

**PENGUJIAN PERANGKAT LUNAK DENGAN
MENGUNAKAN BEHAVIOR UML (STUDI KASUS :
APLIKASI PEMBELAJARAN DARING TERBUKA
TERPADU UB)**

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

Drajad Muhammadi Latief

NIM: 115061000111027



**PROGRAM STUDI SISTEM INFORMASI
JURUSAN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017**

PENGESAHAN

PENGUJIAN PERANGKAT LUNAK DENGAN MENGGUNAKAN BEHAVIOR UML
(STUDI KASUS : APLIKASI PEMBELAJARAN DARING TERBUKA TERPADU UB)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Drajad Muhammadi Latief
NIM: 115061000111027

Skripsi ini telah diuji dan dinyatakan lulus pada
Januari 2017

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Dosen Pembimbing II

Aditya Rachmadi, S.ST., M.TI
NIK. 86042116110426

Denny Sagita R., S.Kom, M.Kom
NIP: 198511242015041001

Mengetahui
Ketua Jurusan Sistem Informasi

Herman Tolle, Dr. Eng., S.T, M.T
NIP: 197107271996031001

PERNYATAAN ORISINALITAS

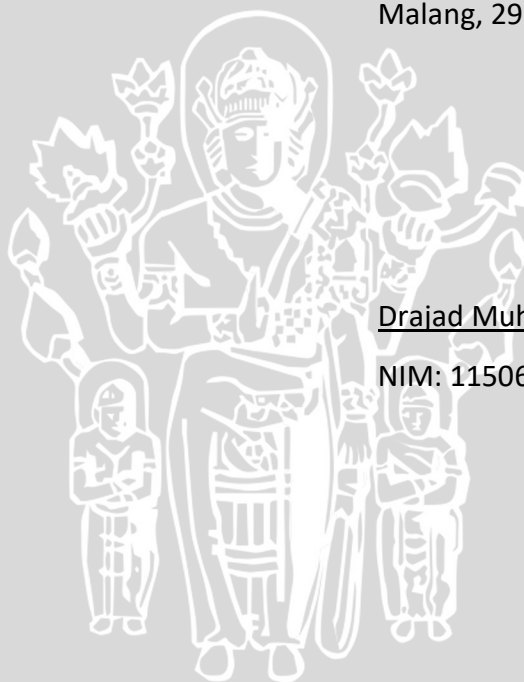
Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 29 Januari 2017

Drajad Muhammadi Latief

NIM: 115061000111027



Kata Pengantar

Puji syukur penulis panjatkan kehadiran Allah SWT, karena berkat limpahan dan rahmat-Nya penulis mampu menyelesaikan skripsi dengan judul “Pengujian Perangkat Lunak Dengan Menggunakan Behavior UML (Studi Kasus : Aplikasi Pembelajaran Daring Terbuka Terpadu UB)”. Skripsi ini merupakan salah satu syarat untuk memenuhi persyaratan akademis guna menyelesaikan studi di Fakultas Ilmu Komputer Universitas Brawijaya.

Selama proses pembuatan skripsi ini, penulis menemui banyak rintangan. Untuk itu, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah membantu dalam menyelesaikan halangan tersebut dan mendukung penyelesaian skripsi ini, terutama kepada :

1. Aditya Rachmadi, S.ST., M.TI selaku dosen pembimbing I dan Denny Sagita R., S.Kom, M.Kom selaku dosen pembimbing II atas arahan serta bimbingannya dalam proses penyusunan skripsi ini.
2. Rekyan Regasasari Mardi Putri, S.T., M.T., selaku dosen pembimbing akademik, atas bimbingan dan motivasi yang diberikan kepada penulis selama masa perkuliahan.
3. Segenap Bapak dan Ibu dosen Fakultas Ilmu Komputer yang telah mendidik, mengajarkan, dan memberikan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Brawijaya.
4. Segenap staff dan karyawan Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi.
5. Kedua orang tua penulis, Drs. Affandi dan Ir. Kundarsih, serta adik penulis, Himawwan Bayu Maulana, dan seluruh anggota keluarga besar penulis, atas doa-doa yang tak putus, dukungan, kesabaran, kepercayaan, dan kasih sayang yang diberikan kepada penulis.
6. Sahabat-sahabatku, Kontrakan Dwiga A3-02, Kelompok belajar berlima yang selalu mendukung, menjadi pendengar cerita setia penulis, memberikan nasihat, dan menghibur penulis.
7. Keluarga besar Sistem Informasi Universitas Brawijaya 2011, yang selama 4 tahun ini sudah menjadi keluarga kedua penulis, yang sudah menerima penulis apa adanya. Tempat belajar, berbagi pengalaman, bercanda untuk penulis. SI JAYA !

Serta semua pihak yang telah membantu terselesaikannya skripsi ini, yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Karena itu penulis mengharapkan kritik dan saran yang membangun dari pembaca. Semoga skripsi ini dapat bermanfaat bagi pembaca.

Malang, 29 Januari 2017

Penulis
drajadmuhammadilatief@gmail.com

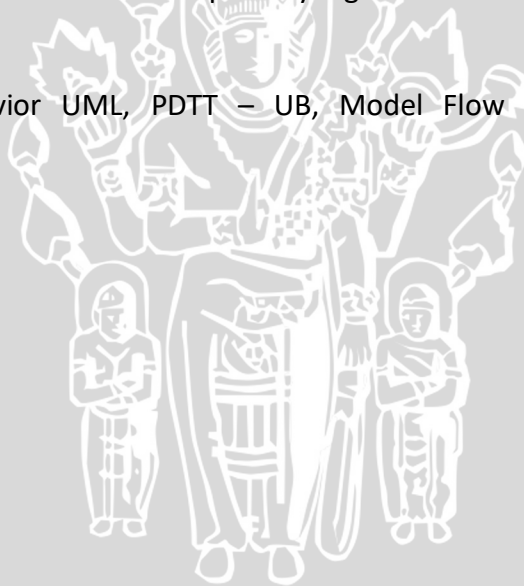


Abstrak

Penelitian ini bertujuan untuk melakukan pengujian aplikasi PDTT – UB menggunakan metode Behavior UML yang nantinya aplikasi tersebut akan digunakan sebagai media pembelajaran di Universitas Brawijaya Malang. Untuk mendapatkan *Test case*, metode yang digunakan yaitu behavior UML, dimana UML yang digunakan adalah *Activity diagram* dan *Sequence diagram*. Dari diagram UML tersebut akan dibuat Model Flow Graph (MFG) yang didapat dari hasil analisis *Activity diagram* dan *Sequence diagram*. Setelah didapatkan MFG dari *Activity diagram* dan *Sequence diagram*, maka MFG akan dibuat jalur-jalur yang akan di analisis menjadi *Test case* menggunakan metode *Test case Generation* untuk pengujian Aplikasi PDTT-UB.

Dengan menggunakan behavior UML, akan muncul beberapa *Test case* yang tidak bisa didapat apabila menggunakan pengujian blackbox maupun whitebox. Karena pengujian Behavior UML sumber datanya adalah fase perancangan yaitu *Activity diagram* dan *Sequence diagram* sehingga *Test case* yang didapatkan akan sesuai dengan fase perencanaan dan kebutuhan aplikasi. Dengan pengujian behavior UML ini, maka rekomendasi yang diberikan kepada developer aplikasi PDTT-UB tidak keluar dari kebutuhan aplikasi yang dibuat di fase perancangan.

Kata Kunci : Behavior UML, PDTT – UB, Model Flow Graph, *Test case Generation*



Abstrac

This research point to test PDTT-UB application using behavior UML that the application will be used as a method of learning in Brawijaya University. To get a *Test case*, the method used is behavior UML which is used *Activity diagram* and *Sequence diagram*. From the UML Diagram will be made Model Flow Graph (MFG) is obtained from the analysis of the *Activity diagram* and *Sequence diagram*. Having obtained MFG of *Activity diagram* and *Sequence diagram*, the MFG will be made pathways to be a *Test case* using *Test case Generation* methods for PDTT-UB.

By using Behavior UML, there will be some *Test cases* that can't be obtained when using others methods of testing such as blackbox testing because in behavior UML, the data source will get from design phase, namely *Activity diagram* and *Sequence diagram*. So that *Test cases* will be obtained in accordance with the planning phase and application needs. By testing Behahavior UML, then the results and recommendations goven to the developer of PDTT-UB doesn't come out the needs of applications made in design phase

Keywords : Behavior UML, PDTT-UB, Model Flow Graph, *Test case Generation*

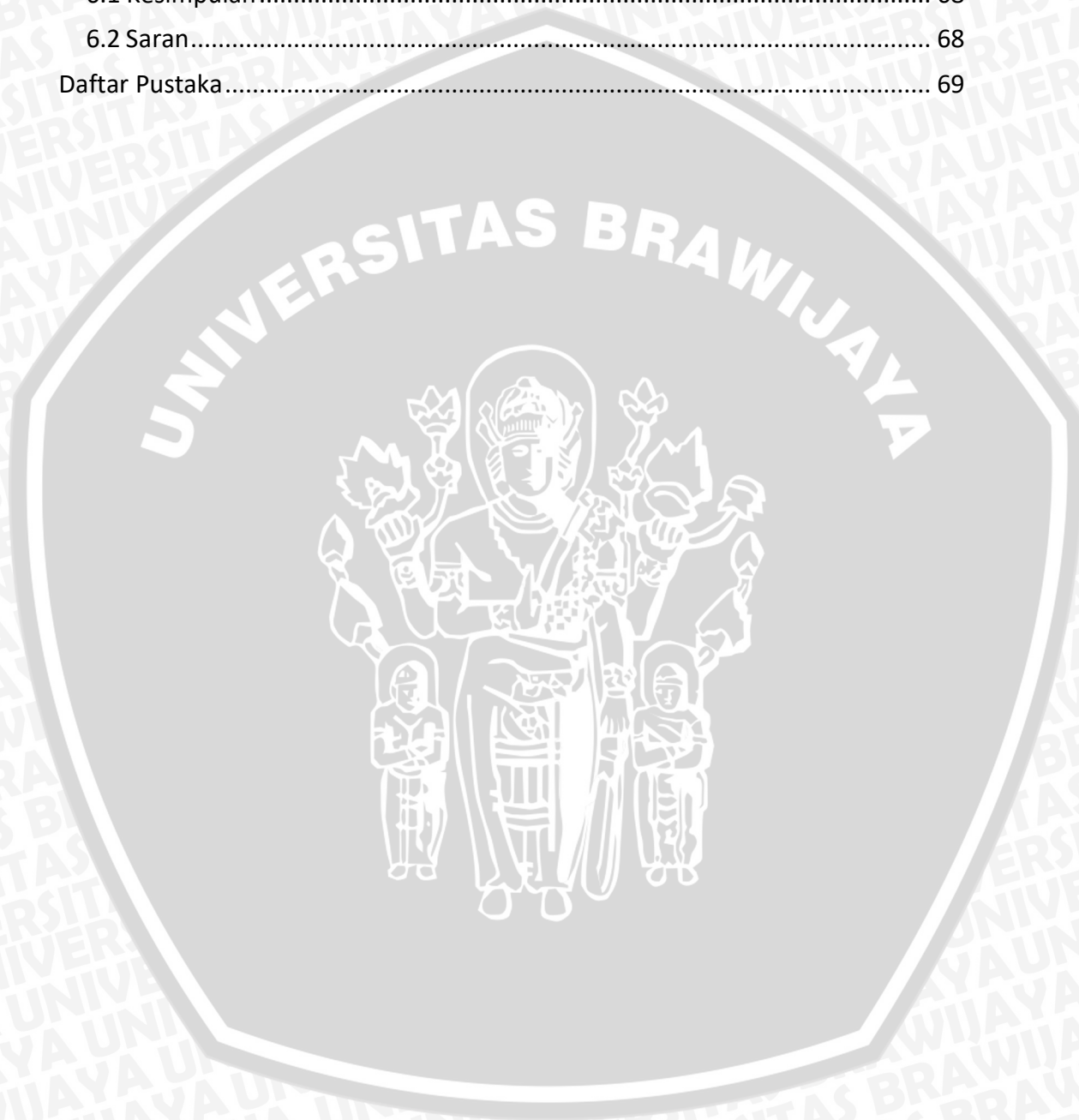


Daftar Isi

PENGESAHAN	ii
PERNYATAAN ORISINALITAS	iii
Kata Pengantar	iv
Abstrak	vi
Abstrac	vii
Daftar Isi	viii
Daftar Tabel	xi
Daftar Gambar	xii
BAB 1 Pendahuluan	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Manfaat	2
1.5 Batasan Masalah	3
1.6 Sistematika Pembahasan	3
BAB 2 Landasan Kepustakaan	4
2.1 Penelitian Terkait	4
2.2 Aplikasi Pembelajaran Daring Terbuka Terpadu UB (PDTT-UB)	4
2.3 Pengujian	5
2.1.1 Definisi	5
2.1.2 Tujuan	5
2.1.3 Klasifikasi Pengujian	5
2.2 Sistem	6
2.3 Informasi	7
2.4 Sistem Informasi	7
2.5 <i>Unified Modeling Language (UML)</i>	8
2.5.1 <i>Sequence diagram</i>	8
2.5.2 <i>Activity diagram</i>	9
2.6 <i>Test case</i>	10
2.6.1 <i>Test Procedure</i>	10
2.7 Pengujian Software Dengan Behavior UML	11

2.8 Model Flow Graph	13
2.9 Error, Fault, dan Failure	13
BAB 3 Metodologi Penelitian	15
3.1 Definisi Masalah	15
3.2 Studi Literatur	16
3.3 Pembuatan Test case Menggunakan Behavior UML	16
3.4 Pelaksanaan Pengujian Berdasarkan Test case dan Test Procedure	18
3.5 Analisis dan Kesimpulan	18
BAB 4 Perancangan dan Analisis	19
4.1 Sequence diagram	19
4.1.1 Sequence diagram Manajemen Folder	19
4.1.2 Sequence diagram Delete Folder	20
4.1.3 Sequence diagram Manajemen File	21
4.1.4 Sequence diagram Delete File	22
4.1.5 Sequence diagram Tambah Modul	23
4.1.6 Sequence diagram Edit Modul	24
4.1.7 Sequence diagram Delete Modul	26
4.2 Activity diagram	26
4.2.1 Activity diagram Manajemen Folder	27
4.2.2 Activity diagram Delete Folder	28
4.2.3 Activity diagram Manajemen File	29
4.2.4 Activity diagram Delete File	30
4.2.5 Activity diagram Tambah Modul	31
4.2.6 Activity diagram Edit Modul	32
4.2.7 Activity diagram Delete Modul	33
BAB 5 Hasil dan Pembahasan	35
5.1 Analisis Model Flow Graph	35
5.1.1 Model Flow Graph Manajemen Folder	35
5.1.2 Model Flow Graph Delete Folder	37
5.1.3 Model Flow Graph Manajemen File	41
5.1.4 Model Flow Graph Delete File	44
5.1.5 Model Flow Graph Tambah Modul	46
5.1.6 Model Flow Graph Edit Modul	49
5.1.7 Model Flow Graph Delete Modul	54

5.2 Hasil Test case	57
5.3 Pelaksanaan Pengujian.....	62
5.4 Analisis Hasil Uji	62
BAB 6 Kesimpulan dan Saran	68
6.1 Kesimpulan	68
6.2 Saran.....	68
Daftar Pustaka	69



Daftar Tabel

Tabel 3.1 Template Tabel Test case dan Test Procedure.....	18
Tabel 3.2 Template Tabel Hasil Pengujian	18
Tabel 4.1 Object Method Association Table Manajemen Folder	20
Tabel 4.2 Object Method Association Table Delete Folder	21
Tabel 4.3 Object Method Association Table Manajemen File	22
Tabel 4.4 Object Method Association Table Delete File	23
Tabel 4.5 Object Method Association Table Tambah Modul.....	24
Tabel 4.6 Object Method Association Table Edit Modul	24
Tabel 4.7 Object Method Association Table Delete Modul	26
Tabel 4.8 Method Activity Table Delete Modul	27
Tabel 4.9 Method Activity Table Delete Modul	28
Tabel 4.10 Method Activity Table Delete Modul	29
Tabel 4.11 Method Activity Table Delete Modul	30
Tabel 4.12 Method Activity Table Delete Modul	31
Tabel 4.13 Method Activity Table Delete Modul	32
Tabel 4.14 Method Activity Table Delete Modul	33
Tabel 5.1 Tabel Test case Aplikasi PDTT UB	58
Tabel 5.2 Tabel Hasil Pengujian Aplikasi PDTT-UB.....	65

Daftar Gambar

Gambar 2.1 Halaman depan aplikasi PDTT-UB	4
Gambar 2.2 Black Box Testing.....	5
Gambar 2.3 White Box Testing	6
Gambar 2.4 Grey Box Testing.....	6
Gambar 2.5 Contoh Activity diagram.....	9
Gambar 2.6 Contoh Bentuk Object Method Association Table (OMAT)	12
Gambar 2.7 Contoh Bentuk Method activity table (MAT).....	12
Gambar 2.8 Contoh Model Flow Graph (MFG).....	13
Gambar 3.1 Alur Proses Metode Behavior UML.....	17
Gambar 4.1 Sequence diagram Manajemen Folder	19
Gambar 4.2 Sequence diagram delete folder	20
Gambar 4.3 Sequence diagram Manajemen File.....	21
Gambar 4.4 Sequence diagram Delete File.....	22
Gambar 4.5 Sequence diagram Tambah Modul	23
Gambar 4.6 Sequence diagram Edit Modul	25
Gambar 4.7 Sequence diagram Delete Modul.....	26
Gambar 4.8 Activity diagram Manajemen Folder.....	27
Gambar 4.9 Activity diagram Delete Folder.....	28
Gambar 4.10 Activity diagram Manajemen File	29
Gambar 4.11 Activity diagram Delete File	30
Gambar 4.12 Activity diagram Tambah Modul.....	31
Gambar 4.13 Activity diagram Edit Modul.....	32
Gambar 4.14 Activity diagram Delete Modul	33
Gambar 5.1 MFG Sequence diagram Manajemen Folder	35
Gambar 5.2 MFG Activity diagram Manajemen Folder	36
Gambar 5.3 MFG Sequence diagram Delete Folder	38
Gambar 5.4 MFG Activity diagram Delete Folder	39
Gambar 5.5 MFG Sequence diagram Manajemen File	41
Gambar 5.6 MFG Activity diagram Manajemen File.....	42
Gambar 5.7 MFG Sequence diagram Delete File	44
Gambar 5.8 MFG Activity diagram Delete File.....	45
Gambar 5.10 MFG Activity diagram Tambah Modul	46

Gambar 5.9 MFG Sequence diagram Tambah Modul	47
Gambar 5.11 MFG Activity diagram Edit Modul.....	49
Gambar 5.12 MFG Sequence diagram Edit Modul	50
Gambar 5.13 MFG Sequence diagram Delete Modul.....	54
Gambar 5.14 MFG Activity diagram Delete Modul.....	55



BAB 1 Pendahuluan

1.1 Latar Belakang

Pada era globalisasi saat ini perkembangan teknologi informasi sangat pesat. Dengan perkembangan tersebut semakin banyak pekerjaan yang menjadi praktis dengan dukungan *software-software* yang canggih. Semakin hari, kebutuhan akan *software* yang berkualitas juga semakin dibutuhkan. Untuk membuat *software* yang berkualitas salah satu cara yang di gunakan adalah memaksimalkan salah satu fase yang ada dalam *Software Development Lifecycle* (SDLC) yaitu fase Testing. SDLC adalah pendekatan bertahap untuk melakukan analisis dan membangun rancangan sistem dengan menggunakan siklus yang spesifik terhadap kegiatan pengguna. *Software testing* atau jika di artikan ke dalam Bahasa Indonesia adalah Pengujian *Software* merupakan hal yang penting untuk menentukan kualitas dari perangkat lunak. Dengan melakukan Software Testing maka bug atau *error* pada *software* dapat segera ditemukan dan diperbaiki.

Universitas Brawijaya merupakan salah satu institusi perguruan tinggi di Indonesia yang memberikan layanan kepada masyarakat dengan menyiapkan sumber daya manusia yang berkualitas baik. UB sendiri memiliki tujuan yaitu “menghasilkan sumber daya manusia yang berkualitas, bertaqwa kepada tuhan YME dan menjadi akademisi yang tangguh dan professional yang mampu bersaing di dalam maupun luar negeri”. Selain itu UB juga mengarahkan mahasiswanya untuk menjadi *Entrepreneur* dan tidak tertinggal dalam hal teknologi. Untuk mendukung hal-hal tersebut, Universitas Brawijaya menggunakan metode pembelajaran jarak jauh yang dapat meningkatkan efektifitas pembelajaran. Untuk mendukung metode pembelajaran jarak jauh tersebut, Badan Pengembangan Teknologi Informasi dan Komunikasi (BPTIK) selaku bagian pengembangan *software* di Fakultas Ilmu Komputer (FILKOM) membuat *software* bernama APLIKASI PEMBELAJARAN DARING TERBUKA TERPADU yang digunakan untuk mendukung metode pembelajaran jarak jauh yang diusung Universitas Brawijaya. Dengan adanya aplikasi ini, maka metode pembelajaran jarak jauh yang di usung Universitas Brawijaya akan dapat berjalan dengan baik.

