LAPORAN WORKSHOP KUALITAS PERANGKAT LUNAK

SEMESTER 3

PROSEDUR INTEGRATION TESTING – SEBAGAI DASAR PENGUJIAN PERANGKAT LUNAK

OLEH:

FADELIO NAUFZA HAMBORO	(E41232066)
JEMBAR BAHRI HAKIM	(E41232272)
SULTHON DIEKO YUSUF MAULANA	(E41232307)
MUHAMMAD FACHRY PRATAMA AL-HASANI	(E41232308)
BINTANG ERLANGGA	(E41232016)
RANOVE FARREL CAVALERA	(E41232061)
INDRIA IQMA	(E41232194)

GOLONGAN E

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI JEMBER

2024

DAFTAR ISI

ABSTRAK

Makalah ini membahas prosedur integration testing yang diterapkan pada pengembangan sistem absensi sekolah. Integration testing merupakan fase kritis dalam siklus pengembangan perangkat lunak yang bertujuan untuk memverifikasi interaksi antar komponen sistem. Studi ini menguraikan metodologi, teknik, dan best practices dalam melaksanakan integration testing, dengan fokus khusus pada sistem absensi sekolah. Hasil penelitian menunjukkan bahwa pendekatan sistematis dalam integration testing dapat secara signifikan meningkatkan kualitas dan reliabilitas sistem, serta mengidentifikasi potensi masalah sebelum implementasi penuh.

BAB I
PENDAHULUAN

1.1. LATAR BELAKANG

Penelitian ini bertujuan untuk:

1. Menganalisis prosedur integration testing yang efektif untuk sistem absensi sekolah.
2. Mengidentifikasi teknik-teknik terbaik dalam pelaksanaan integration testing.
3. Menyusun kerangka kerja yang dapat diaplikasikan pada proyek pengembangan sistem serupa.

1.2. TUJUAN PENELITIAN

Penelitian ini bertujuan untuk:

1. Menganalisis prosedur integration testing yang efektif untuk sistem absensi sekolah.
2. Mengidentifikasi teknik-teknik terbaik dalam pelaksanaan integration testing.
3. Menyusun kerangka kerja yang dapat diaplikasikan pada proyek pengembangan sistem serupa.

1.3. RUANG LINGKUP

Studi ini akan fokus pada integration testing untuk sistem absensi sekolah, mencakup:

1. Modul pendaftaran siswa
2. Pencatatan kehadiran
3. Pelaporan
4. Integrasi database

BAB II
PEMBAHASAN

2.1. PENGUJIAN PERANGKAT LUNAK

Pengujian perangkat lunak adalah proses yang bertujuan untuk mengidentifikasi kesalahan dalam setiap elemen perangkat lunak, mencatat hasil pengujian, dan mengevaluasi setiap aspek dari sistem yang sedang dikembangkan. Menurut Glen Myers, terdapat beberapa prinsip dasar yang dapat menjelaskan pentingnya pengujian perangkat lunak:

1. Pengujian adalah proses eksekusi program dengan tujuan utama untuk menemukan kesalahan.
2. Sebuah kasus pengujian dianggap baik jika memiliki kemungkinan tinggi untuk menemukan kesalahan.
3. Pengujian yang berhasil adalah pengujian yang dapat menemukan kesalahan.

Dari ketiga prinsip tersebut, dapat disimpulkan bahwa pengujian yang efektif tidak hanya bertujuan untuk menemukan kesalahan, tetapi juga berfokus pada pengembangan data uji yang dapat mendeteksi kesalahan dengan lebih tepat dan cepat.

Secara abstrak, proses pengujian perangkat lunak dapat digambarkan melalui program executable P (yang dapat direpresentasikan dengan disket) dan himpunan data uji T, yang dieksekusi pada komputer. Hasil eksekusi program kemudian dibandingkan dengan persyaratan yang telah ditentukan dan dievaluasi. Fokus utama penguji adalah bagaimana menghasilkan data uji T yang mampu secara efektif menemukan kesalahan (fault) dalam program, memberikan informasi tentang kualitas perangkat lunak, serta memenuhi kriteria atau persyaratan tertentu. Pengujian perangkat lunak mencakup berbagai teknik dan metode yang digunakan, serta strategi pengujian yang diterapkan untuk memastikan bahwa perangkat lunak berfungsi dengan baik dan memenuhi harapan pengguna.

