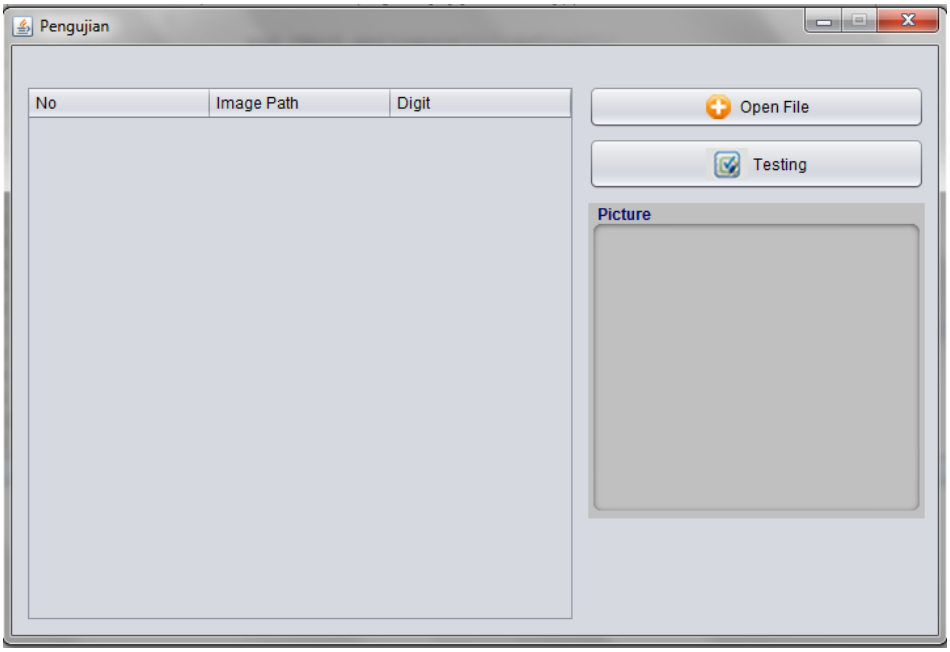
Gambar 1. Antarmuka Menu Utama

Gambar 2 menunjukkan hasil implementasi antarmuka perangkat lunak untuk menu pelatihan.



Gambar 2. Antarmuka Menu Pelatihan.

Gambar 3 menunjukkan hasil implementasi antar muka perangkat lunak untuk menu pengujian.

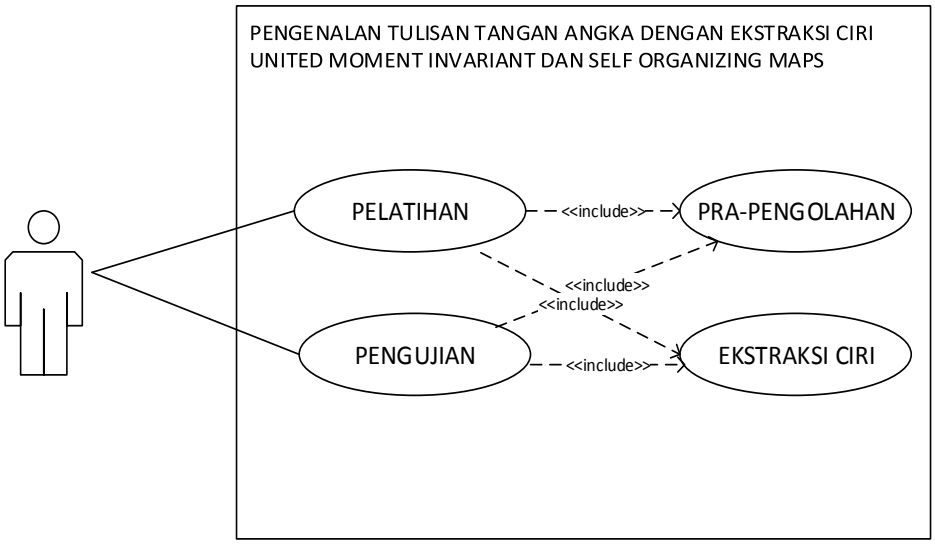


Gambar 3. Antarmuka Menu Pengujian.