Aplikasi Pembelajaran Jarak Jauh ini merupakan *software* baru di kalangan UB. Untuk meminimalisir *error*, *fault* dan *failure* maka perlu dilakukan *software testing* kepada Aplikasi Pembelajaran Jarak Jauh ini. Tujuan dari Testing sendiri adalah memastikan bahwa apa yang direncanakan di awal atau pada requirement dapat sesuai dengan hasil yang telah di buat dalam bentuk aplikasi, proses-proses yang diinginkan dapat berjalan dengan baik sesuai requirement serta meminimalisir *error*, *fault* dan *failure* dalam aplikasi PDTT terutama pada modul administrator. Dalam modul ini perlu di uji dengan tepat karena merupakan modul vital yang menjadi sumber pada aplikasi PDTT. Banyak metode

yang dapat di gunakan untuk melakukan pengujian pada Aplikasi PDTT ini seperti metode *blackbox testing*, *whitebox testing*, dan *greybox testing*.

Salah satu bagian dalam *software testing* yang digunakan dalam metode pengujian apapun adalah *Test case*. "*Test case* adalah sekumpulan dokumen dari input test, kondisi yang akan dieksekusi, dan hasil yang diharapkan" (Galin, 2004). Membuat *Test case* yang baik sangat penting agar kelancaran saat pengujian dapat berjalan baik. Dalam pengujian Aplikasi PDTT ini belum terdapat *Test case* yang jelas sehingga perlu adanya *Test case* untuk pengujian Aplikasi PDTT. Membuat *Test case* tidak mudah, karena perlu ketepatan dengan tujuan dibuatnya aplikasi PDTT. "*Test case* mungkin tidak layak untuk memverifikasi kebenaran untuk setiap kasus pada regresi yang besar" (McCabe, 1996). Namun setidaknya dengan *Test case* dapat meminimalisir *error*, *fault* dan *failure* yang ada, Semakin tepat *Test case* semakin tinggi pula kualitas Aplikasi PDTT ini. Salah satu cara pembuatan *Test case* adalah mengacu pada Behavior UML Aplikasi PDTT. Dengan mengacu pada Behavior UML, *Test case* yang dihasilkan tepat dan pengujian pada aplikasi PDTT dapat berjalan dengan baik sehingga kualitas *software* akan tinggi.

Kesulitan menentukan *Test case* yang tepat dapat diatasi dengan menggunakan metode behavior UML yang kelebihanannya yang pertama adalah biaya design *test case* lebih kecil daripada biaya design *test case* manual seperti *white box testing* yang memiliki biaya design *test case* yang tinggi dan yang kedua adalah *reliability* dari *test coverage* tinggi (Swain & Mohapatra, 2010). Oleh karena itu penulis mengambil judul skripsi "Pengujian Perangkat Lunak Dengan Menggunakan Behavior UML (Studi Kasus : Aplikasi Pembelajaran Daring Terbuka Terpadu UB)".

1.2 Rumusan Masalah

- 1 Bagaimana proses pembuatan *Test case* pada Aplikasi PDTT dengan metode Behavior UML teknik Model Flow Graph (MFG)?
- 2 Apa hasil pengujian pada Aplikasi PDTT dengan metode Behavior UML?

1.3 Tujuan

- 1 Membuat *test case* Behavior UML menggunakan metode Model Flow Graph (MFG)
- 2 Mendapatkan hasil pengujian pada aplikasi PDTT dengan metode behavior UML

1.4 Manfaat

- 1 Mendapatkan dokumentasi Pengujian Aplikasi PDTT UB untuk developer PDTT-UB
- 2 Menghasilkan rekomendasi yang dapat dijadikan acuan oleh developer PDTT-UB untuk Aplikasi PDTT UB memperbaiki *error*, *fault* dan *failure* yang ada didalam aplikasi.

1.5 Batasan Masalah

- 1 Objek yang di uji adalah Aplikasi Pembelajaran Daring Terbuka Terpadu (PDTT) Universitas Brawijaya modul Administrator
- 2 UML yang digunakan adalah *Activity diagram* dan *Sequence diagram*

1.6 Sistematika Pembahasan

Di dalam laporan ini terdapat 6 Bab dengan penjelasan BAB 1 adalah pendahuluan yang berisi Latar Belakang, Rumusan Masalah, Tujuan, Manfaat, Batasan dan Sistematika Pembahasan. Pada bab 1 ini hal-hal yang ditekankan adalah kenapa penelitian ini dibuat, metode apa yang digunakan, objek apa yang digunakan dan seberapa besar cakupan masalah yang dibahas pada bab 1 ini. Tujuan dari bab 1 ini adalah sebagai alasan kenapa penelitian ini dibuat dan batas-batas yang dibahas pada penelitian ini

Pada Bab 2 adalah Landasan Kepustakaan yang berisi Teori-teori yang di gunakan dalam pengujian ini serta beberapa pengertian yang berhubungan dengan pengujian ini. Pada bab 2 ini tujuannya adalah sebagai acuan dasar teori-teori yang digunakan dalam penelitian ini dan sebagai tempat rujukan jika ada beberapa teori atau Bahasa yang belum dimengerti oleh pembaca penelitian ini sehingga tidak melenceng jauh dari maksud penelitian ini.

Bab 3 adalah Metodologi Penelitian yang berisi metode serta langkah-langkah yang digunakan saat melakukan pengujian. Di bab ini dijelaskan bagaimana penelitian ini dilakukan step by step dari awal sampai akhir. Tujuan dari bab 3 ini sebagai acuan dalam melakukan penelitian ini sehingga sesuai dengan tujuan awal dilakukannya penelitian ini.

Pada Bab 4 adalah Perancangan dan Analisis proses pembuatan *Activity diagram* dan *Sequence diagram* dari aplikasi PDTT beserta beberapa komponen yang diperlukan untuk analisis selanjutnya. Di bab ini adalah fase pembuatan *activity diagram* dan *sequence diagram* dari fitur-fitur yang akan di uji pada aplikasi PDTT yang digunakan untuk dasar pembuatan *Test case* pada bab selanjutnya

Bab 5 adalah Hasil dan Pembahasan yang berisi analisis dari diagram yang sudah dibuat sehingga menjadi *Test case* yang juga di uji ke aplikasi PDTT-UB. Pada bab ini berisi hasil dan pembahasan dari bab 4 yang kemudian di implementasikan ke aplikasi PDTT-UB sehingga muncul hasil pengujian aplikasi PDTT-UB berdasarkan analisis yang ada di bab sebelumnya.

Bab 6 adalah Kesimpulan dan saran yang berisi kesimpulan dan rekomendasi untuk aplikasi PDTT-UB. Pada bab ini hasil penelitian akan disimpulkan dan akan menjawab dari rumusan masalah yang ada pada bab sebelumnya, selain itu akan ada juga rekomendasi untuk developer PDTT-UB untuk kedepannya yang didapatkan dari hasil dari penelitian ini.

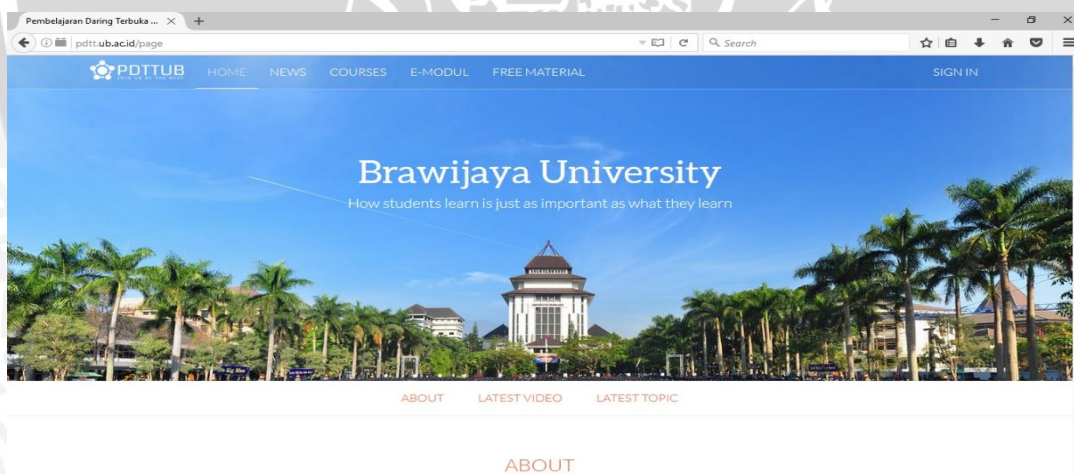
BAB 2 Landasan Kepustakaan

2.1 Penelitian Terkait

Pada tahun 2010 tepatnya di tanggal 8 September 2010, dua orang peneliti yaitu Santosh Kumar Swain asal School of Computer Engineering KIIT University, Bhubaneswar Orissa, India dan Durga Prasad Mohapatra asal Department of Computer Science & Engineering National Institute of Technology, Rourkela Orissa, India membuat sebuah penelitian yang berjudul “*Test case Generation from Behavioral UML*”. Penelitian ini berisikan tentang pembuatan *Test case* dari suatu aplikasi yang dasarnya dibuat dari *Activity diagram* dan *Sequence diagram*.

Dalam penelitian tersebut, dijelaskan bagaimana suatu *activity diagram* dan *sequence diagram* dari sebuah aplikasi di analisis menggunakan metode yang mereka namakan Model Flow Graph (MFG) sehingga diagram-diagram tersebut berubah menjadi suatu flow graph. Dari flow graph tersebut dapat di temukan beberapa alur atau jalan dari suatu program yang bisa digunakan sebagai pembuatan *Test case* seperti yang di lakukan di white box testing. Setelah itu dalam penelitian tersebut juga menjelaskan kelebihan pengujian menggunakan UML dalam hal ini *activity diagram* dan *sequence diagram* daripada menggunakan pengujian biasa.

2.2 Aplikasi Pembelajaran Daring Terbuka Terpadu UB (PDTT-UB)



Gambar 2.1 Halaman depan aplikasi PDTT-UB

Sumber : website www.pdt.ub.ac.id

Aplikasi PDTT- UB adalah aplikasi yang digunakan untuk pembelajaran jarak jauh antara dosen dan mahasiswa untuk menambah model pembelajaran yang ada di Universitas Brawijaya. Awalnya aplikasi ini bernama PJJ (pembelajaran jarak jauh) yang awalnya akan digunakan hanya di Fakultas Ilmu Komputer (FILKOM), tetapi seiring berjalannya waktu seluruh fakultas di Universitas Brawijaya juga membutuhkan aplikasi tersebut sehingga muncul lah aplikasi PJJ yang skalanya adalah seluruh UB yang dinamai Aplikasi Pembelajaran Daring

Terbuka Terpadu (PDTT) Universitas Brawijaya. Aplikasi PDTT-UB ini berbentuk website seperti yang ditunjukkan pada gambar 2.1 sehingga mudah di akses dimana saja baik di desktop maupun mobile.

2.3 Pengujian

2.1.1 Definisi

Menurut ANSI / IEEE 1059 Standart, Pengujian dapat didefinisikan sebagai berikut “suatu proses menganalisis item perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dan diperlukan (cacat / kesalahan / bug) dan mengevaluasi fitur item perangkat lunak”. Selain dari ansi, ada pakar juga yang mendefinisikan pengujian sebagai berikut “Pengujian perangkat lunak adalah proses formal yang dilakukan oleh tim khusus pengujian di mana suatu unit perangkat lunak, beberapa unit perangkat lunak yang terintegrasi atau paket perangkat lunak yang diperiksa secara keseluruhan dengan menjalankan program pada computer”. (Galin, 2004)

2.1.2 Tujuan

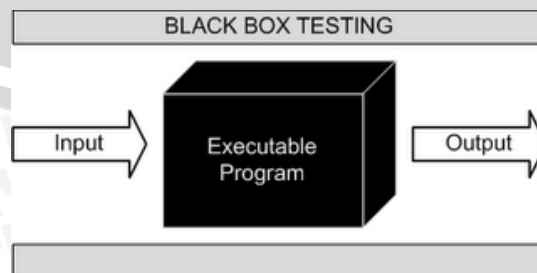
Menurut Daniel Galin pada buku *Software Quality Assurance From Theory to Implementation* (Galin, 2004), ada tujuan langsung dan tujuan tidak langsung dari Pengujian perangkat lunak. Tujuan Langsung sebagai berikut :

1. Untuk mengidentifikasi dan mengungkapkan sebagai kesalahan sebanyak mungkin dalam perangkat lunak yang diuji
2. Untuk membawa perangkat lunak yang telah diuji, setelah memperbaiki kesalahan yang diidentifikasi dan melakukan pengujian ulang pada tingkat kualitas yang memadai
3. Untuk melakukan tes yang diperlukan secara efektif dan efisien

2.1.3 Klasifikasi Pengujian

2.1.3.1 Pengujian Black Box

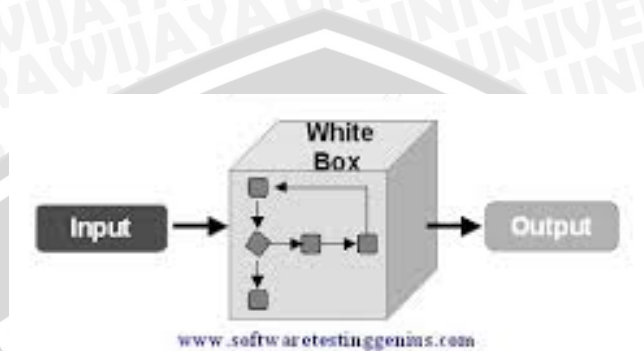
Pengujian dilakukan dengan mengeksekusi data uji dan mengecek apakah fungsional perangkat lunak bekerja dengan baik. Data uji didapatkan dari spesifikasi perangkat lunak, yang dalam hal ini menjelaskan fungsional perangkat lunak. Dengan kata lain, pengujian blackbox hanya menguji input dan output bagaimana tanpa mengetahui isi dalam aplikasi yang di gambarkan pada gambar 2.2



Gambar 2.2 Black Box Testing
sumber : www.softwaretestinggenius.com

2.1.3.2 Pengujian White Box

Pengujian dilakukan dengan menggunakan data uji yang menguji semua elemen program perangkat lunak (data internal, loop, algoritma dan jalur) dari perangkat lunak. Dengan kata lain, pengujian white box menguji hal-hal yang ada di dalam aplikasi seperti yang digambarkan pada gambar 2.3



Gambar 2.3 White Box Testing
sumber : www.softwaretestinggenius.com

2.1.3.3 Pengujian Grey Box

Pengujian Grey box adalah perpaduan antara pengujian black box dengan pengujian whitebox yang saling mengisi satu sama lain seperti digambarkan pada gambar 2.4 .



Gambar 2.4 Grey Box Testing
sumber : www.softwaretestinggenius.com

2.2 Sistem

Sistem berasal dari bahasa latin (systēma) dan bahasa yunani (sustēma) adalah sekumpulan unsur atau elemen yang saling berkaitan dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan. Berikut merupakan pengertian sistem menurut para ahli :

1. Menurut Raymond Mc Leod, Jr,
"Sistem adalah sekelompok elemen-elemen yang terintegrasi dengan maksud yang sama untuk mencapai tujuan" (McLeod, 2008)
2. Menurut Fathansyah, Ir
"Sebuah tatanan (keterpaduan) yang terdiri atas sejumlah komponen fungsional (dengan satuan fungsi / tugas khusus) yang saling

berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses pekerjaan tertentu” (Fathansyah, 2001)

3. Menurut Hartono

“Sistem adalah suatu jaringan kerja prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sarana tertentu” (Hartono, 1999)

Dari beberapa pengertian di atas maka dapat disimpulkan bahwa Sistem adalah sekelompok jaringan kerja yang terdiri dari sejumlah komponen fungsional yang bekerja secara bersama-sama dan memiliki satu tujuan yang sama.

2.3 Informasi

Informasi merupakan hal yang sangat penting dalam suatu organisasi. Suatu sistem yang kurang mendapatkan informasi akan menjadi kurang maju dan pada akhirnya organisasi tersebut akan berakhir. Sedangkan pengertian informasi itu sendiri adalah data yang telah di proses atau data yang memiliki arti. Berikut merupakan pengertian informasi menurut para ahli :

1. Menurut Raymond Mc Leod, Jr

“Informasi sesungguhnya berasal dari kata data yang kemudian di proses sehingga data tersebut memiliki arti bagi pemakainya” (McLeod, 2008)

2. Menurut Gordon B. Davis

“Informasi adalah data yang diolah menjadi suatu bentuk yang penting bagi penerima dan mempunyai nilai yang nyata dan dapat digunakan untuk mengambil keputusan baik sekarang maupun yang akan datang” (Davis, 2008)

Sumber dari informasi adalah data yang berupa huruf, simbol,alphabet, dan sebagainya. Sistem informasi mempunyai elemen utama,yaitu data yang menyediakan informasi prosedur yang memberitahu pengguna bagaimana mengoperasikan sistem informasi,menyelesaikan masalah, membuat keputusan dan menggunakan sistem informasi tersebut. Orang orang dalam sistem informasi membuat prosedur untuk mengelola dan memanipulasi data sehinggamenhasilkan informasi dan memanipulasi data sehinggamenhasilkan informasi dan menyebarkan informasi tersebut kelingkungannya

2.4 Sistem Informasi

Sistem Informasi adalah suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai tujuan,yaitu menyajikan informasi. “Dan didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang ditentukan” (Jogiyanto, 2001).

Beberapa pengertian Sistem Informasi menurut para ahli :

1. Menurut Alter : “Sistem informasi adalah kombinasi antara prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi” (Alter, 1992)
2. Turban, McLean dan Wetherbe: “Sebuah sistem informasi mengumpulkan, memproses, menyimpan, menganalisis, dan menyebarkan informasi untuk tujuan yang spesifik” (Turban, et al., 1999)

2.5 Unified Modeling Language (UML)

UML (Unified Modeling Language) adalah sebuah bahasa yang berdasarkangrafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasiandari sebuah sistem pengembangan *software* berbasis OO (Object-Oriented). UML tidak hanya merupakan sebuah bahasa pemrograman visual saja, namun juga dapat secara langsungdihubungkan ke berbagai bahasa pemrograman, seperti JAVA, C++, Visual Basic, atau bakhandihubungkan secara langsung ke dalam sebuah object-oriented database.

Diagram yang terdapat pada UML sebagai berikut :

1. Use Case Diagram
2. *Activity diagram*
3. *Sequence diagram*
4. Communication Diagram
5. Class Diagram
6. State Machine Diagram
7. Component Diagram
8. Deployment Diagram
9. Composite Diagram
10. Interaction Overview Diagram
11. Object Diagram
12. Package Diagram
13. Timing Diagram

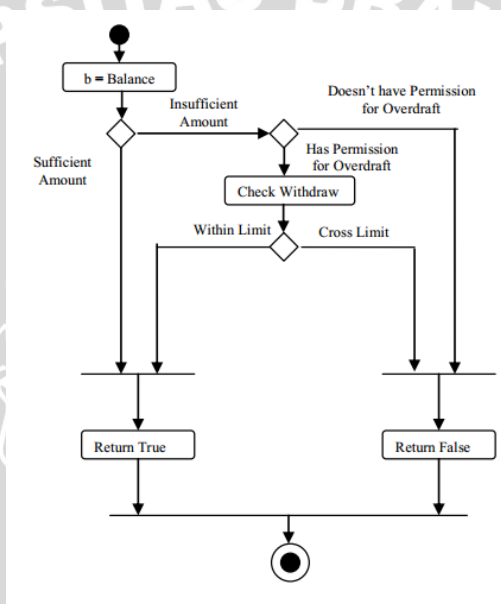
2.5.1 *Sequence diagram*

Sequence diagram (diagram urutan) adalah suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut termasuk pengguna, display, dan sebagainya berupa pesan/message.

Sequence diagram digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai sebuah respon dari suatu kejadian/even untuk menghasilkan output tertentu. *Sequence diagram* diawali dari apa yang me-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan (Pratama, 2015).

2.5.2 Activity diagram

Pada dasarnya diagram Activity sering digunakan oleh flowchart. Diagram ini berhubungan dengan diagram Statechart. Diagram Statechart berfokus pada obyek yang dalam suatu proses (atau proses menjadi suatu obyek), diagram Activity berfokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain. Sebagai contoh, perhatikan proses yang terjadi. “Pengambilan uang dari bank melalui ATM.” Ada tiga aktifitas kelas (orang, dan lainnya) yang terkait yaitu : Customer, ATM, and Bank. Proses berawal dari lingkaran start hitam pada bagian atas dan berakhir di pusat lingkaran stop hitam/putih pada bagian bawah. Aktivitas digambarkan dalam bentuk kotak persegi. Untuk lebih jelasnya dapat dilihat pada gambar 2.5 yang merupakan sebuah contoh dari *activity diagram*.



Gambar 2.5 Contoh Activity diagram

Sumber : Swain & Mohapatra (2010)

Diagram Activity dapat dibagi menjadi beberapa jalur kelompok yang menunjukkan obyek mana yang bertanggung jawab untuk suatu aktifitas. Peralihan tunggal (single transition) timbul dari setiap adanya activity (aktifitas), yang saling menghubungkan pada aktifitas berikutnya. Sebuah transition (transisi) dapat membuat cabang ke dua atau lebih percabangan exclusive transition (transisi eksklusif). Label Guard Expression (ada didalam []) yang menerangkan output (keluaran) dari percabangan. Percabangan akan menghasilkan bentuk menyerupai bentuk intan. Transition bisa bercabang menjadi beberapa aktifitas paralel yang disebut Fork. Fork beserta join (gabungan dari hasil output fork) dalam diagram berbentuk solid bar (batang penuh).