2.2. Metode Pengujian

Metode pengujian adalah teknik yang digunakan untuk menguji perangkat lunak. Metode ini berkaitan dengan perancangan data uji yang akan dieksekusi pada perangkat lunak yang sedang dikembangkan. Idealnya, metode pengujian memiliki mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara menyeluruh (completeness of test) dan memiliki kemungkinan tinggi untuk menemukan kesalahan (high likelihood for uncovering error).

Pengujian perangkat lunak dapat dilakukan dengan dua cara:

1. *Black Box Testing*: Pengujian ini dilakukan dengan mengeksekusi data uji dan memeriksa apakah fungsionalitas perangkat lunak berfungsi dengan baik. Data uji dihasilkan dari spesifikasi perangkat lunak yang menjelaskan fungsionalitasnya.

2. *White Box Testing*: Pada metode ini, pengujian dilakukan dengan menerapkan data uji untuk menguji semua elemen dalam program perangkat lunak, termasuk data internal, loop, logika keputusan, dan jalur. Data uji dihasilkan dengan memahami struktur internal (kode sumber) perangkat lunak.

Kedua metode, yaitu white-box testing dan black-box testing, memiliki berbagai jenis dan kriteria masing-masing. Penjelasan lebih rinci mengenai metode pengujian ini dapat ditemukan pada lampiran. Selain itu, terdapat pendekatan lain dalam pengujian, yaitu pengujian berdasarkan kode sumber dan pengujian berdasarkan spesifikasi.

2.2.1. Pengujian Berdasarkan Kode Sumber

Pengujian berdasarkan kode sumber (*source code-based testing*) adalah metode pengujian di mana data uji dihasilkan dari kode sumber perangkat lunak. Ini merupakan jenis

pengujian yang paling umum digunakan. Dalam proses pembangkitan data uji berdasarkan kode sumber, kriteria pengujian diterapkan pada perangkat lunak untuk menghasilkan persyaratan pengujian. Sebagai contoh, jika kriteria pengujian yang digunakan adalah kriteria pencabangan, maka data uji yang dihasilkan harus mencakup setiap cabang dalam program. Spesifikasi, baik yang bersifat informal maupun formal, berfungsi sebagai dasar untuk penulisan program. Program tersebut kemudian digunakan untuk menghasilkan data uji sesuai dengan kriteria yang telah ditentukan.

Salah satu contoh kriteria pengujian adalah kriteria cakupan (coverage criterion), di mana setiap cabang dalam kode harus diuji setidaknya sekali. Setelah data uji dieksekusi pada program, keluaran aktual dibandingkan dengan keluaran yang diharapkan (expected output). Keluaran yang diharapkan ini diambil dari spesifikasi yang ada. Dengan demikian, pembangkitan data uji berdasarkan kode sumber melibatkan penggunaan spesifikasi untuk menghasilkan kode sumber dan melakukan verifikasi terhadap keluaran program, memastikan bahwa perangkat lunak berfungsi sesuai dengan yang diharapkan.

2.2.2. Pengujian Berdasarkan Spesifikasi

Pengujian berdasarkan spesifikasi (*specification-based testing*) adalah metode pengujian di mana data uji dihasilkan dari spesifikasi perangkat lunak. Spesifikasi ini berfungsi sebagai acuan teknis dalam pengembangan perangkat lunak dan juga sebagai alat untuk pengujian, terutama spesifikasi formal. Spesifikasi formal mendeskripsikan fungsi perangkat lunak dalam format tertentu, yang memungkinkan proses otomasi untuk pembangkitan data pengujian. Selain itu, spesifikasi digunakan sebagai dasar untuk mengecek keluaran, guna memastikan bahwa perangkat lunak memenuhi persyaratan yang diharapkan. Dalam *specification-based testing*, spesifikasi tidak hanya digunakan untuk merancang program, tetapi juga untuk menghasilkan data uji dalam bentuk yang diformalkan.

