KAKAS BANTU UNTUK PENENTUAN PRIORITAS *TEST* SCENARIO BERDASARKAN UML ACTIVITY DIAGRAM

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Fatimatuz Zahro
NIM: 155150207111038

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

KAKAS BANTU UNTUK PENENTUAN PRIORITAS TEST SCENARIO BERDASARKAN UML ACTIVITY DAGRAM

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Fatimatuz Zahro
NIM: 155150207111038

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Juni 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Fajar Pradana, S.ST, M.Eng.	Achmad Arwan, S.Kom, M.Kom.
NIP: 19871121 201504 1 004	NIP: 19840815 200812 1 004

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

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, 27 Juni 2019

Fatimatuz Zahro

NIM: 155150207111038

PRAKATA

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Kakas Bantu Untuk Penentuan Prioritas *Test Scenario* Berdasarkan UML *Activity Diagram*" ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Sholihan dan Ibu Adimah selaku kedua orang, Lailatun Nikmah, S.Pd, Afif Setiaji Andrianto, Imroatus Sholihah, S.Pd.I dan Mahmuda Muthmainnah S.T serta keluarga besar atas doa dan dukungan yang diberikan kepada penulis selama proses studi yang ditempuh penulis selama ini,
2. Bapak Bapak Fajar Pradana, S.ST, M.Eng dan Bapak Achmad Arwan, S.Kom, M.Kom selaku Pembimbing skripsi yang telah bersedia membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini,
3. Bapak Agus Wahyu Widodo, S.T, M.Cs. selaku Ketua Program Studi Teknik Informatika,
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika,
5. Teman-teman teknik informatika 2015, teman-teman TIM RPL dan teman-teman The Bagong's atas bantuan, motivasi dan memberikan waktu untuk berdiskusi selama ini,
6. Seluruh civitas akademika Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya

Malang, 22 Mei 2019

Penulis
fatymatuzzahro@student.ub.ac.id

ABSTRAK

Fatimatuz Zahro, Kakas Bantu Untuk Penentuan Prioritas *Test Scenario* Berdasarkan UML *Activity Diagram*

Pembimbing: Fajar Pradana, S.ST, M. Eng dan Achmad Arwan, S.Kom, M.Kom.

Pada pengembangan perangkat lunak terdapat tahapan pengujian yang memiliki peranan sangat penting dalam menentukan kualitas dan fungsi perangkat lunak. Tujuan pengujian dilakukan untuk menemukan *bug* dan kesalahan. Perubahan perangkat lunak secara terus menerus tidak memungkinkan untuk dilakukan pengujian secara keseluruhan. Namun perubahan perangkat lunak seperti logika bisnis, penambahan komponen baru, dan yang lainnya dapat menyebabkan efek karena kesalahan depedensi, propagasi kesalahan, dan kesalahan yang lain. Karena perubahan sistem perangkat lunak baik dari *environment*, *platform*, atau menjadi versi yang lebih tinggi harus dilakukan pengujian ulang secara keseluruhan. Untuk menguji sistem secara keseluruhan akan memakan waktu, membutuhkan biaya yang mahal, tidak praktis, dan tidak efisien. Oleh karena itu, untuk menangani masalah pengujian secara keseluruhan maka dibutuhkan pengujian regresi dengan strategi pengujian yang optimal yaitu dengan metode *test case prioritization* berbasis model. Namun dalam menentukan prioritas *test scenario* menggunakan *test case prioritization* dengan perhitungan faktor pembobotan (*weighting factors*) masih dilakukan secara manual. Perhitungan manual memiliki kelemahan yaitu membutuhkan waktu yang banyak, *human resource*, dan membutuhkan biaya yang mahal. Sehingga dibutuhkan kakas bantu untuk dapat melakukan perhitungan pada penentuan test scenario secara otomatis. Sistem perhitungan kakas bantu penentuan *test scenario* ini menyediakan fitur untuk melakukan perhitungan prioritas *test scenario* pada *activity diagram* dan menampilkan jalur independen yang menjadi prioritas untuk dilakukan pengujian. Sistem ini menggunakan teknologi C# dan sudah diuji menggunakan *whitebox testing* pada pengujian unit dan menggunakan metode *blackbox testing* pada pengujian validasi. Sistem ini melakukan perhitungan penentuan prioritas *test scenario* 311 kali lebih cepat dibandingkan dengan perhitungan secara manual dan mempunyai tingkat akurasi sebesar 100%.

Kata kunci: perangkat lunak, prioritas *test scenario*, *weighting factors*, *activity diagram*

ABSTRACT

Fatimatuz Zahro, Tools for Determining Test Scenario Prioritization Based on UML Activity Diagram

Adviser: Fajar Pradana, S.ST, M. Eng dan Achmad Arwan, S.Kom, M.Kom.