2.6 Test case

Menurut Para ahli *Test case* dapat didefinisikan sebagai berikut

1. Menurut IEEE Standart : “*Test case* adalah sekumpulan dari input test, kondisi yang akan dieksekusi, dan hasil yang diharapkan”
2. Menurut Boris Beizer : “*Test case* merupakan input yang spesifik yang akan dicoba dan prosedur yang akan diikuti ketika sebuah *software* di *test*.” (Beizer, 1990)

Dari pengertian di atas, *Test case* berisi informasi tentang tujuan dari test, kebutuhan perangkat lunak maupun perangkat keras(jika ada), menspesifikasikan setup atau konfigurasi kebutuhan, gambaran mengenai bagaimana melakukan test tersebut, dan hasil yang diharapkan untuk test tersebut. Sebagai contoh table *Test case* ditunjukkan pada gambar 2.6 . Kegunaan dari mendesain *Test case* adalah sebagai berikut :

1. Memperbaiki efisiensi dalam menjalankan test
2. Menemukan bugs sebanyak mungkin
3. Menemukan bugs secepat mungkin
4. Menjadikan kode program tidak terlalu kompleks dan sederhana
5. Kode program tidak redundant

Test Case Id	Test Case name	Test Case Desc	Test Steps			Test Case Status	Test Status (P/F)	Test Priority	Defect Severity
			Step	Expect	Actual				
Login 01									

Gambar 2.6 Contoh Tabel *Test case*

Sumber : Beizer (1990)

2.6.1 Test Procedure

Test Procedure adalah dokumen yang mendeskripsikan langkah-langkah yang akan di ambil pada saat pengujian (termasuk penentuan urutan dan eksekusi dari suatu pengujian). *Test Procedure* juga disebut sebagai test Script.

Berikut adalah contoh dari *test procedure* :

Test procedure DB15: Set up customers for marketing campaign Y. Step

1: Open database with write privilege Step 2: Set up customer Bob

Flounders

male, 62, Hudsonville, contract

Step 3: Set up customer Jim Green

male, 17, Grand Rapids, pay-as-you-go, \$8.64

Step 4: ...

Test procedure MC03: Special offers for low-credit teenagers

Step 1: Get details for Jim Green from database Step 2:

Send text message offering double credit Step 3: Jim Green

requests \$20 credit, \$40 credited

2.7 Pengujian *Software* Dengan Behavior UML

Pengujian *software* dengan behavior UML adalah pengujian yang dilakukan berdasarkan UML seperti *activity diagram*, *sequence diagram*, *class diagram*, *state machine diagram* maupun diagram yang menggambarkan aplikasi tersebut. Dari diagram-diagram UML akan dianalisis sehingga menghasilkan beberapa point yang akan menjadi *Test case*. Pengujian Behavior UML termasuk grey-box testing yang merupakan perpaduan dari black-box testing dengan white-box testing. Ada beberapa UML yang digunakan dalam behavior UML ini, diantaranya adalah *activity diagram* dan *sequence diagram*. "*Sequence diagram* merepresentasikan dengan lengkap pesan-pesan yang dikirim menuju objek" (Swain & Mohapatra, 2010). Ketika pesan dikirimkan menuju objek, objek akan melakukan operasi sesuai apa yang diminta. Setelah pesan diterima, operasi yang diminta akan di eksekusi. Dengan kata lain, *sequence* yang merupakan white-box di dalam pengujian behavior UML. Sedangkan *activity diagram* mendeskripsikan alur yang terjadi dari masing-masing *activity*. "*Activity diagram* menekankan suatu aktivitas yang dilakukan object atau kumpulan object sehingga ini akan menjadikan *activity diagram* menjadi diagram yang tepat untuk merepresentasikan alur dari suatu aplikasi" (Swain & Mohapatra, 2010). Selain itu *activity diagram* juga menjelaskan *coverage criterion* yang digunakan untuk melengkapi test scenario. Dengan demikian *activity diagram* tidak menggambarkan alur yang ada di dalam sistem, melainkan hanya proses yang terjadi di luar sehingga *activity diagram* yang merupakan black-box di dalam pengujian behavior UML.

Proses behavior UML ada 3 (Swain & Mohapatra, 2010), yaitu :

1. Generate MFG dari *sequence diagram* dan *activity diagram*

MFG dibuat dengan menuluri alur *activity diagram* dan *sequence diagram* dari awal sampai akhir, yang didalamnya ada kondisi, method, eksekusi, dan perulangan. Dari proses yang ada di dalam *activity diagram* dan *sequence diagram* tersebut di kumpulkan dan diberi label. Kumpulan proses yang ada di *activity diagram* adalah *Method activity table (MAT)* dan untuk *sequence diagram* adalah *Object Method Association Table (OMAT)*. Setiap proses di MAT dan OMAT menjadi node dan setiap node akan saling dihubungkan sesuai alur dari diagram oleh edge. Contoh dari MAT dapat dilihat pada gambar 2.6 dan untuk contoh dari OMAT dapat dilihat pada gambar 2.7

TABLE 1: Object Method Association Table (OMAT) for the Fig. 1

SYMBOL	OBJECT-METHOD ASSOCIATION
A	atm:Validate Amount
B	ctact: Withdraw
C	atm: Display Message
D	ctact: Return
E	atm: Dispense Cash
F	atm: Print Receipt
G	atm: Return

Gambar 2.6 Contoh Bentuk Object Method Association Table (OMAT)

Sumber : Swain & Mohapatra (2010)

TABLE 2: Method Activity Table (MAT) for the Fig. 2

SYMBOL	ACTIVITY
A	Amount < Balance
B	Update Balance
C	Check for Overdraft
D	Check WithDraw Limit
E	Does not have Permission for Overdraft
F	With in Limit
G	Beyond Limit
H	Return True
I	Return False
J	Return

Gambar 2.7 Contoh Bentuk Method activity table (MAT)

Sumber : Swain & Mohapatra (2010)

2. Generate *test sequence* dari MFG sesuai dengan *sequence diagram* dan *activity diagram*

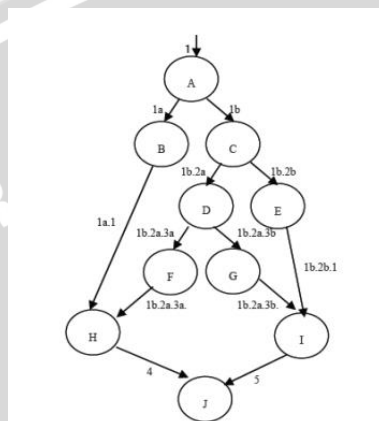
Proses yang ada di MFG dijalankan sesuai dengan kondisi yang didalamnya ada *pre-condition*, *main scenario* dan *post-condition* fitur tersebut yang akan menghasilkan skenario-skenario jalur yang didalam jalur tersebut terdapat *edge*, *method / activity* dan *category partition* yang dinamakan *test sequence*

3. Generate *test case* dari test sequence sesuai jalur uji

Untuk mendapatkan *test case*, yang pertama dilakukan adalah mencari jalur yang mungkin dalam MFG *sequence diagram* dan *activity diagram*. Dari setiap jalur di kombinasikan dengan test sequence sehingga muncul jalur yang sesuai dengan test sequence, jalur yang sesuai dengan test sequence tersebut yang menjadi *test case* yang dinamakan *test case generation*

2.8 Model Flow Graph

Model Flow Graph adalah gabungan dari beberapa titik (*nodes*) dan garis (*edges*). Flow graph digunakan untuk menggambarkan alur dari proses UML seperti *activity diagram*, *sequence diagram* maupun UML yang lain. Node akan merepresentasikan keseluruhan pernyataan atau potongan proses, sedangkan edge merepresentasikan aliran kontrol. Untuk lebih jelasnya dapat dilihat pada gambar 2.8

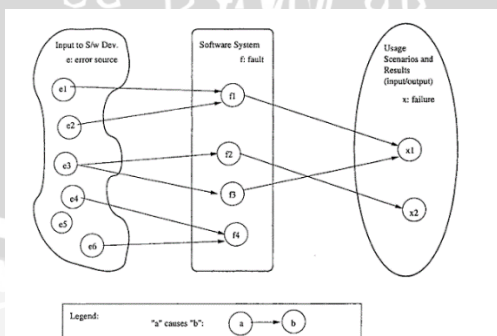


Gambar 2.8 Contoh Model Flow Graph (MFG)

Sumber : Swain & Mohapatra (2010)

2.9 Error, Fault, dan Failure

Tujuan umum dari pengujian *software* adalah mengurangi *error*, *fault*, dan *failure* yang ada dalam suatu aplikasi yang diujikan. "*Error* adalah aksi manusia yang membuat hasil yang salah. *Fault* adalah kesalahan langkah, proses, atau data dalam aplikasi. *Failure* adalah Ketidakmampuan sistem atau komponen untuk melakukan fungsi sesungguhnya" (Galin, 2004). Berikut merupakan penggambaran hubungan antara *error*, *fault* dan *failure* yang dijelaskan dalam gambar 2.8



Gambar 2.7 Hubungan Error, Fault, dan Failure

Sumber : Galin (2004)

Failure bermula dari programmer yang membuat *error*. *Error* disini dapat berarti *error syntax* bahasa pemrograman atau *error logika* yang menyebabkan

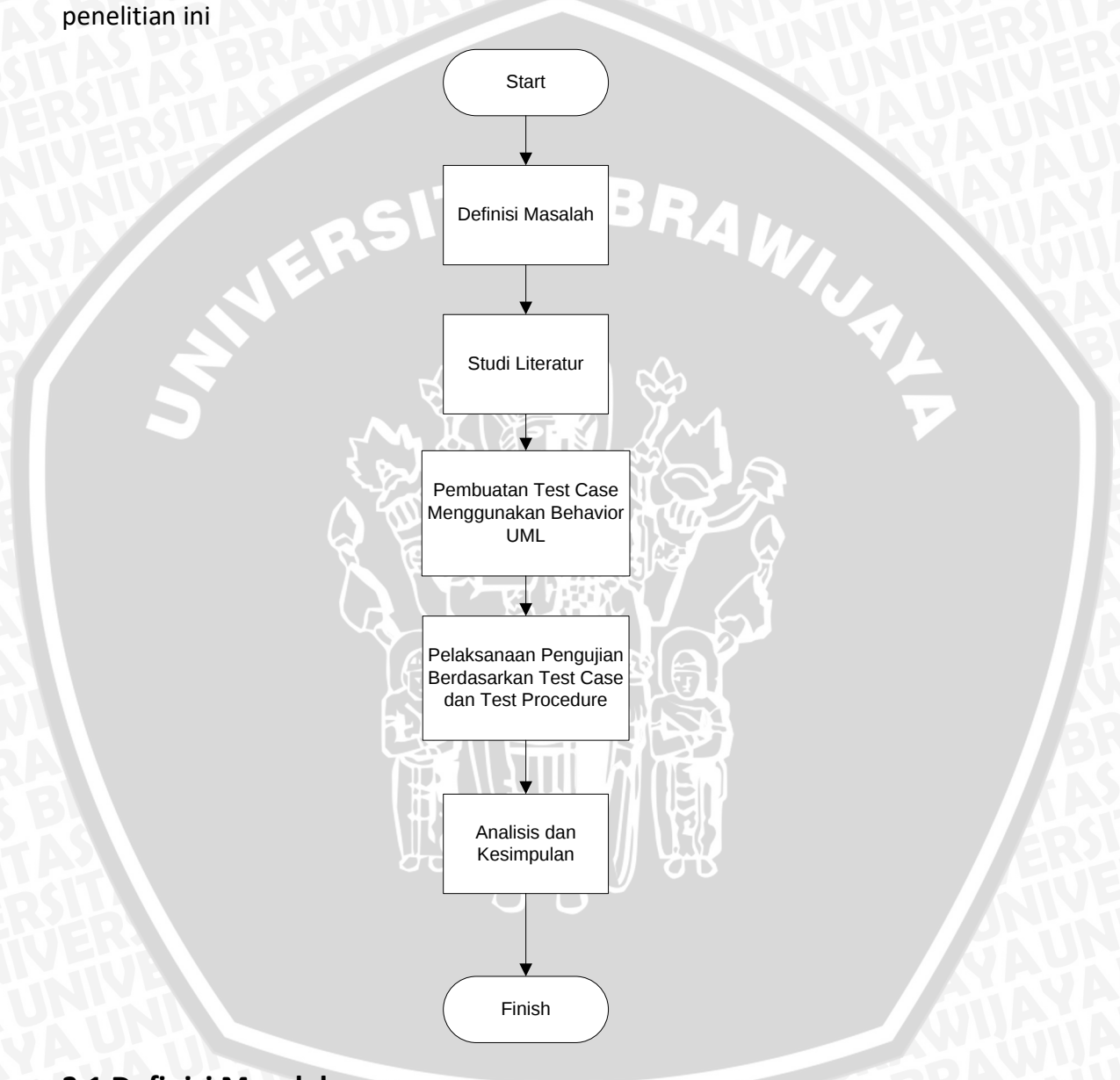
kesalahan dalam fungsionalitas aplikasi. *Error* dapat menyebabkan *fault*, namun tidak semua *error* menghasilkan *fault*. *Fault* akan menghasilkan *failure* apabila ada event pemicu terjadinya *failure*. Tidak semua *fault* akan menghasilkan *failure*

Didalam *fault* ada yang namanya *Activity Synchronization Fault*. *Activity Synchronization Fault* adalah kesalahan yang terjadi ketika suatu aktivitas mulai di eksekusi sebelum ada perintah ataupun *trigger* dari aktivitas yang lain.



BAB 3 Metodologi Penelitian

Kajian yang dilakukan dalam penelitian ini adalah tentang pengujian *software* pada Aplikasi Pembelajaran Daring Terbuka Terpadu UB pada modul Administrator. Untuk melakukan pengujian tersebut memerlukan metode penelitian untuk pengerjaannya. Berikut adalah metode yang di gunakan dalam penelitian ini



3.1 Definisi Masalah

Pada tahap ini yang dilakukan adalah berdasarkan dari pengalaman tentang kesulitan menentukan *Test case* yang tepat untuk pengujian aplikasi yang ada di Universitas Brawijaya, dilakukan pencarian metode-metode lain untuk menentukan *Test case* yang tepat. Pengumpulan literatur serta beberapa contoh *Test case* di lakukan untuk melakukan pengujian terhadap aplikasi PDTT UB modul administrator.

Selama ini proses pembuatan *Test case* hanya dilakukan berdasarkan fitur-fitur yang ada sehingga terkesan ada beberapa hal yang tidak tercover dari *Test case* tersebut. Pengujian pun tidak dapat dilakukan secara maksimal. Berawal dari hal tersebut, maka pengujian aplikasi PDTT UB akan di uji dengan menggunakan *Test case* yang di dapatkan dari metode behavior UML. Dari permasalahan tersebut akan di jadikan sebagai perumusan masalah oleh peneliti dalam pembuatan tugas akhir ini.

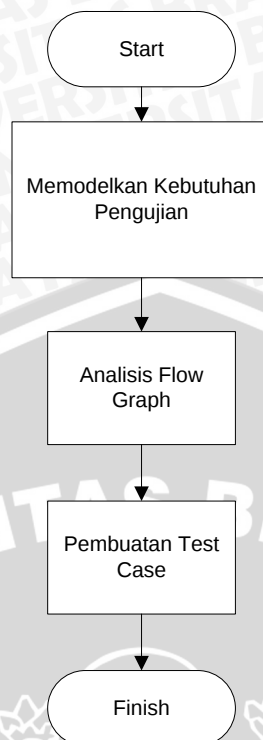
3.2 Studi Literatur

Melakukan Studi Literatur atau kajian pustaka yang berkaitan dengan Pengujian *Software* terutama metode behavior UML, UML secara umum dan secara khusus *Activity diagram*, *Sequence diagram*, (UML yang dipakai) sebagai dasar pembuatan *Test case*. Selain itu juga mempelajari Aplikasi Pembelajaran Daring Terbuka Terpadu (PDTT) UB modul administrator yang merupakan objek dari penelitian ini. Untuk paper yang lebih spesifik, dalam penelitian ini mengacu pada penelitian yang berjudul “*Test case Generation from Behavioral UML Models*” yang ditulis oleh Santosh Kumair Swain dan Durga Prasad Mohapatra pada September 2010.

Setelah melakukan studi literature, akan muncul apa saja yang diperlukan untuk melakukan penelitian ini yaitu objek yang diteliti dalam hal ini aplikasi PDTT-UB, kemudian modul yang diuji adalah modul administrator Karena menurut developer PDTT-UB ini adalah modul yang paling pokok dalam aplikasi ini. Setelah muncul objek dari penelitian, yang dilakukan adalah meminta sumber data dalam hal ini seharusnya adalah *activity diagram* dan *sequence diagram* dari fitur-fitur yang ada di aplikasi PDTT-UB. Namun, pada saat pengumpulan data, developer memberi tahu bahwa aplikasi PDTT-UB ini belum ada dokumennya Karena aplikasi diminta cepat jadi sehingga tidak ada dokumennya. Maka dari itu hal pertama yang dilakukan adalah membuat dahulu *Activity diagram* dan *Sequence diagram* dari masing-masing fitur yang ada di modul administrator aplikasi PDTT-UB.

3.3 Pembuatan *Test case* Menggunakan Behavior UML

Dalam pembuatan *Test case* menggunakan behavior UML ini ada beberapa tahap yang harus dilakukan untuk mendapatkan *Test case*. Proses ini mengacu pada penelitian yang berjudul “*Test case Generation from Behavioral UML Models*” yang ditulis oleh Santosh Kumar Swain dan Durga Prasad Mohapatra. untuk lebih lebih jelasnya dapat dilihat pada gambar 3.1



Gambar 3.1 Alur Proses Metode Behavior UML

Berdasarkan gambar 3.1, proses pertama adalah memodelkan kebutuhan pengujian. Untuk memenuhi kebutuhan pengujian dalam hal ini adalah *activity diagram* dan *sequence diagram* dari aplikasi PDTT-UB adalah observasi. Cara yang di gunakan untuk observasi adalah meminta source atau code dari aplikasi PDTT-UB kepada developer untuk di demo kan. Setelah melakukan demo aplikasi, proses selanjutnya adalah membuat *activity diagram* dan *sequence diagram* berdasarkan demo aplikasi dan source code dari aplikasi PDTT-UB tersebut.

Activity diagram digunakan karena *activity diagram* menekankan suatu aktivitas yang dilakukan objek atau kumpulan objek sehingga ini akan menjadikan *activity diagram* menjadi diagram yang tepat untuk merepresentasikan alur dari suatu aplikasi (Swain & Mohapatra, 2010). *Sequence diagram* digunakan karena *sequence diagram* merepresentasikan dengan lengkap pesan-pesan yang dikirim menuju objek (Swain & Mohapatra, 2010).

Fase kedua dari gambar 3.1 adalah analisis flow graph. Setelah *Activity Diagram* dan *sequence diagram* dibuat, selanjutnya adalah Generate MFG dari *sequence diagram* dan *activity diagram* dari setiap fitur yang hasilnya adalah MFG dari setiap fitur yang ada pada PDTT-UB modul administrator.

Fase ketiga dari gambar 3.1 adalah pembuatan *test case*. Pada fase yang pertama dilakukan adalah menggenerate MFG menjadi *test sequence*. Untuk

mendapatkan *test sequence* dibutuhkan kondisi dari fitur yang akan di buat *test case*, Karena belum ada dokumen dari PDTT UB yang berisi kondisi dari masing-masing fitur, maka perlu dibuat dulu kondisi dari masing-masing fitur berdasarkan proses demo aplikasi PDTT UB. Setelah itu baru dilakukan generate MFG menjadi *test sequence*. Setelah itu proses selanjutnya adalah generate MFG menjadi *Test case* menggunakan algoritma *test case generation* sehingga muncul *test case* dari masing-masing fitur. Setelah *test case* dari masing-masing fitur terbuat, selajutnya di tabelkan menggunakan tabel 3.1 untuk mempermudah pembacaan dan mempermudah proses pengujian

Tabel 3.1 Template Tabel *Test case* dan *Test Procedure*

No	Fitur	<i>Test case</i> ID	Input

3.4 Pelaksanaan Pengujian Berdasarkan *Test case* dan *Test Procedure*

Fase ini adalah fase pengujian aplikasi. Pengujian dilakukan langsung ke aplikasi PDTT-UB sesuai dengan *Test case* dan *test procedure* yang ada. Kemudian hasil pengujian akan ditabelkan dengan template seperti tabel 3.4 untuk mempermudah analisis selanjutnya.

Tabel 3.2 Template Tabel Hasil Pengujian

No	Fitur	<i>Test case</i> ID	Input	Expectation	Realization	Test Status	Note

3.5 Analisis dan Kesimpulan

Setelah pengujian selesai, analisis hasil pengujian dilakukan dengan membaca berapa fitur yang valid dan tidak valid. Kemudian dijelaskan setiap fitur yang tidak valid beserta permasalahannya dan bagaimana seharusnya yang dilakukan. Kemudian diambil kesimpulan yang berisi bagaimana cara pembuatan behavior UML dan bagaimana hasil ujinya.