Pembangkitan data uji dilakukan secara otomatis menggunakan spesifikasi formal sebagai masukan. Karena spesifikasi formal memiliki penjelasan yang lengkap, konsisten, dan tidak ambigu, diharapkan hasil pembangkitan data uji dapat mencakup semua kemungkinan

kesalahan dan memenuhi kelengkapan spesifikasi. Data uji tersebut kemudian dieksekusi pada perangkat lunak, dengan harapan perangkat lunak yang dihasilkan bersifat andal.

Strategi pengujian perangkat lunak merupakan proses integrasi metode perancangan kasus uji ke dalam urutan langkah-langkah pengujian. Strategi ini dapat dianggap sebagai urutan langkah yang mirip dengan pengembangan perangkat lunak. Secara berurutan, strategi pengujian perangkat lunak mencakup pengujian unit (*unit testing*), pengujian integrasi (*integration testing*), dan pengujian sistem (*system testing*).

2.2.3. Pengujian Unit

Pengujian unit (*Unit Testing*) adalah metode pengujian yang fokus pada unit terkecil dari program, yaitu modul. Pengujian ini didasarkan pada informasi dari deskripsi perancangan detail perangkat lunak. Umumnya, pengujian ini dilakukan dengan pendekatan white-box dan source code-based testing, dengan fokus pada pengecekan jalur khusus dalam struktur kendali modul untuk memastikan cakupan yang lengkap dan deteksi kesalahan yang maksimal. Hal-hal yang diuji dalam pengujian unit meliputi:

1. Antarmuka (*Interface*):
   - Memastikan bahwa informasi yang masuk dan keluar dari modul yang diuji mengalir dengan benar.
   - Mencari kesalahan dalam penggunaan parameter modul, baik jumlah, tipe data, urutan parameter, dan penggunaan variabel global yang digunakan lintas modul.
   - Jika modul berinteraksi dengan peralatan I/O eksternal, pengujian tambahan dilakukan untuk memeriksa kesalahan pada penggunaan atribut, perintah open/close, kondisi end-of-file pada file, penanganan kesalahan I/O, dan informasi keluaran.

2. Struktur Data Lokal:
   - Memastikan integritas data yang disimpan selama eksekusi modul.
   - Mencari kesalahan dalam penggunaan tipe data, inisialisasi atau nilai default, nama variabel, serta menangani masalah seperti underflow, overflow, dan addressing exception.

3. Kondisi Batas:
   - Memastikan bahwa modul berfungsi dengan benar pada batas-batas pemrosesan yang ditentukan.

- Mencari kesalahan pada penggunaan struktur data di batas-batas deklarasi, perintah pengulangan pada batas pengulangannya, serta penggunaan nilai maksimum dan minimum yang diperbolehkan.

4. Jalur-jalur Bebas (*Independent Paths*):
- Memastikan bahwa semua kemungkinan jalur kendali telah dieksekusi setidaknya satu kali.
- Mencari kesalahan dalam penghitungan, pembandingan, dan alur kendali.

5. Jalur Penanganan Kesalahan:
- Memastikan bahwa kondisi kesalahan dapat diantisipasi dan ditangani dengan baik tanpa menghentikan proses (antibugging).
- Mencari kesalahan dalam respons kesalahan, pemberitahuan kesalahan, penanganan kesalahan, kondisi kesalahan yang memerlukan intervensi sistem sebelum penanganan, serta respons kesalahan yang tidak memberikan informasi yang cukup untuk menemukan sumber kesalahan.

Dengan demikian, pengujian unit berperan penting dalam memastikan bahwa setiap modul perangkat lunak berfungsi dengan baik dan sesuai dengan spesifikasi yang telah ditentukan.

2.3. PENGUJIAN DARI SPESIFIKASI MODEL BEHAVIOUR UML

2.3.1. Model Diagram Sequence

Diagram Sequence digunakan untuk mengambarkan interaksi antar obyek dalam berkomunikasi saling mengirim message disusun berdasarkan urutan waktu. Diagram sequence ini bersama-sama dengan diagram collaboration biasa disebut juga dengan diagram interaksi, sebab keduanya berfungsi menggambarkan interaksi antar objek.