In software development there are stages of testing that have a very important role in determining the quality and function of software. The purpose of testing is to find bugs and errors. Continuous software changes make it impossible to test as a whole. But changes to software such as business logic, adding new components, and others can cause effects due to depedency errors, error propagation, and other errors. Because software system changes from the environment, platform, or to a higher version must be retested as a whole. To test the system as a whole will be time consuming, expensive, impractical, and inefficient. Therefore, to deal with the testing problem as a whole, regression testing with an optimal testing strategy is needed, namely by model-based test case prioritization method. But in determining the priority test scenario using a test case prioritization with the calculation of weighting factors (weighting factors) is still done manually. Manual calculation has the disadvantage of requiring a lot of time, human resources, and expensive costs. So it takes help to be able to do calculations on determining the test scenario automatically. The aiding calculation system for determining the test scenario provides a feature for calculating the priority of test scenarios in activity diagrams and displaying independent pathways which are prioritized for testing. This system uses C# technology and has been tested using whitebox testing on unit testing and uses the blackbox testing method in validation testing. This system calculates test scenario priority 311 times faster than manual calculations and has an accuracy rate of 100%.

Kata kunci: perangkat lunak, prioritas *test scenario*, *weighting factors*, *activity diagram*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Perangkat lunak merupakan abstraksi fisik yang digunakan untuk berkomunikasi dengan mesin perangkat keras (Maulana, 2017). Perangkat lunak akan terus berkembang sehingga sistem perangkat lunak dengan kualitas yang sangat baik menjadi kebutuhan sangat penting (Widodo *et al.*, 2016). Pada pengembangan perangkat lunak terdapat beberapa tahapan, yang pertama analisis kebutuhan menganalisis kebutuhan, merancang sistem dari kebutuhan yang diperoleh, mengimplementasikan desain, melakukan pengujian pada sistem dan pada tahapan terakhir melakukan pemeliharaan. Beberapa tahapan pengembangan perangkat lunak tersebut, tahap pengujian memiliki peranan yang sangat penting dalam menentukan kualitas dan fungsi perangkat lunak (Khandelwal & Bhadauria, 2013).

Pada tahap pemeliharaan, evolusi perangkat lunak yang terus menerus mustahil untuk melakukan pengujian secara lengkap atau mendalam (Bhuyan, Ray & Das, 2017). Evolusi perangkat lunak seperti perubahan-perubahan dalam logika bisnis, penambahan komponen baru dan lain sebagainya, dapat menyebabkan efek karena kesalahan dependensi, propagasi kesalahan dan beberapa kesalahan lainnya (Mahali & Acharya, 2013). Karena perubahan sistem perangkat lunak baik melalui perubahan *environment*, *platform* atau hanya berkembang menjadi versi yang lebih tinggi harus dilakukan pengujian ulang secara keseluruhan untuk memastikan kualitas perangkat lunak (Rava & Wan-Kadir, 2016). Sehingga pada tahap pemeliharaan evolusi perangkat lunak bergantung pada pengujian regresi.

Pengujian perangkat lunak merupakan langkah yang paling signifikan pada *life cycle* pengembangan perangkat lunak, karena pada sebuah perangkat lunak dapat dikatakan perangkat lunak baik jika tidak terdapat *bug* dan *error* (Arwan & Rusdianto, 2018). Pengujian perangkat lunak memiliki tujuan untuk menemukan *bug* dan *error* perangkat lunak (Sultan *et al.*, 2017). Menurut Putra, Siahaan dan Yuhana (2014) setiap *bug* memiliki tingkat keparahannya terdapat 4 macam keparahan dari *bug*, yang pertama *blockers, critical, major, minor* dan *trivial*. *Blockers* yaitu tidak ada uji coba lanjutan yang dapat dikerjakan. *Critical* yaitu cacat yang harus diperbaiki untuk pemrosesan lebih lanjut dari aplikasi. *Major* yaitu cacat utama yang mempengaruhi pemrosesan terpilih ke tingkat yang signifikan atau kehilangan banyak fungsi. *Minor* yaitu cacat kecil tetapi tidak mempengaruhi proses dan pengujian lebih lanjut. Dan yang terakhir *trivial* yaitu cacat kecil yang terjadi pada antarmuka. Sehingga untuk menemukan *bug* perangkat lunak dalam sistem diperlukan pengujian secara menyeluruh yaitu pengujian regresi untuk memastikan bahwa fungsi yang ada berfungsi dengan baik.