BAB 4 Perancangan dan Analisis

Dalam bab ini akan dijelaskan bagaimana perancangan UML dalam pembuatan *Test case* yang mengacu pada paper yang berjudul “ *Test case Generation from Behavioral UML Models*” yang ditulis oleh Santosh Kumar Swain dan Durga Prasad Mohapatra, sehingga UML yang di gunakan dalam penelitian ini adalah *Activity diagram* dan *Sequence diagram* dari PDTT-UB. Fitur yang akan di buat *Test case* dari aplikasi PDTT-UB ini adalah fitur yang berada di modul administrator yaitu :

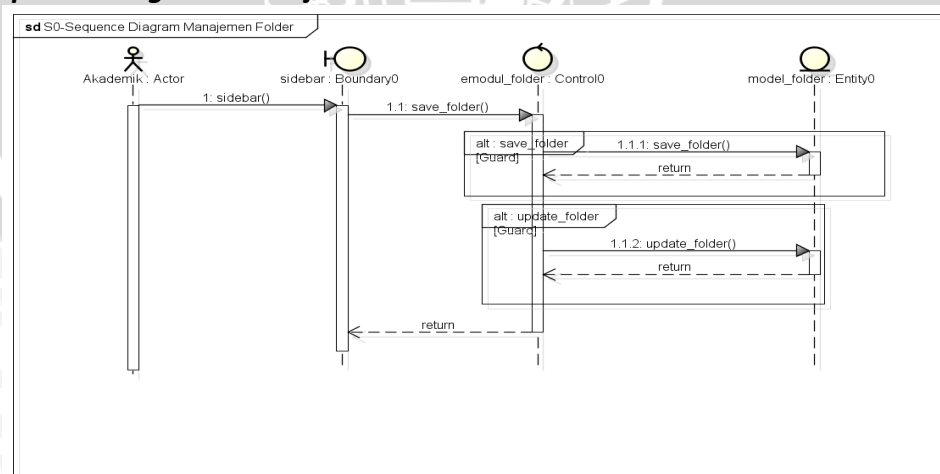
1. Manajemen Folder
2. Delete Folder
3. Manajemen File
4. Delete File
5. Tambah Modul
6. Edit Modul
7. Delete Modul

Untuk mendapatkan *Activity diagram* dan *Sequence diagram* dapat dilihat atau dirujuk dari dokumentasi perancangan aplikasi, tetapi karena aplikasi PDTT-UB ini sudah jadi dulu tanpa ada dokumentasi nya maka harus dilakukan pembuatan *Activity diagram* dan *Sequence diagram* terlebih dahulu

4.1 Sequence diagram

Dalam proses pembuatan *Sequence diagram*, akan dibuat 7 *sequence diagram* dari 7 fitur yang ada PDTT-UB . Setelah jadi *sequence diagram*, proses yang ada di dalam *sequence diagram* akan di buat titik point dan diberi symbol yang dikumpulkan dalam *Object Method Association Table*

4.1.1 Sequence diagram Manajemen Folder



powered by Astah

Gambar 4.1 Sequence diagram Manajemen Folder

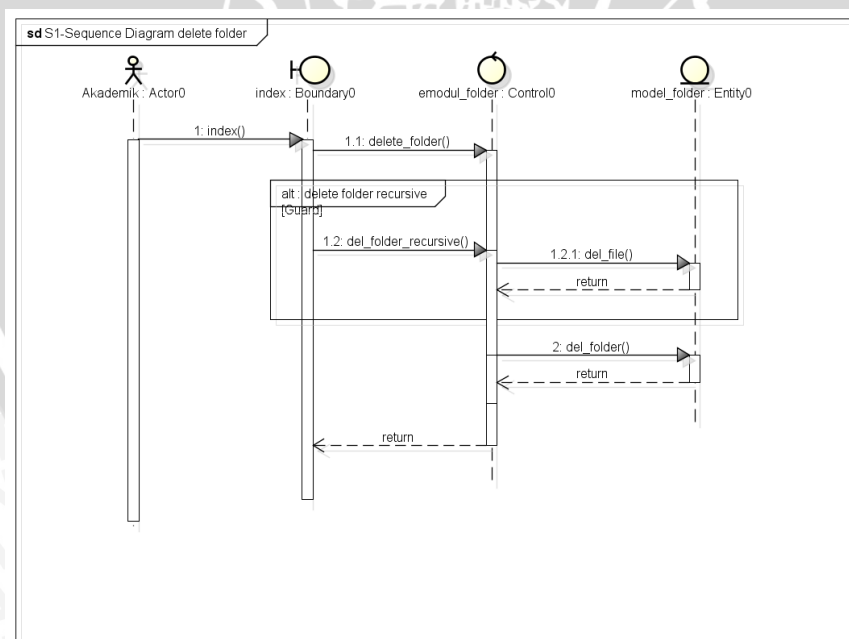
Fitur pertama dari aplikasi PDTT-UB adalah Manajemen Folder. Manajemen folder berisi proses penambahan folder dan pengubahan folder yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *sequence diagram* dari fitur manajemen folder yang dapat dilihat pada gambar 4.1.

Dari *sequence diagram* pada gambar 4.1 terdapat 10 proses yang berlangsung. Dari 10 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.1.

Tabel 4.1 Object Method Association Table Manajemen Folder

Symbol	Object Method
A	Enter_data
B	Save_folder
C	Check_folder_id
D	Id_folder = null
E	Save_folder
F	Return
G	Id_folder = not null
H	Update_folder
I	Return
J	Return

4.1.2 Sequence diagram Delete Folder



Gambar 4.2 Sequence diagram delete folder

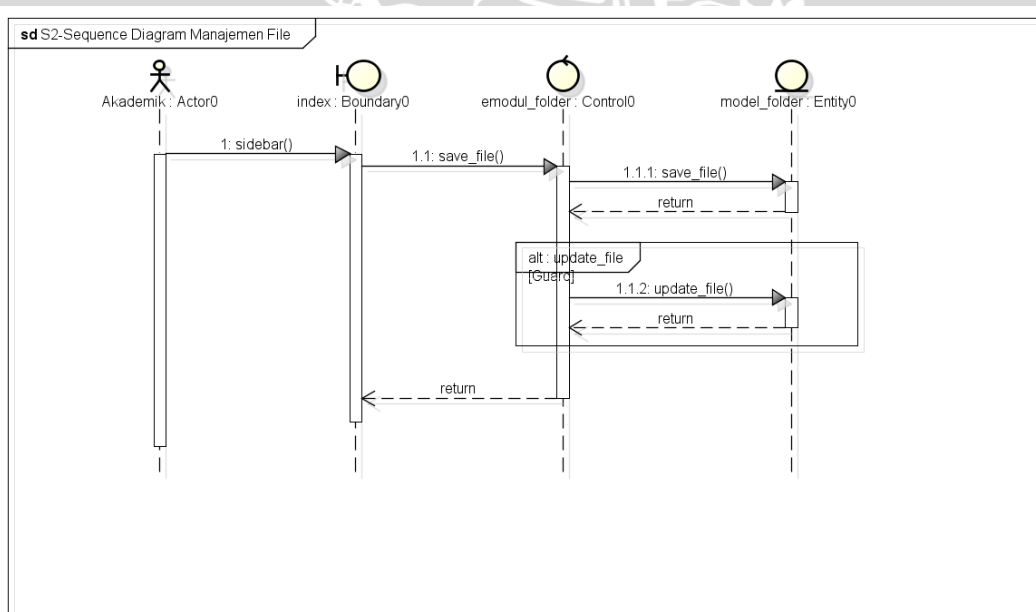
Fitur kedua dari aplikasi PDTT-UB adalah Delete Folder. Delete folder berisi proses menghapus folder yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *sequence diagram* dari fitur delete folder yang dapat dilihat pada gambar 4.2 .

Dari *sequence diagram* pada gambar 4.2 terdapat 11 proses yang berlangsung. Dari 11 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.2.

Tabel 4.2 Object Method Association Table Delete Folder

Symbol	Object Method
A	Delete
B	Delete_folder
C	Check_file
D	Ada file dalam folder
E	Tidak ada file dalam folder
F	Del_folder_recursive
G	Del_file
H	Return
I	Del_folder
J	Return
K	Return

4.1.3 Sequence diagram Manajemen File



powered by Astah

Gambar 4.3 Sequence diagram Manajemen File

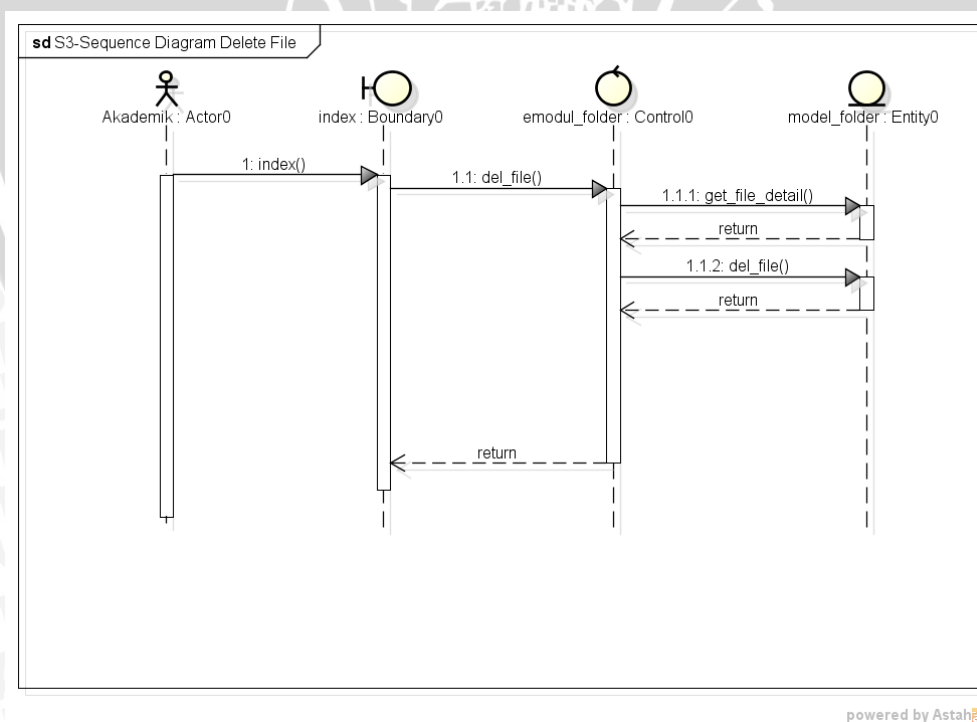
Fitur ketiga dari aplikasi PDTT-UB adalah Manajemen File. Manajemen file berisi proses penambahan dan pengubahan file yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *sequence diagram* dari fitur manajemen file yang dapat dilihat pada gambar 4.3 .

Dari *sequence diagram* pada gambar 4.3 terdapat 10 proses yang berlangsung. Dari 10 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.3.

Tabel 4.3 Object Method Association Table Manajemen File

Symbol	Object Method
A	Enter_data
B	Save_file
C	Check_file_id
D	Id_file = null
E	Id_file = not null
F	Save_file
G	Return
H	Update_file
I	Return
J	Return

4.1.4 Sequence diagram Delete File



powered by Astah

Gambar 4.4 Sequence diagram Delete File

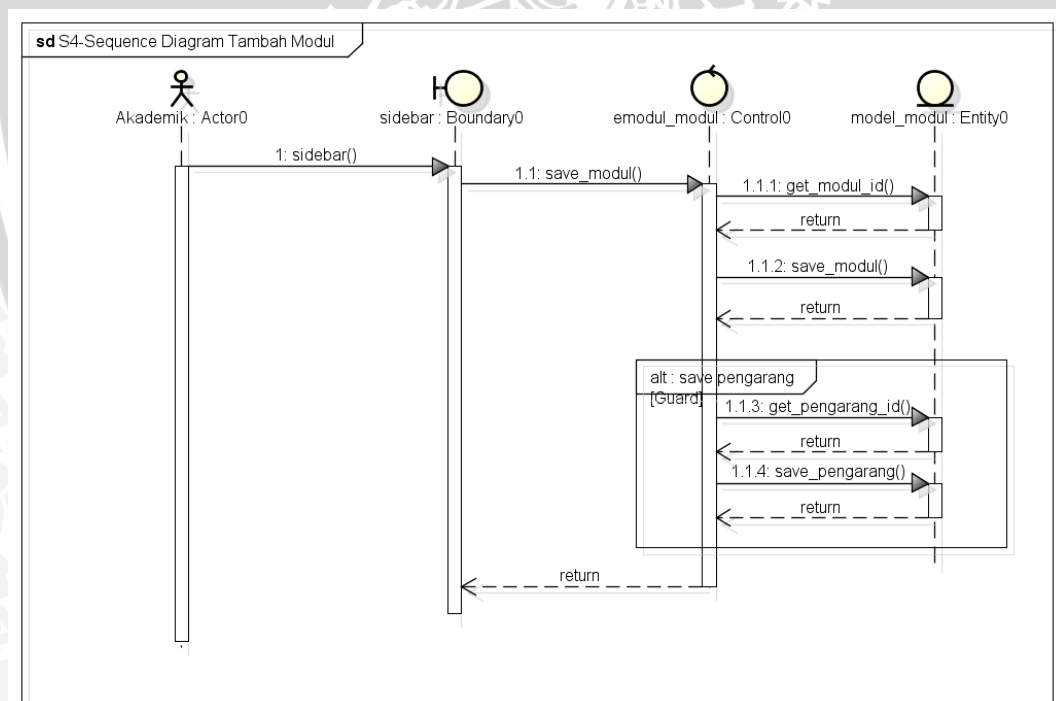
Fitur keempat dari aplikasi PDTT-UB adalah Delete File. Delete file berisi proses penambahan dan pengubahan file yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *sequence diagram* dari fitur delete file yang dapat dilihat pada gambar 4.4 .

Dari *sequence diagram* pada gambar 4.4 terdapat 7 proses yang berlangsung. Dari 7 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.4.

Tabel 4.4 Object Method Association Table Delete File

Symbol	Object Method
A	Delete
B	Del_file
C	Get_file_detail
D	Return
E	Del_file
F	Return
G	Return

4.1.5 Sequence diagram Tambah Modul



powered by Astah

Gambar 4.5 Sequence diagram Tambah Modul

Fitur kelima dari aplikasi PDTT-UB adalah Tambah Modul. Tambah modul berisi proses penambahan modul yang ada di dalam aplikasi PDTT-UB. Berikut

merupakan *sequence diagram* dari fitur tambah modul yang dapat dilihat pada gambar 4.5 .

Dari *sequence diagram* pada gambar 4.5 terdapat 14 proses yang berlangsung. Dari 14 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.5.

Tabel 4.5 Object Method Association Table Tambah Modul

Symbol	Object Method
A	Input_data
B	Save_modul
C	Get_modul_id
D	Return
E	Save_modul
F	Return
G	Check_pengarang_id
H	Pengarang = null
I	Pengarang = not null
J	Get_pengarang_id
K	Return
L	Save_pengarang
M	Return
N	Return

4.1.6 Sequence diagram Edit Modul

Fitur keenam dari aplikasi PDTT-UB adalah Edit Modul. Edit modul berisi proses pengubahan modul yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *sequence diagram* dari fitur edit modul yang dapat dilihat pada gambar 4.6 .

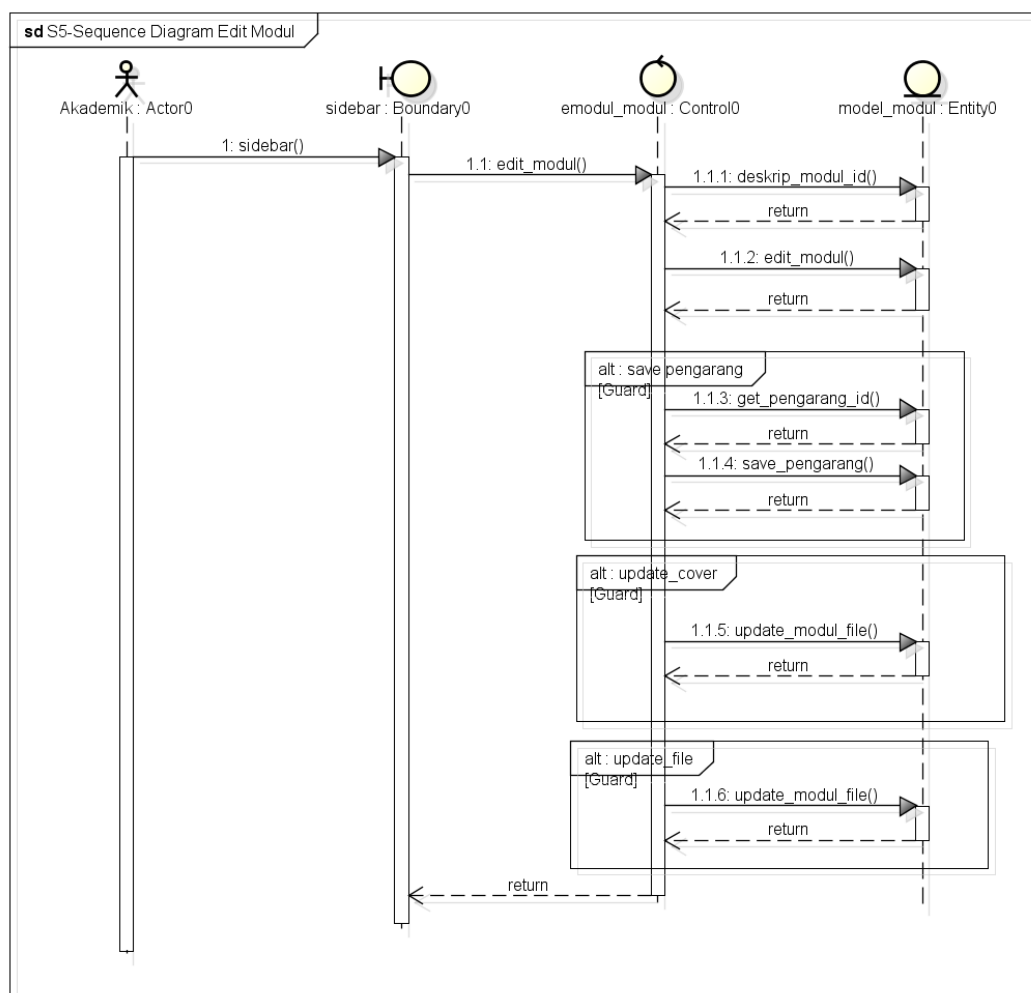
Dari *sequence diagram* pada gambar 4.6 terdapat 18 proses yang berlangsung. Dari 18 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.6.

Tabel 4.6 Object Method Association Table Edit Modul

Symbol	Object Method
A	Input_data
B	Edit_modul
C	Deskrip_modul_id
D	Return
E	Edit_modul
F	Return
G	Check_pengarang_id
H	Get_pengarang_id
I	Return
J	Save_pengarang

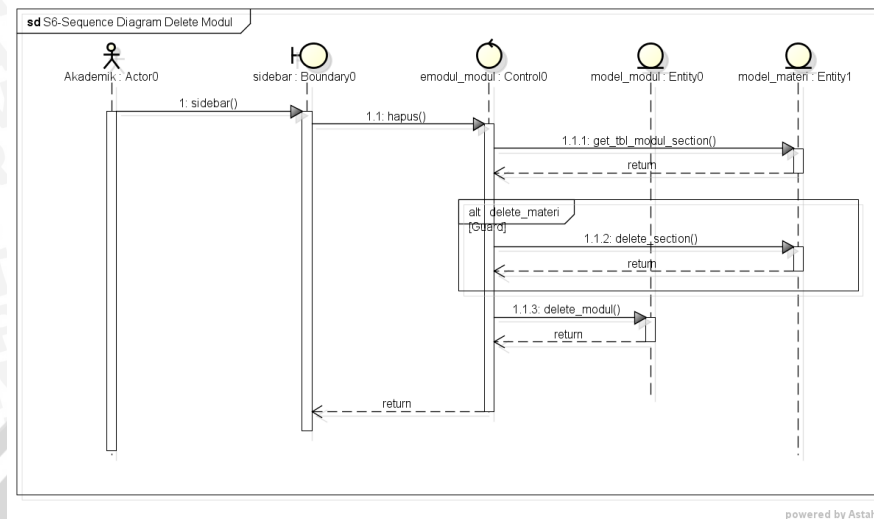
Tabel 4.6 Object Method Association Table Edit Modul (Lanjutan)

Symbol	Object Method
K	Return
L	Check_update_cover
M	Update_modul_file
N	Return
O	Check_update_file
P	Update_modul_file
Q	Return
R	Return



Gambar 4.6 Sequence diagram Edit Modul

4.1.7 Sequence diagram Delete Modul



Gambar 4.7 Sequence diagram Delete Modul

Fitur ketujuh dari aplikasi PDTT-UB adalah Delete Modul. Delete modul berisi proses penghapusan modul yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *sequence diagram* dari fitur delete modul yang dapat dilihat pada gambar 4.7 .

Dari *sequence diagram* pada gambar 4.7 terdapat 10 proses yang berlangsung. Dari 10 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.7.