Diagram sequence disusun , bagian mendatar atas sebagai tempat objek-objek yang berinteraksi dan bagian vertikal menggambarkan urutan waktu objek-objek saling mengirim dan menerima messages. Komponen yang membangun diagram sequence adalah :

1. Object Objek digambarkan didalam kotak. Isi kotak berupa nama objek kemudian diikuti dengan type objek. Tipe objek disesuaikan dengan tipe class yang telah didefinisikan di dalm diagram class.
2. Timeline/ Life line Timeline digambarkan berupa garis putus-putus menempel pada objek secara vertikal dari atas ke bawah.
3. Messages Messages berupa operasi yang ada di dalam obek disertai dengan parameter bjek yang dikirimkan kepada objek lain.
4. Destroy Destroy menggambarkan bahwa pada waktu yang telah ditentukan objek yang didefinisikan sudah dihilangkan / di destroy keberadaannya di memory komputer.

Aktivasi Aktivasi digambarkan dengan persegi panjang. Aktivasi ini melambangkan kapan objek tersebut aktiv berperan di dalam sistem sampai objek tersebut pasif kembali.

Model diagram sequence digunakan untuk :

1. Memodelkan aliran kontrol / pesan-pesan yang dikirim dan diterima oleh masing-masing object dalam kurun waktu tertentu.

2. Memodelkan urutan pemrosesan

3. Memodelkan method yang menangani pesan pada masing-masing object

Memodelkan pesan/ messages apa saja yang digunakan untuk berinteraksi oleh masing-masing object Dewasa ini kebutuhan perangkat lunak dengan tingkat kompleksitas tinggi dan kritis cukup meningkat. Perangkat lunak tersebut biasanya mempunyai deskripsi yang jelas, lengkap dan bahkan dalam bentuk spesifikasi formal. Pada kenyataannya deskripsi yang jelas tersebut kebanyakan hanya ada pada tingkat (level) unit, sedang pada tingkat sistem deskripsi hanya dilakukan secara informal.

UML merupakan notasi pemodelan yang cukup baik untuk menjelaskan perangkat lunak pada semua tingkat pengembangan perangkat lunak. Dan ini merupakan peluang yang dapat digunakan untuk penentuan data uji. Walaupun UML tidak terlalu formal tetapi deskripsi pada UML cukup teliti dan lengkap untuk menjelaskan perangkat lunak. Statechart UML dipilih sebagai awal pembangkitan data uji berdasar spesifikasi karena statechart menjelaskan kelakuan (behavior) sistem. Pengujian kelakuan perangkat lunak sebenarnya dapat dilakukan dengan menguji setiap metode (method) dari suatu obyek, karena kelakuan suatu obyek diimplementasikan dari metodenya. Tetapi pengujian setiap metode obyek hanya menguji sebagian kelakuan obyek bukan keseluruhan dari sistem.

2.3.2. Model Diagram Collaboration

Collaboration bermakna kerjasama antar bagian untuk melaksanakan fungsi tertentu yang dibebankan kepada masing-masing bagian objek. Diagram Collaboration digunakan untuk mengambarkan susunan organisasi antar objek dalam berinteraksi dan bekerjasama untuk melaksanakan fungsi yang didefinisikan di dalam sistem software. Dalam diagram collaboration ini digambarkan objek apa saja yang bekerja sama dan message apa yang dikirim dan diterima untuk berkomunikasi dalam rangka melaksanakan fungsi yang didefinisikan

Kegunaan Diagram Collaboration

Diagram Collaboration digunakan untuk :

1. Memodelkan urutan interaksi antar obyek dalam bentuk susunan organisasi
2. Memodelkan aliran kontrol / pesan-pesan yang dikirim dan diterima oleh masing-masing object.
3. Memodelkan urutan pemrosesan

4. Memodelkan method yang menangani pesan pada masing-masing object Memodelkan pesan/ messages apa saja yang digunakan untuk berinteraksi oleh masing-masing object

2.3.3. Model Diagram Activity

Diagram activity digunakan untuk mengambarkan urutan kerja masing-masing obyek dalam menyelesaiakan permasalahan tertentu sesuai dengan fungsi kerja masing-masing obyek. Diagram activity pada hakekatnya adalah flowchart yang menunjukkan aliran aktivitas ke aktivitas dalam memenuhi fungsi layanan yang didefinisikan didalam use case. Diagram activity digunakan untuk :

1. Memodelkan aliran kerja object / workflow
2. Memodelkan opersional proses

2.3.4. Model Diagram Statechart

Statechart UML adalah diagram yang menggambarkan kelakuan sistem secara keseluruhan. Karena menggambarkan kelakuan sistem secara keseluruhan maka pembangkitan data uji berdasar statechart (state machine) dianggap menguji keseluruhan sistem. Statechart UML dibuat berdasar State Machine yang digunakan oleh David Harel .