Pengujian regresi merupakan proses pengujian ulang sistem atau komponen secara selektif untuk memverifikasi bahwa modifikasi tidak menyebabkan efek yang tidak diinginkan bahwa sistem atau komponen masih sesuai dengan persyaratan spesifiknya (IEEE). Pengujian regresi bertujuan untuk memastikan

kualitas perangkat lunak setelah perubahan atau modifikasi (Mahali & Acharya, 2013). Pengujian regresi berfokus pada aspek perubahan dan memastikan bahwa perubahan yang terjadi pada perangkat tidak memiliki dampak negatif atau tidak mempengaruhi modul lain dari perangkat lunak (Rava & Wan-Kadir, 2016). Untuk meningkatkan ukuran dan kompleksitas perangkat lunak pada saat terjadi perubahan menjalankan kembali seluruh rangkaian uji akan memakan waktu dan mahal, serta hal ini tidak praktis dan tidak efisien untuk menjalankan pengujian secara mendalam (Bhuyan, Ray & Das, 2017).

Dalam melakukan pengujian regresi, dibutuhkan strategi pengujian yang optimal dan mampu mengidentifikasi kesalahan maksimum dengan waktu yang efisien yaitu dengan menggunakan metode *test case prioritization*. *Test case prioritization* membantu mengorganisir kasus uji secara efektif untuk menentukan kasus-kasus pengujian yang menguntungkan akan dilaksanakan terlebih dahulu untuk di uji (Bhuyan, Ray & Das, 2017). Keuntungan dari prioritas adalah mendeteksi sedini mungkin kesalahan yang membantu penguji untuk menyelesaikan kesalahan pada tahap awal (Mahali & Acharya, 2013). Menurut Korel dan Koutsogiannakis (2009) pada *test case prioritization* terdapat 2 kategori: *test case prioritization* berbasis kode dan berbasis model. T*est case prioritization* berbasis kode biasanya menggunakan informasi seperti cakupan pernyataan, cakupan cabang dan jumlah kesalahan. *Test case prioritization* berbasis kode memiliki kerugian yaitu sangat mahal karena pelaksanaanya lambat dan tidak sensitif terhadap informasi yang benar atau salah yang diberikan oleh penguji atau pengembang. *Test case prioritization* berbasis model didasarkan pada model sistem. *Test case prioritization* berbasis model pada sistem perangkat lunak akan mendeteksi kesalahan lebih awal dan membutuhkan biaya yang murah.

Berdasarkan permasalahan-permasalahan tersebut, terdapat penelitian dengan memprioritaskan pengujian dengan strategi *test case prioritization* oleh Bhuyan, Ray dan Das (2017). Pada penelitian tersebut, menggunakan strategi *test case prioritization* berbasis model yaitu menggunakan UML activity diagram. Untuk menentukan prioritas, menggunakan faktor pembobotan (*weighting factors*) yaitu dengan menghitung bobot setiap jalur $W(P_i)$, nilai kompleksitas setiap jalur $CV(P_i)$, cakupan node pada masing-masing jalur $N(P_i)$, decision node $D(P_i)$, dan bobot setiap jalur berdasarkan jenis kopling $W1(P_i)$.

Pada penyelesaian yang diusulkan oleh Bhuyan, Ray dan Das (2017) untuk menentukan prioritas *test scenario* dengan strategi *test case prioritization* dengan perhitungan dari setiap faktor pembobotan masih dilakukan secara manual. Perhitungan yang dilakukan secara manual memiliki kelemahan diantaranya *time consuming* yaitu membutuhkan banyak waktu, *human resource* yaitu membutuhkan lebih banyak penguji, kurang dapat diandalkan karena pengujian tidak dapat dilakukan dengan ketelitian setiap kali karena kesalahan manusia dan pengujian manual lebih mahal (Rathi & Mehra, 2015). Maka untuk proses evaluasi yang lebih efisien, sistem prioritas *test scenario* akan diukur dengan menggunakan kakas bantu.

Kakas bantu sebelumnya sudah dikembangkan oleh peneliti-peneliti, seperti penelitian menurut Ubaidillah, Pradana dan Priyambadha (2017) yaitu sebuah penelitian kakas bantu untuk melakukan perhitungan nilai kopling dengan membandingkan perhitungan secara manual dan sistem dan mendapatkan nilai akurasi 100%. Menurut Rathi dan Mehra (2015) melakukan analisis perbandingan pengujian secara manual dan pengujian secara otomatis menggunakan sebuah *tools*, dari analisis tersebut didapatkan bahwa pengujian manual memakan waktu, membosankan dan membutuhkan investasi besar dalam sumber daya manusia. Sedangkan dengan menggunakan kakas bantu dapat melakukan *record* dan bisa digunakan kembali jika diperlukan. Oleh karena itu, dengan adanya kakas bantu diharapkan dapat mempersingkat waktu yang dibutuhkan untuk melakukan pengujian dengan prioritas *test scenario*.