3. HASIL DAN PEMBAHASAN

3.1 Model Diagram Use Case

Model diagram *use case* merupakan representasi dari perangkat lunak. Digaram ini digunakan untuk memberi tahu apa yang akan dilakukan system dan digunakan untuk berkomunikasi dengan pengguna. Diagram *use case* dapat dilihat pada Gambar 4.



Gambar 4. Diagram *Use Case*

3.2 Skenario Use Case

Skenario adalah urutan spesifik dari aksi dan interaksi antara aktor dan sistem. Berikut ini adalah skenario dari *use case* yang telah didefinisikan sebagai berikut:

1. Skenario *Use case* Pra-pengolahan Citra Tulisan Tangan

Pada bagian ini menjelaskan tentang skenario *use case* pra-pengolahan dapat dilihat pada Tabel 1.

Tabel 1. Skenario *Use Case* Pra-Pengolahan Citra Tulisan Tangan

No.	001	
Nama *Use case*	Pra-pengolahan.	
Aktor	-	
Tujuan	Mempersiapkan gambar yang akan dilakukan proses pembelajaran atau pengenalan.	
Deskripsi	*Use case* ini digunakan untuk melakukan proses binerisasi dari tulisan tangan yang telah diakuisisi untuk mengurangi *noise* dan menstandarisasi gambar.	
Kondisi Awal	Sistem telah berjalan.	
Kondisi Akhir	Citra tulisan tangan telah normal.	
Skenario Utama		
Aktor		Sistem
Skenario Normal		
		1. Menemukan posisi pixel tulisan tangan
		2. Melakukan operasi perubahan gambar binerisasi berdasarkan data tulisan tangan yang dipilih

2. Skenario *Use case* Ekstraksi Ciri Citra Tulisan Tangan

Pada bagian ini menjelaskan tentang skenario *use case* ekstraksi ciri dapat dilihat pada Tabel 2.

Tabel 2. Skenario *Use Case* Ekstraksi Ciri Citra Tulisan Tangan

No.	002	
Nama *Use case*	Ekstraksi ciri.	
Aktor	-	
Tujuan	Mengekstraksi gambar-gambar yang telah tersimpan dengan nilai ekstraksi ciri	
Deskripsi	*Use case* ini digunakan untuk mengekstraksi gambar-gambar yang telah tersimpan untuk mendapatkan fitur sebagai masukan.	
Kondisi Awal	Tulisan tangan yang telah di normalisasi telah.	
Kondisi Akhir	Data hasil ekstraksi ciri telah didapatkan dan tersimpan	
Skenario Utama		
Aktor		Sistem
Skenario Normal		
		1. Mengambil nilai—nilai piksel
		2. Melakukan perhitungan nilai fitur dengan nilai ekstraksi ciri
		3. Melakukan perhitungan titik pusat pada tulisan tangan
		4. Melakukan perhitungan moment pusat pada tulisan tangan

	5. Menghitung moment pusat ternormalisasi pada tulisan tangan
6. Menampilkan nilai fitur	

3. Skenario *Use case* Pelatihan Citra Tulisan Tangan
   Pada bagian ini menjelaskan tentang skenario *use case* pelatihan dapat dilihat pada Tabel 3.

Tabel 3. Skenario *Use Case* Pelatihan Ciri Citra Tulisan Tangan

No.	003
Nama *Use case*	Melatih Citra Tulisan Tangan.
Aktor	Pengguna.
Tujuan	Mempelajari setiap tulisan tangan yang akan dikenali.
Deskripsi	*Use case* ini digunakan untuk melakukan pelatihan dengan *Self Organizing Maps*.
Kondisi Awal	Sistem telah berjalan.
Kondisi Akhir	Citra tulisan tangan telah dilatih.
Skenario Utama	
Aktor	Sistem
Skenario Normal	
1. Pengguna menekan tombol *Training*	
	2. Menampilkan form *Training*
3. Pengguna menekan tombol *Open Image*	
	4. Menampilkan kotak dialog open
5. Pengguna memilih citra tulisan tangan data primer atau data sekunder	
6. Pengguna memberi nama cluster pada data yang akan di *training*	
	7. Menampilkan *path* dan citra tulisan tangan yang akan di *training*
8. Pengguna menekan tombol *Training*	
	9. Menginisialisasi nilai-nilai masukan
	10. Menginisialisasi bobot-bobot dengan nilai acak
	11. Menghitung keluaran nilai output dengan menggunakan *euclidean distance* untuk mencari nilai terkecil sebagai pemenang
	12. Memperbarui nilai pemenang dan nilai tetangga
	13. Melakukan ulang proses penghitungan *euclidean distance* hingga *epoch* tercapai
	14. Jika nilai iterasi (*epoch*) tercapai maka proses training selesai
	15. Menampilkan alert *Train Finished!*
16. Pengguna menekan tombol	

simpan bobot	
	17. Menampilkan alert Bobot Hasil Training Disimpan!