Mesin Status (State machine) adalah mesin yang menggambarkan atau memodelkan kelakuan dari obyek individual. Mesin Status adalah kelakuan yang menggambarkan urutan status dari suatu obyek yang hidup pada suatu waktu (lifetime) karena respon dari event.

Mesin Status dan komponen-komponen pendukungnya secara konseptual dapat dijelaskan sebagai berikut :

1. Mesin Status : suatu behavior (kelakuan) yang menggambarkan urutan status (state) dari suatu obyek yang hidup pada waktu hidup (lifetime) nya karena respon dari event.

2. Status (state) : kondisi atau situasi pada saat suatu obyek hidup yang memenuhi suatu kondisi, melakukan suatu aktivitas, atau menunggu suatu event. Pada statechart UML status digambarkan dengan kotak bersudut tumpul.

3. Event : spesifikasi suatu kejadian (occurrence) yang mempunyai alokasi ruang dan waktu. Didalam kontek Mesin Status, event adalah kejadian (occurrence) dari suatu pemicu (stimulus) yang memicu suatu transisi status. Pada statechart UML event digambarkan (dituliskan) sebagai teks yang menyertai transisi.

4. Transisi : adalah hubungan antara dua status yang menunjukkan bahwa obyek pada pada saat status pertama akan melakukan suatu aksi tertentu dan masuk ke status kedua jika suatu event terjadi dan suatu kondisi tertentu dipenuhi. Pada statechart UML transisi digambarkan dengan anak panah berarah, dengan asal anak panah adalah status sumber (asal) dan anak panah tujuan adalah status target (tujuan).

2.4. Pengujian Integrasi

Pengujian Integrasi (*Integration Testing*) adalah pengujian yang difokuskan pada gabungan unit-unit atau modul-modul yang membentuk kesatuan fungsional. Pengujian ini didasarkan pada informasi dari deskripsi perancangan awal perangkat lunak. Pengujian ini dilakukan untuk menemukan kesalahan antarmuka antar modul. Pengujian ini umumnya dilakukan oleh pengembang sendiri atau dilakukan antar pengembang. Pada umumnya pengujian ini dilakukan secara white-box dan blackbox.

Pada pengujian integrasi dikenal istilah integrasi non-incremental, dan integrasi incremental. Integrasi non-incremental adalah proses integrasi yang menggunakan cara penggabungan langsung modul-modul yang terlibat. Program kemudian diuji secara keseluruhan. Hal ini bisa menyebabkan terjadinya kesalahan yang kompleks karena perkembangan program yang besar. Sedang integrasi incremental adalah integrasi yang dilakukan secara bertahap. Pengujian juga dilakukan per segmen sehingga kesalahan dapat dengan mudah diisolasi dan diperbaiki.

2.5. Pengujian Sistem

Pengujian sistem adalah proses yang dilakukan untuk menguji sistem komputer secara keseluruhan. Pengujian ini umumnya dilakukan oleh pengembang perangkat lunak bersama tim lain, karena berkaitan dengan berbagai elemen dalam perangkat lunak. Tujuan utama

pengujian sistem adalah untuk mengantisipasi masalah yang mungkin muncul pada antarmuka dan perancangan jalur penanganan kesalahan antar sistem.

Pengujian ini melibatkan simulasi data yang salah atau berpotensi salah pada antarmuka perangkat lunak. Proses pengujian sistem memiliki karakteristik yang berbeda-beda, dan fokus utamanya adalah untuk memvalidasi sistem secara keseluruhan, memastikan bahwa integrasi dan kinerja masing-masing elemen sistem sesuai dengan kebutuhan, serta mengecek apakah perangkat lunak memenuhi harapan pemakai.