Tabel 4.7 Object Method Association Table Delete Modul

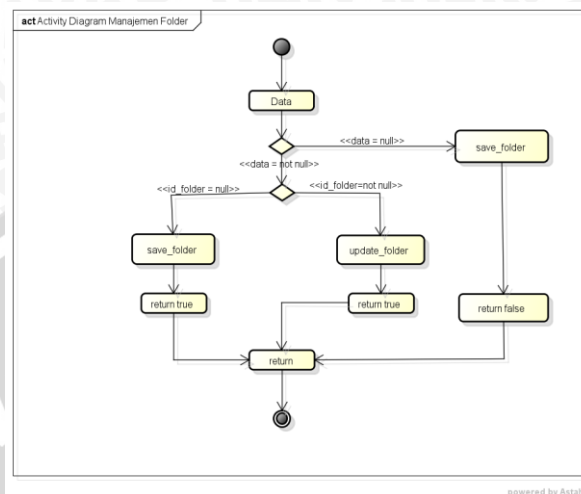
Symbol	Object Method
A	Delete
B	Hapus
C	Get_tbl_modul_section
D	Return
E	Check_section
F	Delete_section
G	Return
H	Delete_modul
I	Return
J	Return

4.2 Activity diagram

Dalam proses pembuatan *Activity diagram*, akan dibuat 7 *activity diagram* dari 7 fitur yang ada PDTT-UB . Setelah jadi *activity diagram*, proses yang yang

ada di dalam *sequence diagram* akan di buat titik point dan diberi symbol yang dikumpulkan dalam *Method Activity Table*

4.2.1 Activity diagram Manajemen Folder



Gambar 4.8 Activity diagram Manajemen Folder

Fitur pertama dari aplikasi PDTT-UB adalah Manajemen Folder. Manajemen folder berisi proses penambahan folder dan pengubahan folder yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *activity diagram* dari fitur manajemen folder yang dapat dilihat pada gambar 4.8.

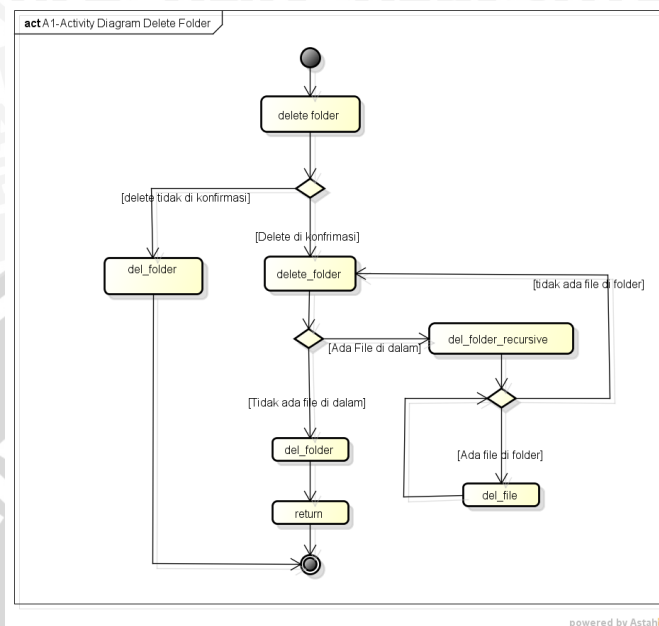
Dari *activity diagram* pada gambar 4.8 terdapat 14 proses yang berlangsung. Dari 14 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.8.

Tabel 4.8 Method Activity Table Delete Modul

Symbol	Object Method
A	Input Data
B	Check input data
C	Data = null
D	Data = not null
E	Check folder id
F	Id_folder = null
G	Id_folder = not null
H	Save_folder
I	Update_folder
J	Return true
K	Return true
L	Save_folder
M	Return false
N	Return

4.2.2 Activity diagram Delete Folder

Berikut merupakan *activity diagram* dari fitur delete folder



Gambar 4.9 Activity diagram Delete Folder

Fitur kedua dari aplikasi PDTT-UB adalah Delete Folder. Delete folder berisi proses menghapus folder yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *activity diagram* dari fitur delete folder yang dapat dilihat pada gambar 4.9 .

Tabel 4.9 Method Activity Table Delete Modul

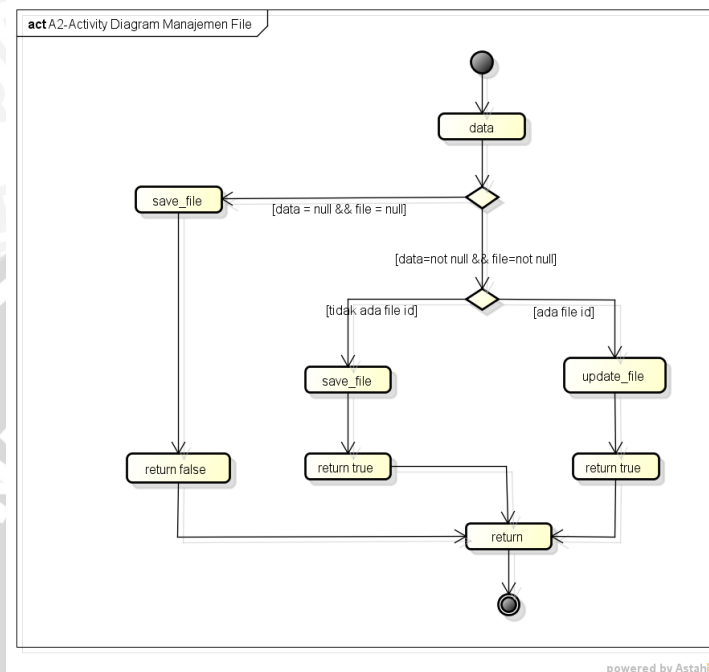
Symbol	Object Method
A	Delete Folder
B	Check konfirmasi
C	Delete di konfirmasi
D	Delete tidak di konfirmasi
E	Delete_folder
F	Check ada file di dalam folder
G	Ada file di dalam
H	Tidak ada file di dalam
I	Del_folder
J	Del_folder_recursive
K	Check masih ada file di dalam folder
L	Ada file di folder
M	Tidak ada file di folder
N	Return

Dari *activity diagram* pada gambar 4.9 terdapat 14 proses yang berlangsung. Dari 14 proses tersebut masing-masing diberi tanda atau symbol yang akan

dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.9.

4.2.3 Activity diagram Manajemen File

Berikut merupakan *activity diagram* dari fitur manajemen file



Gambar 4.10 Activity diagram Manajemen File

Fitur ketiga dari aplikasi PDTT-UB adalah Manajemen File. Manajemen file berisi proses penambahan dan pengubahan file yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *activity diagram* dari fitur manajemen file yang dapat dilihat pada gambar 4.10 .

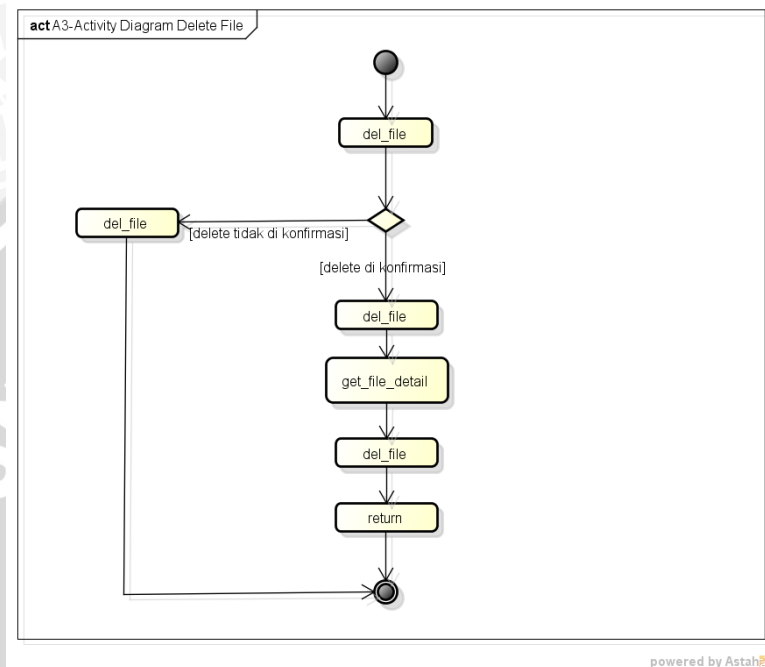
Tabel 4.10 Method Activity Table Delete Modul

Symbol	Object Method
A	Input Data
B	Check input data
C	Data null && file = null
D	Data = not null && file = not null
E	Check file id
F	Tidak ada file id
G	Ada file id
H	Save_file
I	Update_file
J	Return true
K	Return true
L	Save_file
M	Return false
N	Return

Dari *activity diagram* pada gambar 4.10 terdapat 14 proses yang berlangsung. Dari 14 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.10.

4.2.4 Activity diagram Delete File

Berikut merupakan *activity diagram* dari fitur delete file



Gambar 4.11 Activity diagram Delete File

Fitur keempat dari aplikasi PDTT-UB adalah Delete File. Delete file berisi proses penambahan dan pengubahan file yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *activity diagram* dari fitur delete file yang dapat dilihat pada gambar 4.11 .

Tabel 4.11 Method Activity Table Delete Modul

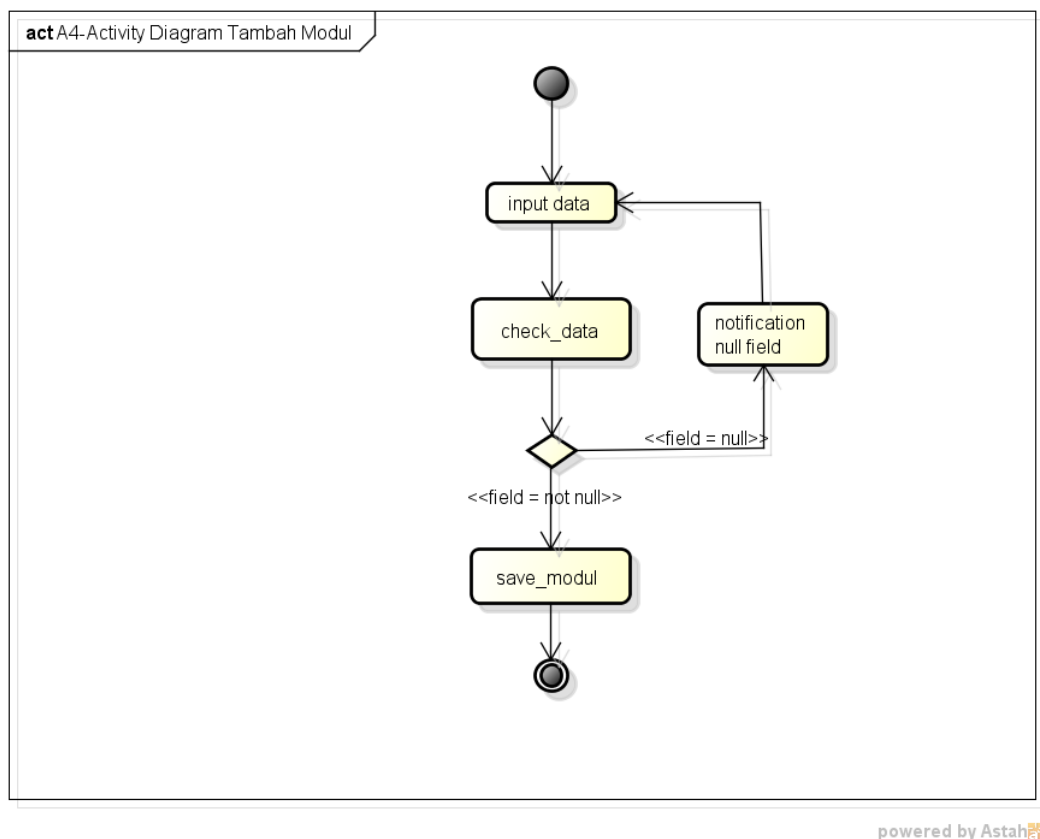
Symbol	Object Method
A	Del_file
B	Check Konfirmasi
C	Delete tidak di konirmasi
D	Delete di konfirmasi
E	Del_file
F	Get_file_detail
G	Del_file
H	Del_file
I	Return

Dari *activity diagram* pada gambar 4.11 terdapat 9 proses yang berlangsung. Dari 9 proses tersebut masing-masing diberi tanda atau symbol yang akan

dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.11.

4.2.5 Activity diagram Tambah Modul

Berikut merupakan *activity diagram* dari fitur tambah file



Gambar 4.12 Activity diagram Tambah Modul

Fitur kelima dari aplikasi PDTT-UB adalah Tambah Modul. Tambah modul berisi proses penambahan modul yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *activity diagram* dari fitur tambah modul yang dapat dilihat pada gambar 4.12 .

Tabel 4.12 Method Activity Table Delete Modul

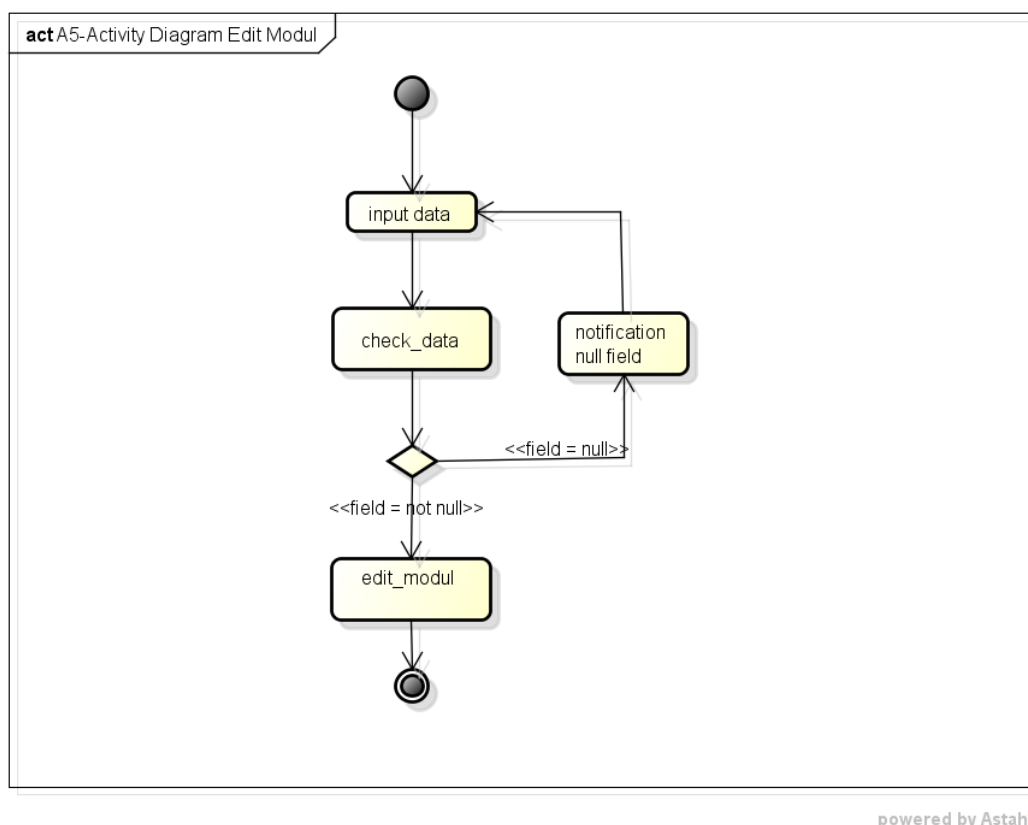
Symbol	Object Method
A	Input_data
B	Check_data
C	Field = not null
D	Field = null
E	Notification null field
F	Save_modul

Dari *activity diagram* pada gambar 4.12 terdapat 6 proses yang berlangsung. Dari 6 proses tersebut masing-masing diberi tanda atau symbol yang akan

dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.12.

4.2.6 Activity diagram Edit Modul

Berikut merupakan *activity diagram* dari fitur edit modul



Gambar 4.13 Activity diagram Edit Modul

Fitur keenam dari aplikasi PDTT-UB adalah Edit Modul. Edit modul berisi proses pengubahan modul yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *activity diagram* dari fitur edit modul yang dapat dilihat pada gambar 4.13 .

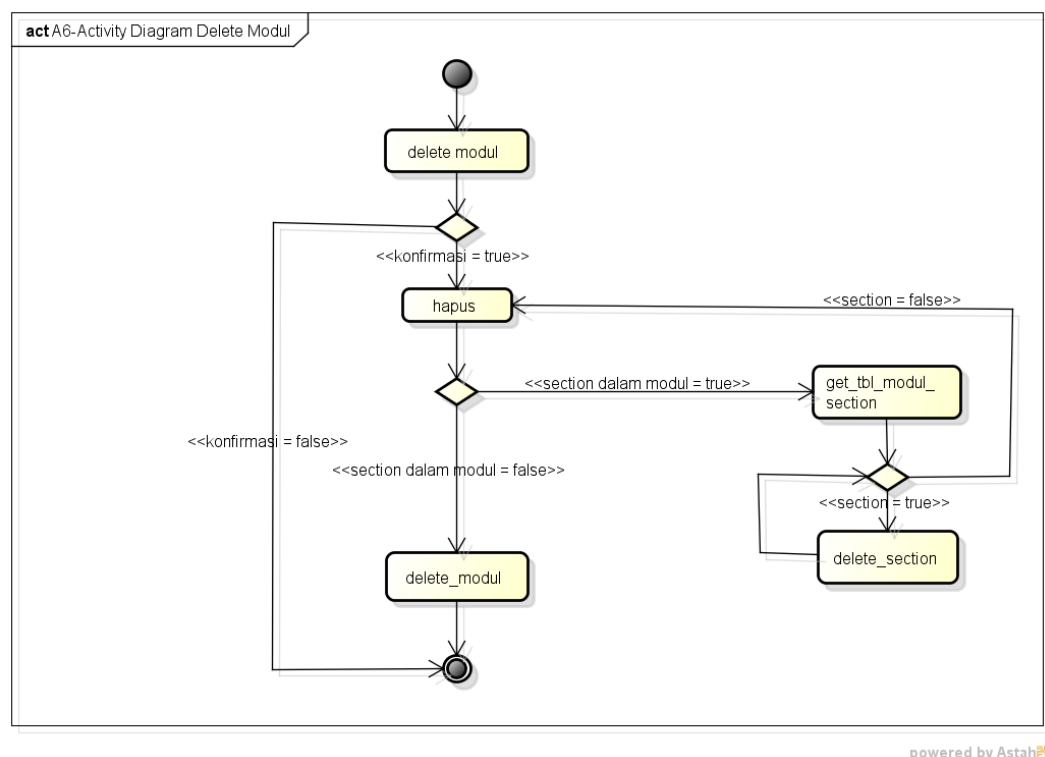
Tabel 4.13 Method Activity Table Delete Modul

Symbol	Object Method
A	Input_data
B	Check_data
C	Field = not null
D	Field = null
E	Notification null field
F	edit_modul

Dari *activity diagram* pada gambar 4.13 terdapat 6 proses yang berlangsung. Dari 6 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.13.

4.2.7 Activity diagram Delete Modul

Berikut merupakan *activity diagram* dari fitur delete modul



Gambar 4.14 Activity diagram Delete Modul

Fitur ketujuh dari aplikasi PDTT-UB adalah Delete Modul. Delete modul berisi proses penghapusan modul yang ada di dalam aplikasi PDTT-UB. Berikut merupakan *activity diagram* dari fitur delete modul yang dapat dilihat pada gambar 4.14 .

Dari *activity diagram* pada gambar 4.14 terdapat 15 proses yang berlangsung. Dari 15 proses tersebut masing-masing diberi tanda atau symbol yang akan dipergunakan sebagai dasar pembuatan Model Flow Graph di fase selanjutnya. Untuk lebih jelasnya dapat dilihat pada tabel 4.14.

Tabel 4.14 Method Activity Table Delete Modul

Symbol	Object Method
A	Delete_modul
B	Check_konfirmasi
C	Konfirmasi = true
D	Konfirmasi = false
E	Hapus
F	Check section dalam modul
G	Section dalam modul = true

Tabel 4.14 Method Activity Table Delete Modul (lanjutan)

H	Section dalam modul = false
I	Get_tbl_modul_section
J	Check section
K	Section = true
L	Section = false
M	Delete_section
N	Delete_modul
O	Return



BAB 5 Hasil dan Pembahasan

Pada Bab ini akan dijelaskan analisis dan hasil dari UML yang ada di bab 4, dalam hal ini adalah *Sequence diagram* dan *Activity diagram* sehingga dapat dibuat menjadi *Test case* menggunakan metode Model Flow Graph (MFG) pada aplikasi PDTT UB. Analisis yang dilakukan dalam bab ini meliputi pembuatan MFG, pembuatan kondisi pengujian, *Test Sequence*, hingga hasilnya menjadi *Test case* dan dilakukan pengujian terhadap aplikasi PDTT-UB sesuai *Test case* yang dihasilkan.