4. Skenario *Use case Testing* Citra Tulisan Tangan

Pada bagian ini menjelaskan tentang skenario *use case* pengujian citra tulisan tangan dapat dilihat pada Tabel 4.

Tabel 4. Skenario *Use Case Testing* Citra Tulisan Tangan

No.	004
Nama *Use case*	Mengenali citra tulisan tangan.
Aktor	Pengguna.
Tujuan	Mengenali Tulisan Tangan.
Deskripsi	*Use case* ini digunakan untuk mengenali tulisan tangan angka yang diuji pada sistem.
Kondisi Awal	Citra tulisan tangan angka belum dikenali sistem.
Kondisi Akhir	Citra tulisan tangan angka sudah dikenali sistem.
Skenario Utama	
Aktor	Sistem
Skenario Normal	
1. Pengguna menekan tombol *Testing*	
	2. Menampilkan kotak dialog open
3. Pengguna memilih citra tulisan tangan yang akan di*testing*	
4. Pengguna menekan tombol *testing*	
	5. Melakukan pengenalan dan menampilkan citra tulisan tangan angka dikenali

Identifikasi pengujian pada aplikasi dari *usecase* diagram diuraikan pada Tabel 5, Tabel 6, Tabel 7, dan Tabel 8.

Tabel 5. Identifikasi Pengujian *Use Case* Pra-Pengolahan Citra Tulisan Tangan

No.	Identifikasi	Pengujian
1.	P-1-101	Pemrosesan tanpa gambar
2	P-1-102	Pemrosesan dengan gambar
3.	P-1-103	Pemrosesan gambar dengan ekstensi bukan BMP

Tabel 6. Identifikasi Pengujian *Use Case* Ekstraksi Fitur Citra Tulisan Tangan

No.	Identifikasi	Pengujian
1.	P-2-101	Penyimpanan nilai ekstraksi ciri
2.	P-2-102	*Alert* ekstraksi fitur telah

		selesai

Tabel 7 Identifikasi Pengujian *Use Case* Pelatihan Citra Tulisan Tangan

No.	Identifikasi	Pengujian
1.	P-1-101	Tampil antamuka pelatihan
2.	P-1-102	Menekan tombol *Open Image* dan mengisi nama *cluster*
3.	P-1-103	Menekan tombol *training* tanpa memasukan data citra tulisan tangan
4.	P-1-104	Menekan tombol *training* dengan memasukan data citra tulisan tangan
5.	P-1-105	Menekan tombol Simpan Bobot
6.	P-1-106	*Alert* tersimpan

Tabel 8. Identifikasi Pengujian *Use Case Testing* Citra Tulisan Tangan

No.	Identifikasi	Pengujian
1.	P-2-101	Tampil antar muka pengujian
2.	P-2-102	Menekan tombol testing tanpa memasukan data pengujian
3.	P-2-103	Menekan tombol *Open Image*, memilih data citra tulisan tangan yang akan dikenali
4.	P-2-104	Menekan tombol *Testing* untuk dikenali

3.3 Pengujian Sistem

Pengujian sistem bertujuan untuk memastikan bahwa perangkat lunak yang dihasilkan sesuai dengan kebutuhan (*requirement*) yang sebelumnya telah ditentukan. Pengujian ini dilakukan untuk mengantisipasi masalah-masalah antarmuka dan perancangan jalur penanganan kesalahan antar sistem pada perangkat lunak. Pengujian sistem dilakukan dengan mensimulasikan data salah atau data yang berpotensi salah pada antarmuka perangkat lunak. Pengujian tiap *use case* ditampilkan pada Tabel 10, Tabel 11, Tabel 12, dan Tabel 13.