Pengujian sistem dilakukan dengan pendekatan black-box dan specification-based testing. Rencana pengujian (test plan) akan mendefinisikan prosedur pengujian dan menentukan data uji yang diperlukan. Pengujian sistem dirancang untuk memastikan bahwa:

- Semua kebutuhan fungsional perangkat lunak terpenuhi
- Kinerja perangkat lunak sesuai dengan yang diharapkan
- Dokumentasi akurat
- Kebutuhan lain, seperti transportability, compatibility, error recovery, dan maintainability, terpenuhi

Untuk perangkat lunak yang dibuat khusus, pengujian ini dikenal sebagai acceptance test, yang dilakukan oleh pengguna untuk memvalidasi spesifikasi kebutuhan. Pengujian ini bisa dilakukan secara informal atau sistematis selama periode tertentu, sehingga kesalahan kumulatif dapat terdeteksi.

Sementara itu, pengujian sistem untuk produk perangkat lunak dikenal sebagai Alpha Testing dan Beta Testing. Alpha Testing dilakukan oleh pengguna di lingkungan pengembang yang terkendali, sedangkan Beta Testing dilakukan di lingkungan pengguna, di mana pengembang tidak lagi dapat mengendalikan kondisi. Selain itu, pengujian sistem juga mencakup:

1. Pengujian Recovery (*Recovery Testing*): Menguji kemampuan perangkat lunak untuk pulih dari kegagalan yang dipaksa terjadi dalam berbagai cara.
2. Pengujian Keamanan (*Security Testing*): Menguji sistem perlindungan perangkat lunak dengan mencoba penetrasi melalui serangkaian proses ilegal.

3. *Stress Testing*: Memaksa perangkat lunak beroperasi di luar batas normal, baik dari segi kuantitas, frekuensi, maupun volume data.

Dengan pendekatan ini, pengujian sistem bertujuan untuk memastikan perangkat lunak berfungsi dengan baik dan memenuhi semua kriteria yang telah ditetapkan.

2.6. PERANCANGAN PERANGKAT LUNAK

Pada perancangan pengujian model behaviour UML yang digunakan adalah Collaboration Diagram, Sequence Diagram, Activity Diagram dan Statechart Diagram. Perancangan perangkat lunak dilakukan dengan tahapan sebagai berikut :

1. Perancangan format masukan dan keluaran
2. Perancangan aspek statis perangkat lunak, yaitu penentuan kelas dan hubungan antar kelasnya. Penentuan kelas meliputi data anggota dan fungsi anggota. Hubungan antar kelas digambarkan dengan class diagram.
3. Perancangan aspek dinamis, yaitu urutan penghidupan obyek, pemanggilan fungsi anggota dan pemusnahan obyek dilakukan dengan collaboration dan sequence diagram.

Pada perancangan aspek statis, penentuan struktur data dari anggota data kelas dilakukan dengan mencari representasi paling optimal untuk memenuhi spesifikasi kebutuhan perangkat lunak. Untuk representasi internal dari kelas banyak digunakan multi-list. Tumpukan digunakan pada algoritma untuk mengubah ekpresi predikat dari linked-list ke bentuk struktur pohon. Struktur tumpukan menggunakan tipe tumpukan dari pointer yang menunjuk elemen dari pohon.

Perancangan aspek dinamis pada perangkat lunak digambarkan dengan collaboration dan sequence diagram. Diagram ini digunakan untuk menggambarkan organisasi obyek dan interaksi antar obyek dan menggambarkan urutan-urutan message serta aliran kendali. Perangkat lunak mempunyai tiga sub program yang bisa dieksekusi. Program tersebut adalah :

Program pembangkit file spesifikasi, Program pembangkit data uji dengan kriteria Full Predicate Coverage dan Program pembangkit data uji dengan kriteria Transition Pair. Masing-masing program dijelaskan dalam sub bab berikutnya. Program ini adalah program yang bersifat interaktif yang mempunyai beberapa pertanyaan yang harus diisikan oleh pengguna.

Pertanyaan tersebut adalah :

1. Apa deskripsi dari state machine file spesifikasi yang akan dibangkitkan ? Deskripsi ini diisi string yang boleh tidak diisi.
2. Berapa jumlah status (state) yang dipunyai state machine ? Jumlah ini harus diisikan nilai digit lebih besar dari 0 dan kurang dari 31. Status yang bisa ditangani oleh perangkat lunak terbatas untuk 30 status.
3. Status apa saja yang ada state machine ? Semua status yang ada pada state machine harus dituliskan disini. Status ke-1 diset sebagai status awal (initial state). boleh ada status yang tidak mempunyai nama Kesulitan yang dihadapi pada saat perancangan file eksternal untuk keluaran yang dalam hal ini adalah spesifikasi uji adalah penentuan apakah file eksternal ini berisi spesifikasi uji yang memenuhi semua kriteria pembangkitan data uji atau berisi spesifikasi uji untuk satu kriteria saja.