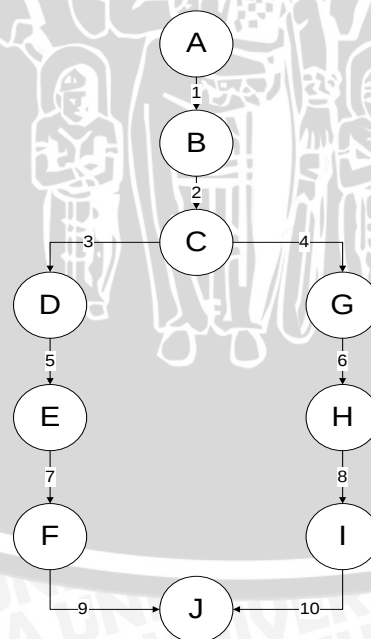
5.1 Analisis Model Flow Graph

Model Flow Graph (MFG) dari manajemen folder di buat berdasarkan dari *Sequence diagram* dan *Activity diagram* dari fitur manajemen folder. MFG ini akan digunakan untuk melihat alur yang lebih sederhana dari proses menambah dan mengubah folder yang ada di aplikasi PDTT UB. Berikut

5.1.1 Model Flow Graph Manajemen Folder

5.1.1.1 Flow Graph *Sequence diagram* Manajemen Folder

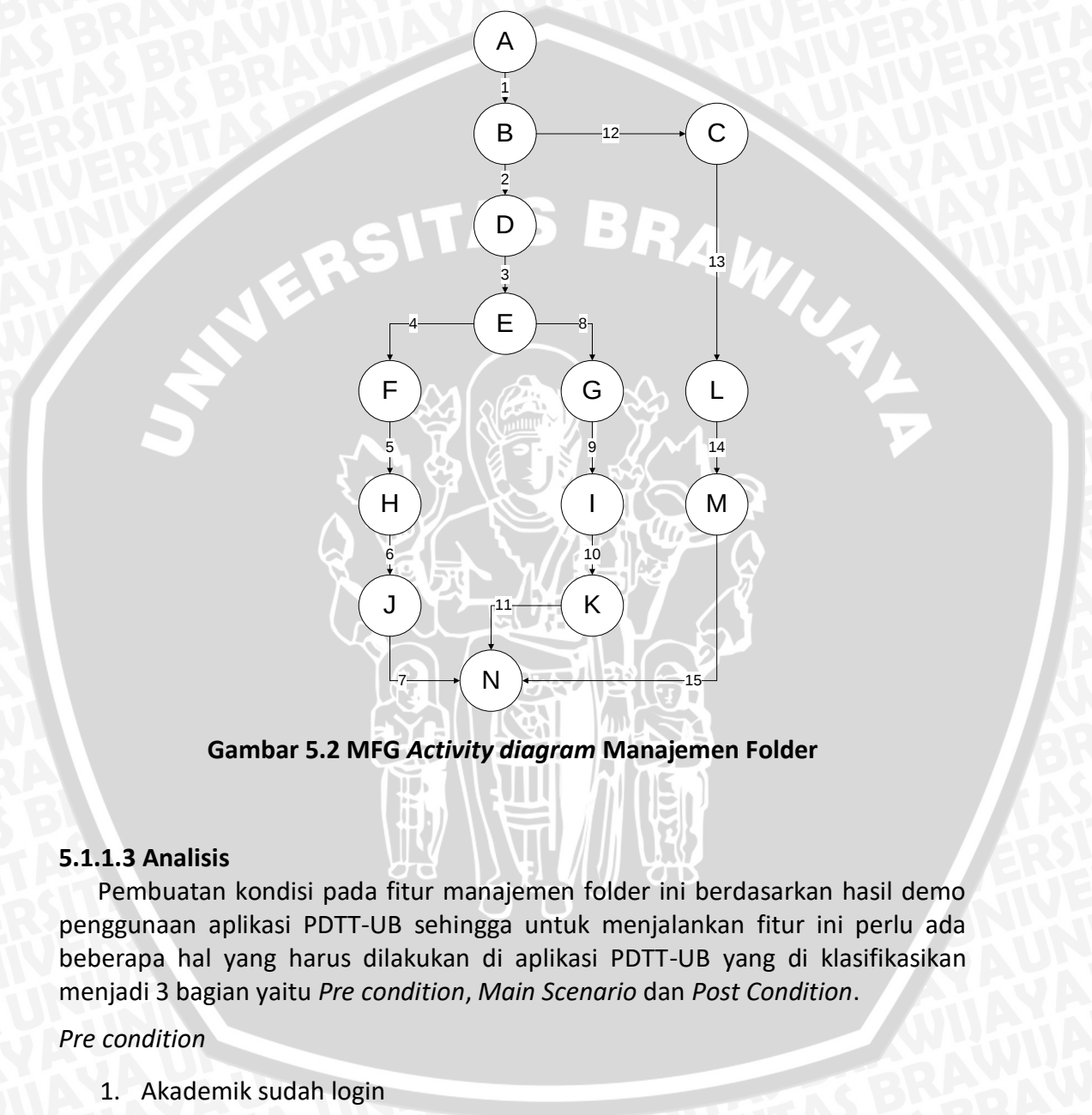
MFG *sequence diagram* manajemen folder dapat dilihat pada gambar 5.1. Method-method dari tabel 4.1 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.1 MFG *Sequence diagram* Manajemen Folder

5.1.1.2 Flow Graph Activity diagram Manajemen Folder

MFG *activity diagram* manajemen folder dapat dilihat pada gambar 5.2. Method-method dari tabel 4.8 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.2 MFG Activity diagram Manajemen Folder

5.1.1.3 Analisis

Pembuatan kondisi pada fitur manajemen folder ini berdasarkan hasil demo penggunaan aplikasi PDTT-UB sehingga untuk menjalankan fitur ini perlu ada beberapa hal yang harus dilakukan di aplikasi PDTT-UB yang di klasifikasikan menjadi 3 bagian yaitu *Pre condition*, *Main Scenario* dan *Post Condition*.

Pre condition

1. Akademik sudah login

Main Scenario

1. Akademik menambah folder
2. Akademik mengubah folder

Post Condition

1. Folder baru berhasil dibuat
2. Folder berhasil diubah
3. Folder gagal di tambah atau di ubah

Test Sequence

1. Test Sequence 1

Edge Label	Message / Activity	Category Partition
1	Input data	Null
2	Validate data	Data = not null
3	Save_folder	ID_folder = null
5	Save_folder	Null
7	Return	Null
9	Return	Null

2. Test Sequence 2

Edge Label	Message / Activity	Category Partition
1	Input data	Null
2	Validate data	Data = not null
4	Save_folder	ID_folder = not null
6	Save_folder	Null
8	Return	Null
10	Return	Null

3. Test Sequence 3

Edge Label	Message / Activity	Category Partition
1	Input Data	Null
12	Validate data	Data = null
13	Return False	Null
14 & 15	Return	Null

Test case Generation

Dari analisis kondisi pengujian dan *Test Sequence* dapat dibuat *Test case* yang hasilnya sebagai berikut :

Test case 1 { input : data not null + id folder = null, output : folder baru berhasil dibuat}

Test case 2 { input : data not null + id folder = not null, output : folder berhasil diubah}

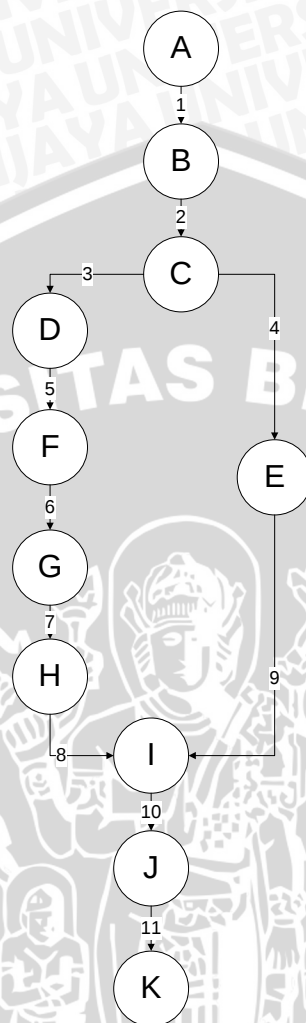
Test case 3 { input : data null, output : folder gagal ditambah atau di ubah}

5.1.2 Model Flow Graph Delete Folder

5.1.2.1 Flow Graph Sequence diagram Delete Folder

MFG *sequence diagram* delete folder dapat dilihat pada gambar 5.3. Method-method dari tabel 4.2 dinotasikan dengan bulatan (node) yang diberi

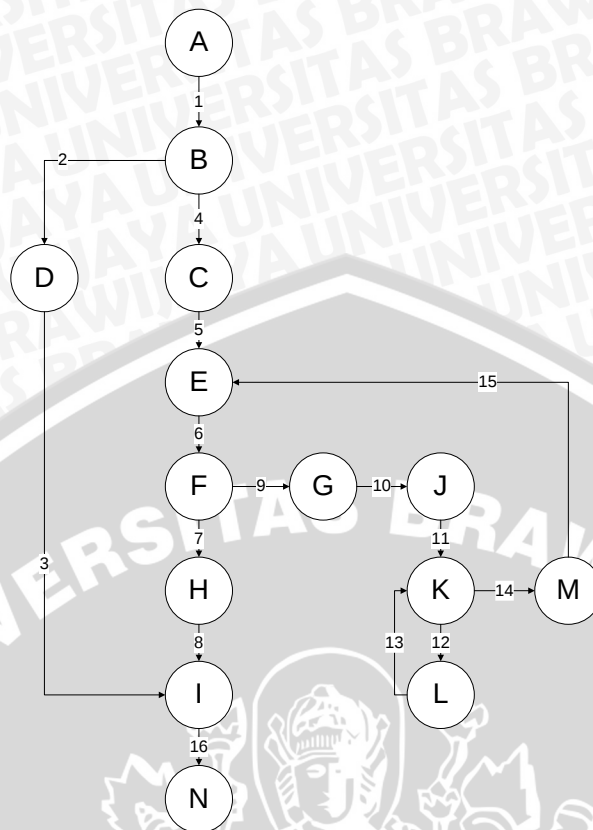
kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.3 MFG Sequence diagram Delete Folder

5.1.2.2 Flow Graph Activity diagram Delete Folder

MFG *activity diagram* delete folder dapat dilihat pada gambar 5.2. Method-method dari tabel 4.8 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.4 MFG Activity diagram Delete Folder

5.1.2.3 Analisis

Pembuatan kondisi pada fitur delete folder ini berdasarkan hasil demo penggunaan aplikasi PDTT-UB sehingga untuk menjalankan fitur ini perlu ada beberapa hal yang harus dilakukan di aplikasi PDTT-UB yang di klasifikasikan menjadi 3 bagian yaitu *Pre condition*, *Main Scenario* dan *Post Condition*.

Pre condition

1. Akademik sudah login

Main Scenario

1. Akademik menghapus folder yang ada file di dalamnya
2. Akademik menghapus folder yang tidak ada file di dalamnya

Post Condition

1. Folder berhasil dihapus
2. Folder beserta file di dalamnya berhasil di hapus
3. Folder gagal dihapus

Test Sequence

1. Test Sequence 1

Edge Label	Message / Activity	Category Partition
1	Delete_folder	Null
2	Check_file	File = null
4 & 9	Del_folder	Null
10	Return	Null
11	Return	Null

2. Test Sequence 2

Edge Label	Message / Activity	Category Partition
1	Delete	Null
2	Check Konfirmasi	Konfirmasi = False
3	Delete tidak di konfirmasi	Null
16	Return	Null

3. Test Sequence 3

Edge Label	Message / Activity	Category Partition
1	Delete	Null
4	Check Konfirmasi	Konfirmasi = True
5	Delete_folder	Null
6 & 9	Check ada file di dalam folder	File di dalam folder = True
10	Del_folder_recursive	Null
11	Check masih ada file di folder	File di dalam folder = True
12	Del_file	Null
13	Check masih ada file di folder	File di dalam folder = False
14	Delete_folder	Null
15	Delete_folder	Null
16	Return	Null

Test case Generation

Dari analisis kondisi pengujian dan *Test Sequence* dapat dibuat *Test case* yang hasilnya sebagai berikut :

Test case 1 { input : folder tidak ada file didalamnya, output : Folder berhasil dihapus }

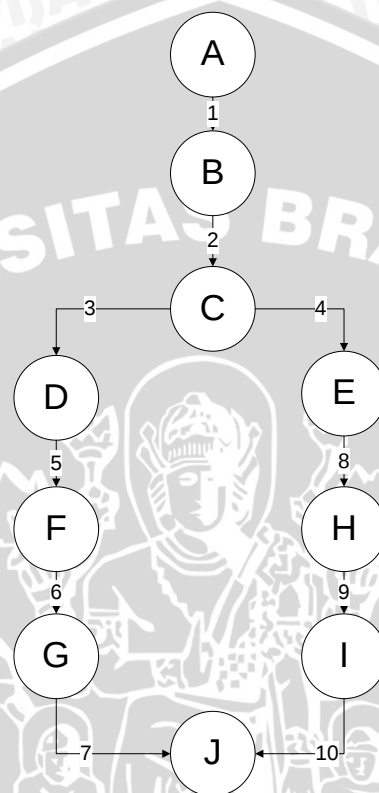
Test case 2 { input : folder tidak ada file dalamnya atau ada file didalamnya + konfirmasi = false , output : Folder gagal dihapus }

Test case 3 { input : folder ada file di dalamnya + konfirmasi = true , output : folder beserta file di dalamnya berhasil dihapus }

5.1.3 Model Flow Graph Manajemen File

5.1.3.1 Flow Graph Sequence diagram Manajemen File

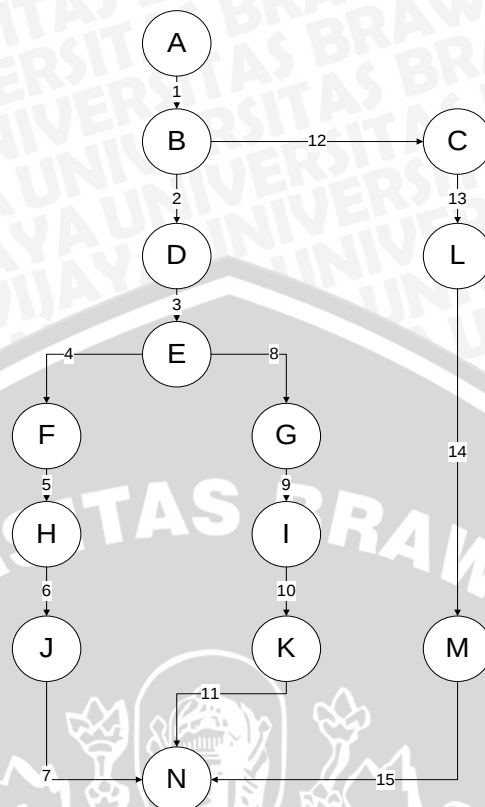
MFG sequence manajemen file dapat dilihat pada gambar 5.5. Method-method dari tabel 4.3 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.5 MFG Sequence diagram Manajemen File

5.1.3.2 Flow Graph Activity diagram Manajemen File

MFG activity diagram manajemen file dapat dilihat pada gambar 5.6. Method-method dari tabel 4.10 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.6 MFG Activity diagram Manajemen File

5.1.3.3 Analisis

Pembuatan kondisi pada fitur manajemen file ini berdasarkan hasil demo penggunaan aplikasi PDTT-UB sehingga untuk menjalankan fitur ini perlu ada beberapa hal yang harus dilakukan di aplikasi PDTT-UB yang di klasifikasikan menjadi 3 bagian yaitu *Pre condition*, *Main Scenario* dan *Post Condition*.

Pre condition

1. Akademik sudah login

Main Scenario

1. Akademik menambah file
2. Akademik mengubah file

Post Condition

1. File baru berhasil dibuat
2. File berhasil diubah
3. File gagal di tambah atau diubah

Test Sequence

1. Test Sequence 1

Edge Label	Message / Activity	Category Partition
1	Input data	Null
2	Validate data	Data = not null && file = not null
3	Save_file	Input tidak ada file id
5	Save_file	Null
6	Return	Null
7	Return	Null

2. Test Sequence 2

Edge Label	Message / Activity	Category Partition
1	Input data	Null
2	Validate data	Data = not null && file = not null
4	Update_file	Input ada file id
8	Update_file	Null
9	Return	Null
10	Return	Null

3. Test Sequence 3

Edge Label	Message / Activity	Category Partition
1	Input data	Null
12	Validate data	Data = null && file = null
13	Save_file	Null
14	Return false	Null
15	Return	Null

4. Test Sequence 4

Edge Label	Message / Activity	Category Partition
1	Input data	Null
12	Validate data	Data = not null && file = null
13	Save_file	Null
14	Return false	Null
15	Return	Null

5. Test Sequence 5

Edge Label	Message / Activity	Category Partition
1	Input data	Null
12	Validate data	Data = null && file = not null
13	Save_file	Null
14	Return false	Null
15	Return	Null

Test case Generation

Dari analisis kondisi pengujian dan *Test Sequence* dapat dibuat *Test case* yang hasilnya sebagai berikut :

Test case 1 { input : data=not null + file=not null + tidak ada file id, output : file baru berhasil dibuat }

Test case 2 { input : data=not null + file=not null + ada file id, output : file berhasil diubah }

Test case 3 { input : data=null + file = null, output : file gagal di tambah maupun di ubah }

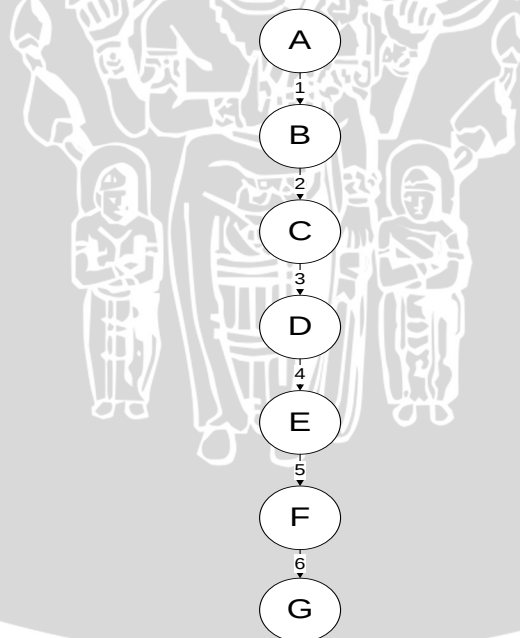
Test case 4 { input : data=not null + file= null, output : file gagal di tambah maupun di ubah }

Test case 5 { input : data=null + file= not null, output : file gagal di tambah maupun di ubah }

5.1.4 Model Flow Graph Delete File

5.1.4.1 Flow Graph Sequence diagram Delete File

MFG sequence delete file dapat dilihat pada gambar 5.7. Method-method dari tabel 4.4 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric

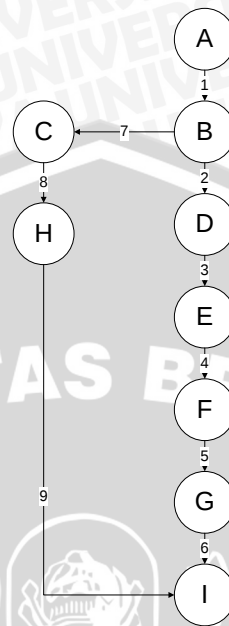


Gambar 5.7 MFG Sequence diagram Delete File

5.1.4.2 Flow Graph Activity diagram Delete File

MFG activity diagram delete file dapat dilihat pada gambar 5.8. Method-method dari tabel 4.11 dinotasikan dengan bulatan (node) yang diberi kode

alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.8 MFG Activity diagram Delete File

5.1.4.3 Analisis

Pembuatan kondisi pada fitur delete file ini berdasarkan hasil demo penggunaan aplikasi PDTT-UB sehingga untuk menjalankan fitur ini perlu ada beberapa hal yang harus dilakukan di aplikasi PDTT-UB yang di klasifikasikan menjadi 3 bagian yaitu *Pre condition*, *Main Scenario* dan *Post Condition*.