Tabel 10. Pengujian *Use Case* Pra-pengolahan Citra Tulisan Tangan

Identifikasi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Kriteria evaluasi hasil	Hasil yang didapat	Kesimpulan
P-1-101	Menjalankan aplikasi	-	Sistem tidak dapat menangani proses pra pengolahan	-	Sistem tidak dapat menangani proses pra pengolahan	Diterima
P-1-102	Memasukkan objek gambar		Sistem dapat memproses pra-pengolahan	-	Sistem dapat memproses pra-pengolahan	Diterima
P-1-103	Memasukan objek gambar	Angka 0.png	Menampilkan di list pemilihan	-	Menampilkan di list pemilihan	Ditolak

	format berbeda		gambar format selain BMP		gambar format selain BMP	

Tabel 11. Pengujian *Use Case* Ekstraksi Fitur Citra Tulisan Tangan

Identifikasi	**Prosedur pengujian**	**Masukan**	**Keluarkan yang diharapkan**	**Kriteria evaluasi hasil**	**Hasil yang didapat**	**Kesimpulan**
P-2-101	Memproses gambar untuk di ekstraksi fitur	-	Sistem menampilkan nilai ekstraksi ciri	-	Sistem menampilkan nilai ekstraksi ciri	Diterima
P-2-102	Memproses gambar untuk di ekstraksi fitur dengan menekan tombol *Training*	-	Alert dengan isi “esktraksi fitur telah selesai”	-	Alert dengan isi “esktraksi fitur telah selesai”	Diterima

Tabel 12. Pengujian *Use Case* Pelatihan Citra Tulisan Tangan

Identifikasi	**Prosedur pengujian**	**Masukan**	**Keluaran yang diharapkan**	**Kriteria evaluasi hasil**	**Hasil yang didapat**	**Kesimpulan**
P-3-101	Memilih menu *Training*	-	Sistem menampilkan form menu *Training*	-	Form *Training*	Diterima
P-3-102	• Melakukan skenario P-1-101 • Menekan tombol *Open Image* • Mengisi nama *cluster*	Angka 0	Nama cluster dan data citra tulisan tangan masuk ke dalam list	Data yang dimasukkan ditambahkan ke dalam list	Nama cluster dan data citra tulisan tangan masuk ke dalam list.	Diterima
P-4-103	• Melakukan skenario P-1-101 • Menekan tombol training	-	Alert dengan isi “Belum ada data atau data tidak cukup”	Proses pengenalan tulisan tangan menggunakan JST tidak dapat dilakukan	Alert dengan isi “Belum ada data atau data tidak cukup”	Diterima
P-5-104	• Melakukan	Angka 0	Alert dengan	Data yang	Alert	Diterima

	skenario P-1-101 • Melakukan skenario P-1-102 • Menekan tombol *training*		isi "Train Finished"	dimasukkan telah dilatih	dengan isi "Train Finished"	
P-6-105	• Melakukan skenario P-1-101 • Melakukan skenario P-1-102 • Menekan tombol Simpan Bobot	-	Alert dengan isi "Bobot hasil training berhasil disimpan"	Data yang dimasukkan telah disimpan	Alert dengan isi "Bobot hasil training berhasil disimpan"	Diterima
P-7-106	• Melakukan skenario P-1-101 • Melakukan scenario P-1-102 • Melakukan scenario P-1-103 • Menekan tombol OK	-	Alert Tertutup	-	Alert Tertutup	Diterima

Tabel 13. Pengujian *Use Case Testing* Citra Tulisan Tangan

Identifikasi	**Prosedur pengujian**	**Masukan**	**Keluaran yang diharapkan**	**Kriteria evaluasi hasil**	**Hasil yang didapat**	**Kesimpulan**
P-4-101	Memilih menu *Testing*	-	Sistem menampilkan form menu *Testing*	-	Form *Testing*	Diterima
P-4-102	• Melakukan Melakukan skenario P-1-101 • Menekan tombol *Testing*	-	Sistem menampilkan *alert* berisi" Tidak ada yang dapat diproses"	Proses pengenalan tulisan tangan menggunakan JST tidak dapat dilakukan	Sistem menampilkan *alert* berisi" Tidak ada yang dapat diproses"	Diterima