Pemisahan atau tidaknya file eksternal spesifikasi uji untuk masing-masing kriteria ini berhubungan dengan pekerjaan selanjutnya yaitu otomatisasi pembangkitan test script. Ada dua alternatif untuk memecahkan masalah ini :

1. Penggunaan sebuah file yang mencakup semua kriteria pengujian

2. Penggunaan beberapa file sesuai dengan jumlah kriteria yang diinginkan. Pemisahan modul dan file eksternal keluaran akan memudahkan pembuatan laporan dan analisis hasil pengujian.

BAB III
KESIMPULAN

Pengujian perangkat lunak merupakan proses penting untuk memastikan kualitas dan fungsionalitas sistem. Prinsip dasar pengujian menekankan penemuan kesalahan dan pengembangan data uji yang efektif. Berbagai metode, seperti black-box dan white-box testing, menyediakan pendekatan berbeda dalam perancangan pengujian. Pengujian berbasis

spesifikasi, terutama menggunakan model UML, membantu menghasilkan data uji yang terarah dan relevan melalui diagram seperti Sequence, Collaboration, Activity, dan Statechart. Strategi pengujian meliputi tahapan dari pengujian unit hingga sistem, memastikan semua aspek perangkat lunak diuji dengan baik. Meskipun pembangkitan data uji berbasis statechart terbatas pada transisi tertentu, ini menawarkan peluang untuk pengembangan lebih lanjut dengan spesifikasi lain. Secara keseluruhan, meskipun terdapat batasan, metodologi ini dapat dikembangkan untuk menghadapi tantangan perangkat lunak yang semakin kompleks dan kritis.

DAFTAR PUSTAKA

Bahrami Ali, Object Oriented System Development,, Singapore, McGraw-Hill International Edition Singapore, 1999.

Booch, G, Rumbaugh J., Jacobson I., The Unified Modelling anguage User Guide. Massachusetts., Addison Wesley Longman Inc, 1999.

Larman C., Appllying UML and Pattern An Introduction to Object Oriented Analysis and Design, New Jersey, Prentice Hall PTR, 1998.

Offut, A Jefferson. dan Abdulrazik, Aynur. Generating test cases from UML specifications. In Proceeding of the second IEEE International Conference on Unified Modeling Language (UML99), pages 416-429, Fort Collins, CO, IEEE Computer Society Press, October 1999.

Offut, A Jefferson. dan Liu, Shaoying. Generating test data from SOFL specifications. The Journal of Systems and Software, 1999.

Offut, A.Jefferson. Xiong, Yiwei. dan Liu, Shaoying. Criteria for Generating Specificationbased tests. http://www.isse.gmu.edu

Perry, William. Effective Methods for Software Testing. John Wiley & Sons, Inc., 1995.
Pressman, Roger S., Software Engineering – A Practitioner's Approach, New York, McGraw-Hill Inc., 1997

Valacich J.S. , George J.F. , Hoffer J.A., Essentials of System Analysis and Design, New Jersey, Prentice Hall, 2001.

Burnstein, I. (2006). Practical Software Testing: A Process-Oriented Approach. Springer Science & Business Media.

Myers, G. J., Sandler, C., & Badgett, T. (2011). The Art of Software Testing (3rd ed.). John Wiley & Sons.

Kaner, C., Bach, J., & Pettichord, B. (2008). Lessons Learned in Software Testing: A Context-Driven Approach. John Wiley & Sons.

Crispin, L., & Gregory, J. (2009). Agile Testing: A Practical Guide for Testers and Agile Teams. Addison-Wesley Professional.

Black, R. (2011). Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional. John Wiley & Sons.

Dustin, E., Rashka, J., & Paul, J. (2008). Automated Software Testing: Introduction, Management, and Performance. Addison-Wesley Professional.

Naik, K., & Tripathy, P. (2011). Software Testing and Quality Assurance: Theory and Practice. John Wiley & Sons.