Pre condition

1. Akademik sudah login

Main Scenario

1. Akademik menghapus file

Post Condition

1. File berhasil di hapus
2. File gagal dihapus

Test Sequence

1. Test Sequence 1

Edge Label	Message / Acitivity	Category Partition
1	Del_file	Null
2	Check Konfirmasi	Konfirmasi = true
3	Del_file	Null

4	Get_file_detail	Null
5	Del_file	Null
6	Return	Return = true

2. Test Sequence 2

Edge Label	Message / Activity	Category Partition
1	Del_file	Null
7	Check Konfirmasi	Konfirmasi = false
8	Del_file	Null
9	Return	Return = false

Test case Generation

Dari analisis kondisi pengujian dan *Test Sequence* dapat dibuat *Test case* yang hasilnya sebagai berikut :

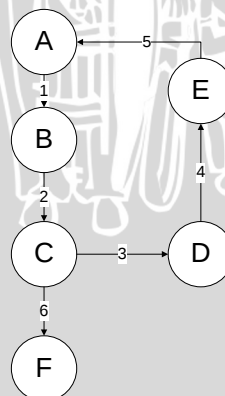
Test case 1 { input : file + konfirmasi = true, output : file berhasil di hapus }

Test case 2 { input : file + konfirmasi = false, output : file gagal dihapus }

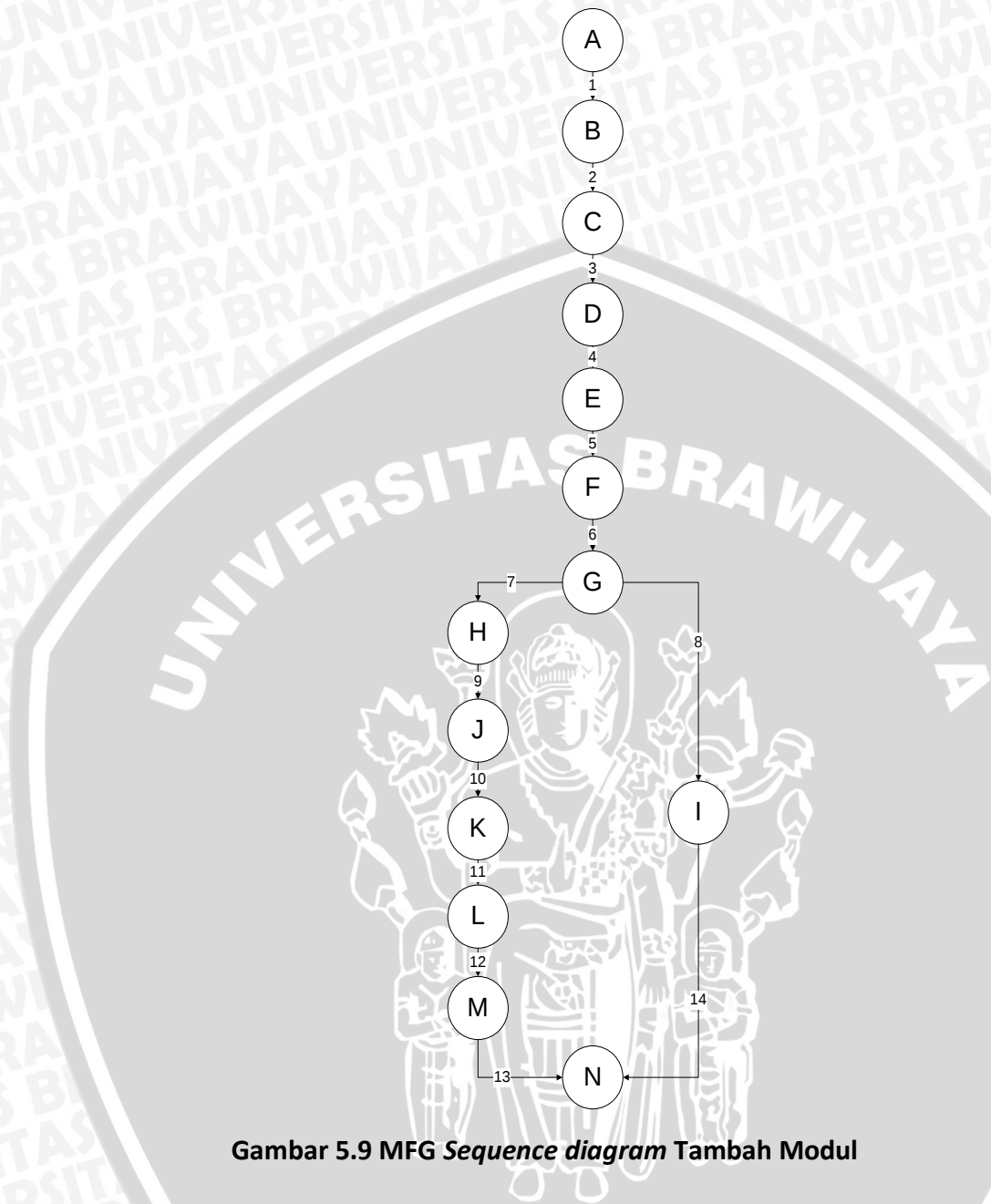
5.1.5 Model Flow Graph Tambah Modul

5.1.5.1 Flow Graph Sequence diagram Tambah Modul

MFG sequence tambah modul dapat dilihat pada gambar 5.9. Method-method dari tabel 4.5 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.10 MFG Activity diagram Tambah Modul



Gambar 5.9 MFG Sequence diagram Tambah Modul

5.1.5.2 Flow Graph Activity diagram Tambah Modul

MFG activity diagram tambah modul dapat dilihat pada gambar 5.10. Method-method dari tabel 4.12 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric

5.1.5.3 Analisis

Pembuatan kondisi pada fitur tambah modul ini berdasarkan hasil demo penggunaan aplikasi PDTT-UB sehingga untuk menjalankan fitur ini perlu ada

beberapa hal yang harus dilakukan di aplikasi PDTT-UB yang di klasifikasikan menjadi 3 bagian yaitu *Pre condition*, *Main Scenario* dan *Post Condition*.

Pre condition

1. Akademik Sudah login

Main Scenario

1. Akademik menambah modul

Post Condition

1. Modul berhasil dibuat
2. Modul berhasil dibuat dengan pengarang baru
3. Muncul notifikasi field belum diisi

Test Sequence

1. *Test Sequence 1*

Edge Label	Message / Activity	Category Partition
1	Input_data	Null
2	Check_data	Field = not null
6	Save_modul	Null

2. *Test Sequence 2*

Edge Label	Message / Activity	Category Partition
1	Input_data	Null
2	Check_data	Field = null
3 & 4	Notification null field	Null
5	Back to input_data	Null

3. *Test Sequence 3*

Edge Label	Message / Activity	Category Partition
1	Input_data	Null
2	Save_modul	Field = not_null
3	Get_modul_id	Null
4 & 5	Save_modul	Null
6 & 8	Check_pengarang_id	Id_pengarang = not null
14	Return	null

4. *Test Sequence 4*

Edge Label	Message / Activity	Category Partition
1	Input_data	Null
2	Save_modul	Field = not_null
3	Get_modul_id	Null
4 & 5	Save_modul	Null
6 & 7	Check_pengaran_id	Id_pengarang = null
9 & 10	Get_pengarang_id	Pilih pengarang
11 & 12	Save_pengarang	null
13	Return	null

Test case Generation

Dari analisis kondisi pengujian dan *Test Sequence* dapat dibuat *Test case* yang hasilnya sebagai berikut :

Test case 1 { input : data = field not null, output : modul berhasil dibuat }

Test case 2 { input : data = field null, output : muncul notifikasi field belum diisi }

Test case 3 { input : data = field not null + pengarang = not null, output : modul berhasil dibuat }

Test case 4 { input : data = field not null + pengarang = null, output : modul berhasil dibuat beserta pengarang baru}

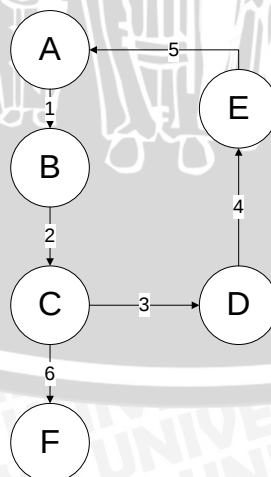
5.1.6 Model Flow Graph Edit Modul

5.1.6.1 Flow Graph Sequence diagram Edit Modul

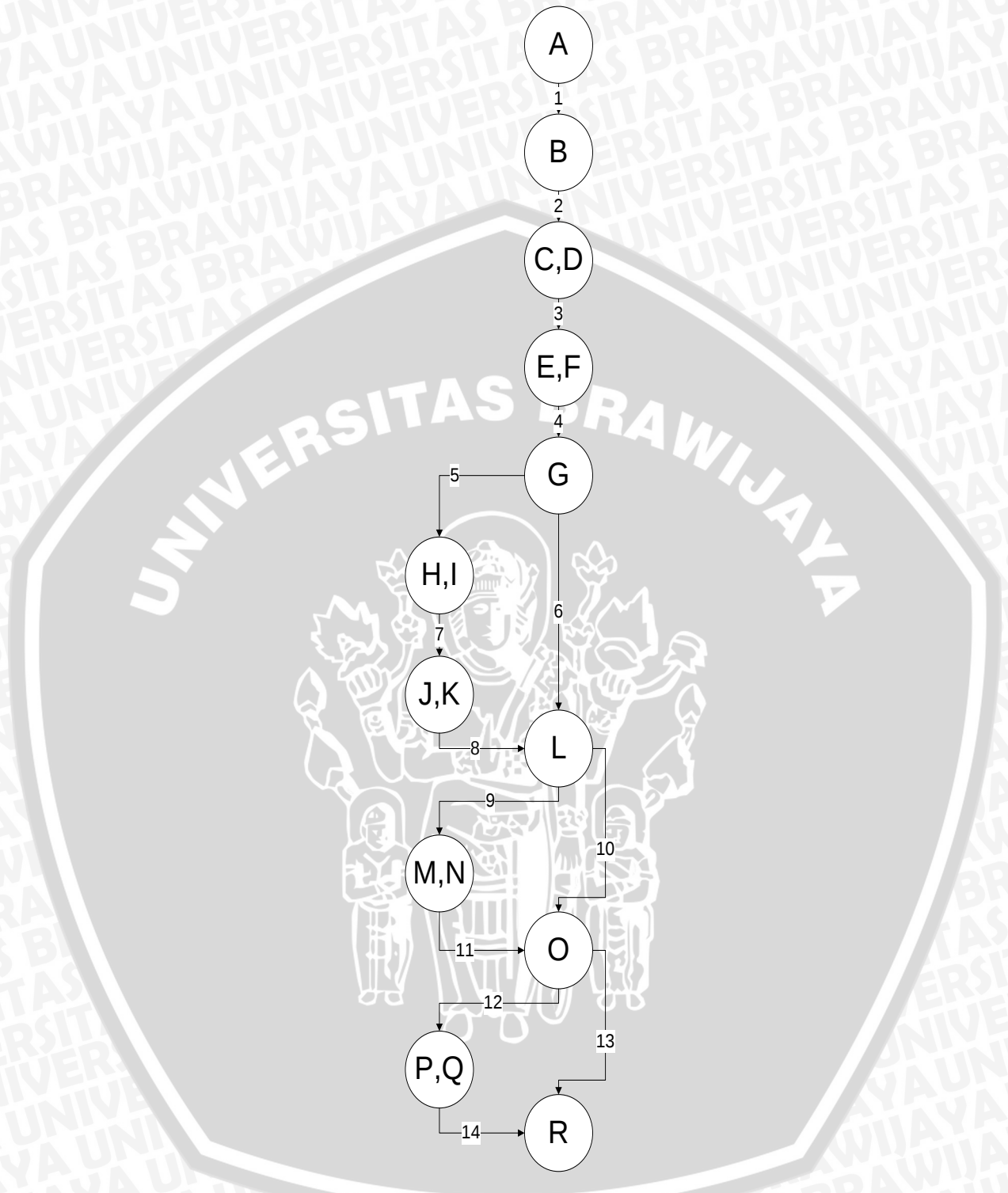
MFG sequence edit modul dapat dilihat pada gambar 5.12. Method-method dari tabel 4.6 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric

5.1.6.2 Flow Graph Activity diagram Edit Modul

MFG activity diagram edit modul dapat dilihat pada gambar 5.11. Method-method dari tabel 4.13 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.11 MFG Activity diagram Edit Modul



Gambar 5.12 MFG Sequence diagram Edit Modul

5.1.6.3 Analisis

Pembuatan kondisi pada fitur edit modul ini berdasarkan hasil demo penggunaan aplikasi PDTT-UB sehingga untuk menjalankan fitur ini perlu ada beberapa hal yang harus dilakukan di aplikasi PDTT-UB yang di klasifikasikan menjadi 3 bagian yaitu *Pre condition*, *Main Scenario* dan *Post Condition*.

Pre condition

1. Akademik sudah login

Main Scenario

1. Akademik mengubah modul

Post Condition

1. Modul berhasil ubah
2. Modul berhasil diubah dengan pengarang baru
3. Modul berhasil diubah dengan *cover* baru
4. Modul berhasil diubah dengan file baru
5. Modul berhasil diubah dengan pengarang dan *cover* baru
6. Modul berhasil diubah dengan pengarang dan file baru
7. Modul berhasil diubah dengan pengarang, *cover*, dan file baru
8. Modul berhasil diubah dengan *cover* dan file baru
9. Muncul notifikasi field belum diisi

Test Sequence

1. *Test Sequence 1*

Edge Label	Message / Activity	Category Partition
1	Input_data	Null
2	Check_data	Field = not null
6	Edit_modul	Null

2. *Test Sequence 2*

Edge Label	Message / Activity	Category Partition
1	Input_data	Null
2	Check_data	Field = null
3 & 4	Notification null field	Null
5	Back to input_data	Null

3. *Test Sequence 3*

Edge Label	Message / Activity	Category Partition
1	Input_data	Field = not null
2	Deskrip_modul_id	Null
3	Edit_modul	Null
4	Check_pengarang_id	Id_pengarang = not null
6	Check_update_cover	Cover = not null
10	Check_update_file	File = not null
13	Return	Null

4. *Test Sequence 4*

Edge Label	Message / Activity	Category Partition
1	Input_data	Field = not null
2	Deskrip_modul_id	Null
3	Edit_modul	Null
4	Check_pengarang_id	Id_pengarang = null

5	Get_pengarang_id	Pilih pengarang
7	Save_pengarang	Null
8	Check_update_cover	Cover = not null
10	Check_update_file	File = not null
13	Return	Null

5. Test Sequence 5

Edge Label	Message / Activity	Category Partition
1	Input_data	Field = not null
2	Deskrip_modul_id	Null
3	Edit_modul	Null
4	Check_pengarang_id	Id_pengarang = null
5	Get_pengarang_id	Pilih pengarang
7	Save_pengarang	Null
8	Check_update_cover	Cover = null
9	Update_modul_file	Pilih file
11	Check_update_file	File = not null
13	Return	Null

6. Test Sequence 6

Edge Label	Message / Activity	Category Partition
1	Input_data	Field = not null
2	Deskrip_modul_id	Null
3	Edit_modul	Null
4	Check_pengarang_id	Id_pengarang = null
5	Get_pengarang_id	Pilih pengarang
7	Save_pengarang	Null
8	Check_update_cover	Cover = null
9	Update_modul_file	Pilih cover
11	Check_update_file	File = null
12	Update_modul_file	Pilih file
14	Return	Null

7. Test Sequence 7

Edge Label	Message / Activity	Category Partition
1	Input_data	Field = not null
2	Deskrip_modul_id	Null
3	Edit_modul	Null
4	Check_pengarang_id	Id_pengarang = null
5	Get_pengarang_id	Pilih pengarang
7	Save_pengarang	Null
8	Check_update_cover	Cover = not null
10	Check_update_file	File = null
12	Update_modul_file	Pilih file
14	Return	Null

8. Test Sequence 8

Edge Label	Message / Activity	Category Partition
1	Input_data	Field = not null
2	Deskrip_modul_id	Null
3	Edit_modul	Null

4	Check_pengarang_id	Id_pengarang = not null
6	Check_update_cover	Cover = null
9	Update_module_file	Pilih cover
11	Check_modul_file	File = not null
13	Return	Null

9. *Test Sequence 9*

Edge Label	Message / Activity	Category Partition
1	Input_data	Field = not null
2	Deskrip_modul_id	Null
3	Edit_modul	Null
4	Check_pengarang_id	Id_pengarang = not null
6	Check_update_cover	Cover = null
9	Update_module_file	Pilih cover
11	Check_modul_file	File = null
12	Update_modul_file	Pilih file
13	Return	Return

10. *Test Sequence 10*

Edge Label	Message / Activity	Category Partition
1	Input_data	Field = not null
2	Deskrip_modul_id	Null
3	Edit_modul	Null
4	Check_pengarang_id	Id_pengarang = not null
6	Check_update_cover	Cover = not null
10	Check_modul_file	File = null
12	Update_modul_file	Pilih file
14	Return	Return

Test case Generation

Dari analisis kondisi pengujian dan *Test Sequence* dapat dibuat *Test case* yang hasilnya sebagai berikut :

Test case 1 { input : data = field not null, output : modul berhasil diubah }

Test case 2 { input : data = field null, output : muncul notifikasi field belum diisi }

Test case 3 { input : data = field not null + pengarang = not null + cover = not null + file = not null, output : modul berhasil diubah }

Test case 4 { input : data = field not null + pengarang = null + cover = not null + file = not null , output : modul berhasil diubah beserta pengarang baru }

Test case 5 { input : data = field not null + pengarang = null + cover = null + file = not null , output : modul berhasil diubah dengan pengarang baru dan cover baru }

Test case 6 { input : data = field not null + pengarang = null + cover = null + file = null , output : modul berhasil diubah dengan pengarang baru, cover baru dan file baru }

Test case 7 { input : data = field not null + pengarang = null + cover = not null + file = null , output : modul berhasil diubah dengan pengarang baru dan file baru }

Test case 8 { input : data = field not null + pengarang = not null + cover = null + file = not null , output : modul berhasil diubah dengan cover baru}

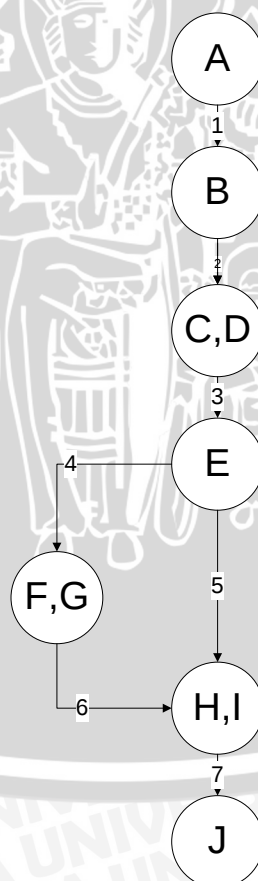
Test case 9 { input : data = field not null + pengarang = not null + cover = null + file = null , output : modul berhasil diubah dengan cover dan file baru }

Test case 10 { input : data = field not null + pengarang = not null + cover = not null + file = null , output : modul berhasil diubah dengan file baru }

5.1.7 Model Flow Graph Delete Modul

5.1.7.1 Flow Graph Sequence diagram Delete Modul

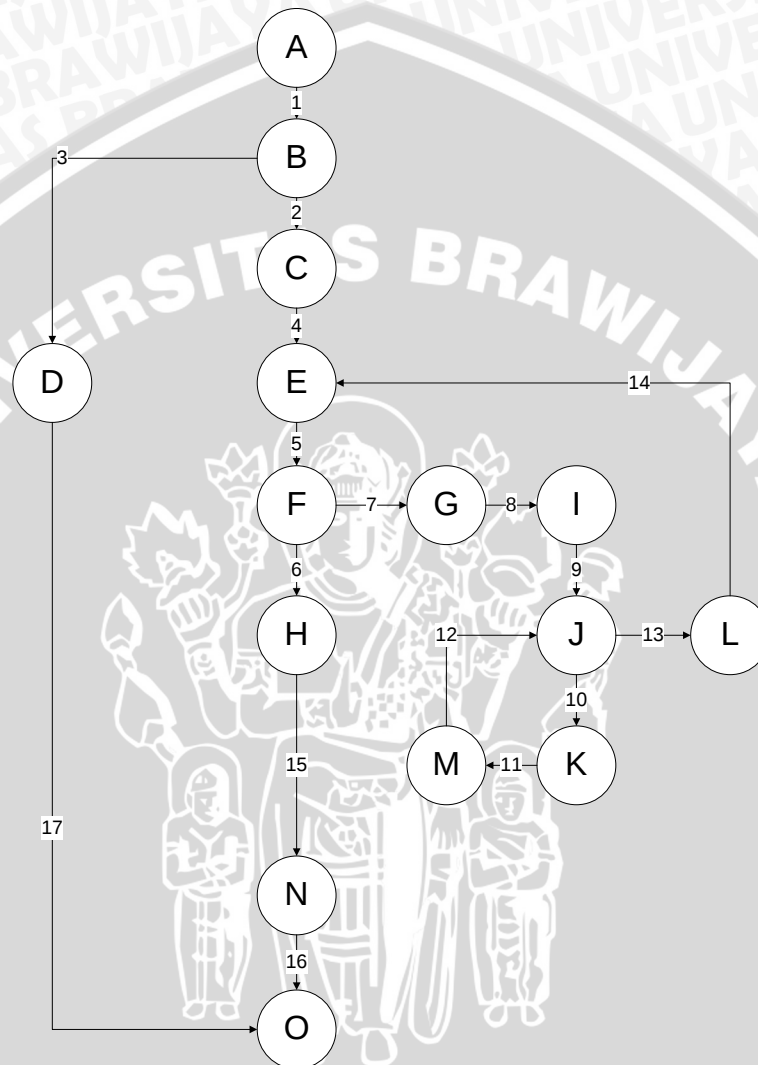
MFG sequence delete modul dapat dilihat pada gambar 5.13. Method-method dari tabel 4.7 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.13 MFG Sequence diagram Delete Modul

5.1.7.2 Flow Graph Activity diagram Delete Modul

MFG *activity diagram* edit modul dapat dilihat pada gambar 5.12. Method-method dari tabel 4.13 dinotasikan dengan bulatan (node) yang diberi kode alphabet sedangkan proses yang menghubungkan method 1 dengan method yang lain dalam proses menambah dan mengubah folder dinotasikan dengan garis yang diberi kode numeric



Gambar 5.14 MFG Activity diagram Delete Modul

5.1.7.3 Analisis

Pembuatan kondisi pada fitur delete modul ini berdasarkan hasil demo penggunaan aplikasi PDTT-UB sehingga untuk menjalankan fitur ini perlu ada beberapa hal yang harus dilakukan di aplikasi PDTT-UB yang di klasifikasikan menjadi 3 bagian yaitu *Pre condition*, *Main Scenario* dan *Post Condition*.