P-4-103	• Melakukan skenario P-1-101 • Menekan *Open Image*	-	Data citra tulisan tangan masuk ke dalam list	Proses pengenalan terhadap citra tulisan tangan dapat di proses	Data citra tulisan tangan masuk ke dalam list	Diterima
P-4-104	• Melakukan skenario P-1-101 • Melakukan skenario P-1-103 • Menekan tombol *Testing*	-	Sistem memberikan respon terhadap pengguna dikenali dan menampilkan hasil pengenalan pada kotak dialog	Proses pengenalan terhadap pengguna dikenali	Sistem memberikan respon terhadap pengguna dikenali dan menampilkan hasil pengenalan pada kotak dialog	Diterima

4. KESIMPULAN

Pengujian ini menggunakan diagram *use case* dari sisi fungsionalitas perangkat lunak. Berdasarkan pengujian system dibagi atas 4 pengujian sesuai dengan diagram *use case*. Masing-masing pengujian memiliki jumlah pengujian sesuai dengan fungsionalitas dari perangkat lunak. Dari 15 pengujian terdapat 14 pengujian yang berhasil, 1 pengujian yang ditolak. Berdasarkan pengujian dengan persentase 93 %. P-1-103 mengalami penolakan karena yang masukan (input) berupa PNG tidak dapat dideteksi sehingga akan dilakukan perbaikan agar gambar yang berekstensi selain BMP dapat diterima.

5. SARAN

Penelitian selanjutnya dapat menerapkan pengujian model behaviour UML selain use case, yaitu model sequence diagram, model activity diagram dapat dilakukan untuk melanjutkan pengujian pada form menu lainnya.

DAFTAR PUSTAKA

[1] "Project Management Institute and IEEE Computer Society, *Software Extension to the PMBOK® Guide Fifth Edition*, Project Management Institute, 2013."

[2] F. B. Waskitho Wibisono, "*Pengujian Perangkat Lunak Dengan Menggunakan Model Behaviour UML," JUTI, Vol. 1, No. JULI 2002, pp. 43–50,* 2002.

[3] G. J. Myers, *The Art of Software Testing , Second Edition*. 2004.

[4] D. Budgen, *Software Design, 2nd ed., Addison-Wesle*. 2003.

[5] OMG, *OMG, 2017. OMG Unified Modeling Language (OMG UML) Ver. 2.5.1. Object Management Group.* 2017.

[6] T. A. Kurniawan, "*Pemodelan Use Case (Uml): Evaluasi Terhadap Beberapa Kesalahan Dalam Praktik*," *J. Teknol. Inf. dan Ilmu Komput.*, Vol. 5 No 1, pp. 77–86, 2018.

[7] G. F. Fitriana, "*Handwriting Digit Recognition Using United Moment Invariant Feature Extraction and Self Organizing Maps*," in *2014 Third ICT International Student Project Conference (ICT-ISPC)*, 2014.

[8] Pierre Bourque and R. E. F. (Dick), *SWEBOK V3.0*. 2004.

[9] A. H, "S*istem Penunjang Keputusan Pemilihan Guru Terbaik dengan Metode TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) Studi Kasus: SDN Bendungan Hilir 01 Pagi Jakarta Pusat*.," *J. Inform. Univ. Pamulang*, pp. 89–96, 2017.

[10] D. Fakhri, M. A., Aknuranda, I., & Pramono, "*Implementasi Sistem Informasi Showroom Mobil (SISMOB) dengan Pemrograman Berbasis Objek (Studi Kasus: UD. Tomaru Oto)*.," *J. Pengemb. Teknol. Inf. dan Ilmu Komputer*, pp. 2967–2974, 2018.

[11] M. Sukamto, R. A., Shalahuddin, *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek.* Bandung: Informatika. 2013.

[12] R. A. Sukamto and M. Shalahuddin, *Kolaborasi Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek.* Bandung: Informatika,. 2015.

[13] L. C, *LARMAN, C., 2005. Applying UML and Patterns. 3rd Edition.* NJ: Prentice Hall. 2005.

[14] S.J. Mellor and M.J. Balcer, *Executable UML: A Foundation for Model-Driven Architecture, 1st ed.*, Addison-Wesley. 2002.