Pre condition

1. Akademik sudah login

Main Scenario

1. Akademik menghapus Modul

Post Condition

1. Modul berhasil di hapus
2. Modul beserta materi didalamnya berhasil dihapus
3. Modul gagal dihapus

Test Sequence

1. Test Sequence 1

Edge Label	Message / Activity	Category Partition
1	Delete modul	Null
3	Check Konfirmasi	Konfirmasi = false
17	Return	Null

2. Test Sequence 2

Edge Label	Message / Activity	Category Partition
1	Del_file	Null
2	Check Konfirmasi	Konfirmasi = true
4	Delete	Null
5	Check_section_dalam_modul	Section dalam modul = false
6 & 15	Delete Modul	Null
16	Return	Null

3. Test Sequence 3

Edge Label	Message / Activity	Category Partition
1	Del_file	Null
2	Check Konfirmasi	Konfirmasi = true
4	Delete	Null
5	Check_section_dalam_modul	Section dalam modul = true
7 & 8	Get_tbl_modul_section	Null
9 & 10	Check_section	Section = true
11	Delete_section	Null
12 & 13	Check_section	Section = false
14	Delete	Null
5	Check_section_dalam_modul	Section dalam modul = false
6 & 15	Delete Modul	Null
16	Return	Null

Test case Generation

Dari analisis kondisi pengujian dan *Test Sequence* dapat dibuat *Test case* yang hasilnya sebagai berikut :

Test case 1 { input : modul tidak ada materi didalamnya + konfirmasi = false, output : modul gagal dihapus}

Test case 2 { input : modul tidak ada materi didalamnya + konfirmasi = true, output : modul berhasil dihapus }

Test case 3 { input : modul ada materi didalamnya + konfirmasi = true, output : modul beserta materi didalamnya berhasil dihapus }

5.2 Hasil *Test case*

Setelah masing-masing dari flow graph di generate menjadi beberapa *Test case* sesuai hasil analisisnya, untuk mempermudah pembacaan, *Test case* akan dikumpulkan menjadi sebuah tabel pada tabel 5.2 dimana isinya dari fitur Manajemen folder terdapat 3 *Test case*, Delete folder terdapat 3 *Test case*, Manajemen file terdapat 5 *Test case*, delete file terdapat 2 *Test case*, Tambah modul 4 *Test case*, edit modul terdapat 10 *Test case* dan yang terakhir delete modul terdapat 3 testcase. Total dari 7 fitur yang ada pada modul admin aplikasi PDTT-UB terdapat 30 *Test case*.



Tabel 5.1 Tabel Test case Aplikasi PDDT UB

No	Fitur	Test case ID	Input
1	Manajemen Folder	TC-MFO-01	data : not null
			id_folder : null
		TC-MFO-02	data : not null
			id_folder : not null
2	Delete Folder	TC-MFO-03	data : null
		TC-DFO-01	folder tidak ada file didalam
		TC-DFO-02	folder tidak ada file didalamnya atau ada file didalamnya
			konfirmasi : false
		TC-DFO-03	folder ada file di dalamnya
			konfirmasi : true
3	Manajemen File	TC-MFI-01	data : not null
			file : not null
			id_file : null
		TC-MFI-02	data : not null
			file : not null
			id_file : not null
		TC-MFI-03	data : null
			file : not null
		TC-MFI-04	data : not null

				file : null
			TC-MFI-05	data : null
				file : null
4	Delete File		TC-DFI-01	file : true
				konfirmasi : true
			TC-DFI-02	file : true
				konfirmasi : false
5	Tambah Modul		TC-TMO-01	data : field not null
			TC-TMO-02	data : field null
			TC-TMO-03	data : field not null
				pengarang : not null
			TC-TMO-04	data : field not null
				pengarang : null
6	Edit Modul		TC-EMO-01	data : field not null
			TC-EMO-02	data : field null
			TC-EMO-03	data : field not null
				pengarang : not null
				cover : not null
				file : not null
			TC-EMO-04	data : field not null
				pengarang : null
				cover : not null
				file : not null

		TC-EMO-05	data : field not null
			pengarang : null
			cover : null
			file : not null
		TC-EMO-06	data : field not null
			pengarang : null
			cover : null
			file : null
		TC-EMO-07	data : field not null
			pengarang : null
			cover : not null
			file : null
		TC-EMO-08	data : field not null
			pengarang : not null
			cover : null
			file : not null
		TC-EMO-09	data : field not null
			pengarang : not null
			cover : null
			file : null
		TC-EMO-10	data : field not null
			pengarang : not null
			cover : not null
			file : null
7	Delete Modul	TC-DMO-01	modul tidak ada materi didalamnya
			konfirmasi : false
		TC-DMO-02	modul tidak ada materi didalamnya

		konfirmasi : true
	TC-DMO-03	modul ada materi didalamnya
		konfirmasi : true



5.3 Pelaksanaan Pengujian

Setelah mendapatkan tabel *Test case*, maka proses selanjutnya yang dilakukan adalah pelaksanaan pengujian. Dalam pelaksanaan pengujian ini, langkah-langkah yang dilakukan berdasar tabel *Test case* yang sudah dibuat. Nantinya hasil pengujian akan di bagi menjadi 2 bagian yaitu Valid dan Tidak Valid.

Valid : Hasil pengujian (realization) dengan apa yang diharapkan (expectation) sesuai

Tidak Valid : Hasil pengujian (realization) dengan apa yang diharapkan (expectation) tidak sesuai dan di beri warna merah

5.4 Analisis Hasil Uji

Berdasarkan pelaksanaan pengujian yang terkumpul pada tabel 5.2 pelaksanaan pengujian aplikasi PDTT-UB yang di dalamnya terdapat 30 fitur, hasilnya adalah ada 18 *test case* yang mendapat label Valid dan 12 *test case* label tidak valid.

Test case pertama yang tidak valid adalah pada fitur delete folder dengan nomor *test case* TC-DFO-01 dimana yang diuji adalah menghapus folder yang tidak ada file didalamnya. *Test case* tersebut tidak valid karena pada saat proses penghapusan folder, notifikasi yang muncul adalah “Apakah anda ingin menghapus folder ini beserta semua isinya?” yang seharusnya notifikasi tersebut muncul pada saat menghapus folder yang ada file didalamnya.

Test case kedua yang tidak valid adalah pada fitur manajemen file dengan nomor *test case* TC-MFI-01 dimana yang diuji adalah menambah file. *Test case* tersebut tidak valid Karena pada saat input file yang tidak berformat PDF (.PDF), maka tambah file akan gagal yang seharusnya file yang diinput bias berformat selain PDF(.PDF)

Test case ketiga yang tidak valid adalah pada fitur manajemen file dengan nomor *test case* TC-MFI- 02 dimana yang diuji adalah mengubah file yang sudah pernah diinputkan. *Test case* tersebut tidak valid Karena pada saat masuk form edit file, file yang sudah ada sebelumnya otomatis terhapus sehingga harus upload ulang file lagi meskipun hanya ingin mengganti nama dari file tersebut yang seharusnya file tersebut tetap ada di form pada saat menginputkan perubahan pada form edit file.

Test case keempat yang tidak valid adalah pada fitur Delete file dengan nomor *test case* TC-DFI-01 dimana yang diuji adalah menghapus file yang sudah pernah diinputkan. *Test case* tersebut tidak valid Karena pada saat menekan tombol hapus, tidak ada konfirmasi penghapusan yang seharusnya muncul notifikasi konfirmasi penghapusan pada saat menekan tombol hapus tersebut.

Test case kelima yang tidak valid adalah pada fitur delete file dengan nomor *test case* TC-DFI-02 dimana yang diuji adalah pembatalan penghapusan setelah

menekan tombol menghapus file. *Test case* tersebut tidak valid Karena pada saat menekan tombol hapus, tidak ada konfirmasi penghapusan yang seharusnya muncul notifikasi konfirmasi penghapusan pada saat menekan tombol hapus tersebut.

Test case keenam yang tidak valid adalah pada fitur edit modul dengan nomor *test case* TC-EMO-01 dimana yang diuji adalah proses mengubah data modul saja. *Test case* tersebut tidak valid Karena pada saat masuk form edit modul, kolom *cover* dan file modul otomatis hilang sehingga harus melakukan upload ulang meskipun tidak mengganti *cover* dan file modul. Seharusnya *cover* dan file modul tidak otomatis hilang pada saat masuk kedalam form edit modul sehingga tidak perlu melakukan upload ulang ketika tidak menggantinya.

Test case ketujuh yang tidak valid adalah pada fitur edit modul dengan nomor *test case* TC-EMO -03 dimana yang diuji adalah proses mengubah keseluruhan data modul. *Test case* tersebut tidak valid Karena pada saat masuk form edit modul, kolom *cover* dan file modul otomatis hilang sehingga harus melakukan upload ulang yang dampaknya menjadi tidak tahu file apa yang sudah didalam aplikasi sebelum diubah. Seharusnya *cover* dan file modul tidak otomatis hilang padasaat masuk ke dalam form edit modul sehingga tidak perlu melakukan upload ulang ketika tidak menggantinya.

Test case kedelapan yang tidak valid adalah pada fitur edit modul dengan nomor *test case* TC-EMO-04 dimana yang diuji adalah proses mengubah data modul dengan mengosongi kolom pengarang. *Test case* tersebut tidak valid Karena pada saat masuk form edit modul, kolom *cover* dan file modul otomatis hilang sehingga harus melakukan upload ulang meskipun tidak mengganti *cover* dan file modul. Seharusnya *cover* dan file modul tidak otomatis hilang pada saat masuk kedalam form edit modul sehingga tidak perlu melakukan upload ulang ketika tidak menggantinya.

Test case kesembilan yang tidak valid adalah pada fitur edit modul dengan nomor *test case* TC-EMO-05 dimana yang diuji adalah proses mengubah data modul dengan mengosongi kolom pengarang dan *cover*. *Test case* tersebut tidak valid Karena pada saat masuk form edit modul, kolom *cover* dan file modul otomatis hilang sehingga harus melakukan upload ulang meskipun tidak mengganti *cover* modul. Seharusnya *cover* dan file modul tidak otomatis hilang pada saat masuk kedalam form edit modul sehingga tidak perlu melakukan upload ulang ketika tidak menggantinya.

Test case kesepuluh yang tidak valid adalah pada fitur edit modul dengan nomor *test case* TC-EMO-07 dimana yang diuji adalah proses mengubah data modul dengan mengosongi file modul. *Test case* tersebut tidak valid Karena pada saat masuk form edit modul, kolom *cover* dan file modul otomatis hilang sehingga harus melakukan upload ulang meskipun tidak mengganti file modul. Seharusnya *cover* dan file modul tidak otomatis hilang pada saat masuk kedalam form edit modul sehingga tidak perlu melakukan upload ulang ketika tidak menggantinya.

Test case kesebelas yang tidak valid adalah pada fitur edit modul dengan nomor *test case* TC-EMO-08 dimana yang diuji adalah proses mengubah data modul dengan mengosongi kolom *cover*. *Test case* tersebut tidak valid Karena pada saat masuk form edit modul, kolom *cover* dan file modul otomatis hilang sehingga harus melakukan upload ulang meskipun tidak mengganti *cover* dan file modul. Seharusnya *cover* dan file modul tidak otomatis hilang pada saat masuk kedalam form edit modul sehingga tidak perlu melakukan upload ulang ketika tidak menggantinya.

Test case keduabelas yang tidak valid adalah pada fitur edit modul dengan nomor *test case* TC-EMO-10 dimana yang diuji adalah proses mengubah data modul dengan mengosongi kolom file. *Test case* tersebut tidak valid Karena pada saat masuk form edit modul, kolom *cover* dan file modul otomatis hilang sehingga harus melakukan upload ulang meskipun tidak mengganti *cover* dan file modul. Seharusnya *cover* dan file modul tidak otomatis hilang pada saat masuk kedalam form edit modul sehingga tidak perlu melakukan upload ulang ketika tidak menggantinya.



Tabel 5.2 Tabel Hasil Pengujian Aplikasi PDTT-UB

No	Fitur	Test case ID	Input	Expectation	Realization	Test Status	Note
1	Manajemen Folder	TC-MFO-01	data : not null id_folder : null	Folder baru berhasil dibuat	Folder baru berhasil dibuat	Valid	
		TC-MFO-02	data : not null id_folder : not null	Folder berhasil diubah	Folder berhasil diubah	Valid	
		TC-MFO-03	data : null	folder gagal ditambah atau diubah	folder gagal ditambah atau diubah	Valid	Muncul notifikasi "masukkan data dengan lengkap"
2	Delete Folder	TC-DFO-01	folder tidak ada file didalam	folder berhasil dihapus	folder berhasil dihapus	Tidak Valid	Muncul notifikasi " Apakah anda ingin menghapus folder ini beserta semua isinya?"
		TC-DFO-02	folder tidak ada file didalamnya atau ada file didalamnya konfirmasi : false	folder gagal dihapus	folder gagal dihapus	Valid	Muncul notifikasi " Apakah anda ingin menghapus folder ini beserta semua isinya?"
		TC-DFO-03	folder ada file di dalamnya konfirmasi : true	folder beserta file didalamnya berhasil dihapus	folder beserta file didalamnya berhasil dihapus	Valid	Muncul notifikasi " Apakah anda ingin menghapus folder ini beserta semua isinya?"
3	Manajemen File	TC-MFI-01	data : not null file : not null id_file : null	File baru berhasil dibuat	File baru berhasil dibuat	Tidak Valid	tambah gagal jika file tidak berformat .pdf
			data : not null file : not null id_file : not null	file berhasil diubah	file otomatis hilang	Tidak Valid	file otomatis hilang dan harus upload ulang
			data : null file : not null	file gagal di tambah maupun diubah	file gagal di tambah maupun diubah	Valid	Muncul notifikasi "masukkan data dengan lengkap"
		TC-MFI-04	data : not null file : null	file gagal di tambah maupun diubah	file gagal di tambah maupun diubah	Valid	Muncul notifikasi "masukkan data dengan lengkap"
		TC-MFI-05	data : null file : null	file gagal di tambah maupun diubah	file gagal di tambah maupun diubah	Valid	Muncul notifikasi "masukkan data dengan lengkap"
4	Delete File	TC-DFI-01	file : true konfirmasi : true	file berhasil dihapus	file berhasil dihapus	Tidak Valid	Konfirmasi penghapusan tidak muncul
		TC-DFI-02	file : true konfirmasi : false	file gagal dihapus	file berhasil dihapus	Tidak Valid	Konfirmasi penghapusan tidak muncul
5	Tambah Modul	TC-TMO-01	data : field not null	modul berhasil dibuat	modul berhasil dibuat	Valid	

6	Edit Modul	TC-TMO-02	data : field null	muncul notifikasi field belum diisi	modul gagal dibuat	Valid	Muncul Notifikasi "please fill out this field" pada field yang kosong
		TC-TMO-03	data : field not null	modul berhasil dibuat	modul berhasil dibuat	x	
			pengarang : not null				
		TC-TMO-04	data : field not null	modul berhasil dibuat beserta pengarang		Valid	Muncul Notifikasi "please fill out this field" pada field yang kosong
			pengarang : null				
		TC-EMO-01	data : field not null	modul berhasil diubah	modul berhasil diubah	Tidak Valid	Cover dan file modul otomatis hilang saat masuk di form ubah, jadi harus upload ulang
		TC-EMO-02	data : field null	muncul notifikasi field belum diisi	modul gagal diubah	Valid	
		TC-EMO-03	data : field not null	modul berhasil diubah	modul berhasil diubah	Tidak Valid	Cover dan file modul otomatis hilang saat masuk di form ubah, jadi harus upload ulang
			pengarang : not null				
			cover : not null				
6	Edit Modul	TC-EMO-04	data : field not null	modul berhasil diubah beserta dengan pengarang baru	modul gagal diubah	Tidak Valid	Cover dan file modul otomatis hilang saat masuk di form ubah, jadi harus upload ulang
			pengarang : null				
			cover : not null				
		TC-EMO-05	data : field not null	modul berhasil diubah dengan pengarang baru dan cover baru	modul gagal diubah	Tidak Valid	Muncul Notifikasi "please fill out this field" pada field yang kosong. File modul juga ikut hilang
			pengarang : null				
			cover : null				
		TC-EMO-06	data : field not null	modul berhasil diubah dengan pengarang baru, cover baru dan file baru	modul gagal diubah	Valid	Muncul Notifikasi "please fill out this field" pada field yang kosong
			pengarang : null				
			cover : null				
		TC-EMO-07	data : field not null	modul berhasil diubah dengan pengarang baru dan file baru	modul gagal diubah	Tidak Valid	Cover otomatis hilang saat masuk di form ubah, jadi harus upload ulang
			pengarang : null				
			cover : not null				
6	Edit Modul	TC-EMO-08	data : field not null	modul berhasil diubah dengan cover baru	modul gagal diubah	Tidak Valid	file otomatis hilang dan harus upload ulang
			pengarang : not null				
			cover : null				
		TC-EMO-09	data : field not null	modul berhasil diubah dengan cover dan file baru	modul gagal diubah	valid	Muncul Notifikasi "please fill out this field" pada field yang kosong
			pengarang : not null				
			cover : null				
		TC-EMO-10	data : field not null	modul berhasil diubah	modul gagal diubah	Tidak Valid	Cover otomatis hilang saat
			file : not null				
			file : null				
		TC-EMO-10	data : field not null	modul berhasil diubah	modul gagal diubah	Tidak Valid	Cover otomatis hilang saat

			pengarang : not null cover : not null file : null	dengan file baru			masuk di form ubah, jadi harus upload ulang
7	Delete Modul	TC-DMO-01	modul tidak ada materi didalamnya	modul gagal dihapus	modul gagal dihapus	Valid	
			konfirmasi : false				
		TC-DMO-02	modul tidak ada materi didalamnya	modul berhasil dihapus	modul berhasil dihapus	Valid	
			konfirmasi : true				
		TC-DMO-03	modul ada materi didalamnya	modul beserta file didalamnya berhasil dihapus	modul beserta file didalamnya berhasil dihapus	Valid	
			konfirmasi : true				



BAB 6 Kesimpulan dan Saran

6.1 Kesimpulan

Berdasarkan pembahasan yang telah dilakukan pada penelitian ini, maka kesimpulan yang diperoleh dari penelitian ini adalah :

1. Pembuatan *Test case* pada aplikasi PDTT-UB dengan metode behavior UML teknik model flow graph (MFG) dapat dilakukan dengan cara membuat *Activity diagram* dan *Sequence diagram*, kemudian diagram tersebut di buat flow graph dengan metode MFG yaitu pengelompokan setiap proses yang ada di dalam diagram tersebut, setelah itu flow graph akan di generate menjadi *Test case* dengan metode *Test case Generation*.
2. Setelah melakukan pengujian berdasarkan *Test case* yang dibuat dengan metode behavior UML yang terkumpul pada tabel 5.2 didapat kan ada 18 *test case* yang mendapat label Valid dan 12 *test case* label tidak valid. *Test case* yang medapat label tidak valid tersebut berada pada fitur Delete folder terdapat 1 *test case* tidak valid, Manajemen file terdapat 2 *test case* tidak valid, Delete file terdapat 2 *test case* tidak valid dan Edit modul terdapat 8 *test case* tidak valid

6.2 Saran

Penelitian yang dilakukan tentunya tidak terlepas dari kekurangan dan kelemahan. Oleh karena itu, peneliti menyarankan beberapa hal guna penelitian lebih lanjut terkait tugas akhir ini. Berikut merupakan saran untuk selanjutnya :

1. Untuk developer PDTT-UB agar memperbaiki *error* dan bug yang ada dalam aplikasi PDTT-UB
2. Untuk developer PDTT-UB agar membuat dokumentasi aplikasi PDTT-UB untuk mempermudah perbaikan *error* dan bug aplikasi
3. Untuk penelitian selanjutnya, dapat mencoba UML selain *activity diagram* dan *sequence diagram* yang dapat digunakan untuk membuat *Test case*

Daftar Pustaka

- Alter, S., 1992. *Information System : A Management Perspective*. s.l.:The Benjamin / Cummings Publishing : Company, Inc.
- Beizer, B., 1990. *Software testing Techniques*. 2nd ed. New York: Van Nostrand Reinhold.
- Biswal, B. N., 2010. *Test case Generation and Optimization of Objected-Oriented Software using UML Behavioral Models*. s.l.:s.n.
- Davis, G. B., 2008. *Analisis dan Desain Sistem Informasi*. s.l.:s.n.
- Fathansyah, I., 2001. *Basis Data*. Bandung: CV Informatika.
- Galin, D., 2004. *Software Quality Assurance From Theory to Implementation*. England: Pearson Education Limited.
- Graham, D., 2012. *Foundation of Software Testing ISTQB Certification*. England: Thomson.
- Hartono, J., 1999. *Analisis dan Desain Sistem Informasi*. Yogyakarta: Andi.
- Jogiyanto, 2001. *Sistem Teknologi Informasi*. Yogyakarta: Andi.
- McCabe, T. J., 1996. *Structured Testing : A Testing methodology Using the Cyclomatic Complexity Metric*. MD 20899-0001 ed. Gaithersburg: National Institute Standards and Technology.
- McLeod, R. J., 2008. *Sistem Informasi Manajemen*. Jakarta: Salemba Empat.
- Swain, S. K. & Mohapatra, D. P., 2010. *Test case Generation from Behavioral UML Models*. *International Journal of Computer Applications*, Volume 6, p. 6.
- Turban, McLeand & Wetherbe, 1999. *Information Technology for Management*. New York: John Wiley & Sons.