Berdasarkan dari latar belakang yang telah diuraikan penulis menarik sebuah judul "Kakas Bantu Untuk Penentuan Prioritas *Test Scenario* Berdasarkan UML *Activity Diagram*" dengan kakas bantu ini agar dapat melakukan pengujian pada test scenario yang menggunakan *test case prioritization* berbasis model dengan mudah dan lebih cepat.

1.2 Rumusan masalah

Dari latar belakang yang telah dijabarkan, rumusan dari permasalahan adalah:

1. Bagaimana hasil analisis, perancangan dan implementasi kakas bantu penentuan prioritas *test scenario* berdasarkan UML *activity diagram* ?
2. Bagaimana cara menentukan prioritas *test scenario* dengan strategi *test case prioritization* berdasarkan model UML *activity diagram* berdasarkan perhitungan faktor pembobotan?
3. Bagaimana hasil dari pengujian pada kakas bantu penentuan prioritas *test scenario* berdasarkan UML *activity diagram* ?

1.3 Tujuan

Tujuan dari kakas bantu untuk penentuan *test scenario* berdasarkan UML *activity diagram* adalah:

1. Menganalisis, merancang dan mengimplementasikan kakas bantu penentuan prioritas *test scenario* berdasarkan UML *activity diagram.*
2. Mengetahui cara menentukan prioritas *test scenario* dengan strategi *test case prioritization* berdasarkan model UML *Activity Diagram* berdasarkan perhitungan faktor pembobotan.
3. Mengetahui hasil dari pengujian pada kakas bantu penentuan prioritas *test scenario* berdasarkan UML *activity diagram.*

1.4 Manfaat

Manfaat yang diperoleh dari penelitian adalah:

1. Mempermudah proses perhitungan penentuan prioritas *test scenario*.

2. Membantu *developer* untuk mengetahui prioritas pada jalur independen yang akan dilakukan pengujian terlebih dahulu

1.5 Batasan masalah

Batasan masalah pada kakas bantu penentuan prioritas test scenario berdasarkan UML *activity diagram* yang sudah ditetapkan adalah:

1. Pembahasan hanya difokuskan pada penentuan prioritas *test scenario* yang berbasis UML *activity diagram.*
2. Format yang dapat digunakan dalam melakukan proses *parsing* dengan ekstensi ".xml".
3. Menggunakan tools visual paradigm untuk membuat dokumen XML.
4. Prioritas *test scenario* hanya dilakukan pada pengujian regresi.
5. Kakas bantu penentuan prioritas *test scenario* berdasarkan UML *activity diagram* menggunakan bahasa pemrograman C#.

1.6 Sistematika pembahasan

Sistematika pembahasan untuk menjelaskan secara garis besar dalam penelitian adalah:

BAB 1: PENDAHULUAN

Bab pendahuluan ini menjelaskan tentang latar belakang dari penelitian, rumusan masalah pada penelitian, tujuan penelitian, manfaat dari penelitian, batasan masalah, dan sistematika pembahasan.

BAB 2: LANDASAN KEPUSTAKAAN

Bab landasan kepustakaan menjelaskan kajian pustaka, penelitian – penelitian sebelumnya, dan menjelaskan dasar teori pada penelitian. Dasar teori yang digunakan pada penelitian ini adalah hasil penelitian relevan dengan penentuan prioritas *test scenario* berdasarkan UML *activity diagram.*

BAB 3: METODOLOGI

Bab metodologi menganalisis kebutuhan, merancang dan implementasi Pengembangan Sistem Untuk Penentuan Prioritas *Test Scenario* Berdasarkan UML a*ctivity diagram.*

BAB 4: ANALISIS KEBUTUHAN

Bab analisis kebutuhan yaitu menganalisis kebutuhan yang akan diterapkan pada kakas bantu penentuan prioritas *test scenario* berdasarkan UML *activity diagram*.

BAB 5: PERANCANGAN DAN IMPLEMENTASI SISTEM

Bab perancangan dan implementasi menjabarkan perancangan dari analisis kebutuhan pada kakas bantu penentuan prioritas *test scenario* dan menjelaskan gambaran sistem pada sistem penentuan prioritas *test scenario* berdasarkan UML *activity diagram*.

BAB 6: PENGUJIAN SISTEM

Bab pengujian akan menjelaskan tentang pengujian unit dan pengujian validasi pada kakas bantu penentuan prioritas *test scenario.*

BAB 7: PENUTUP

Bab penutup menjelaskan kesimpulan yang diperoleh dari hasil penelitian serta saran untuk pengembangan penelitian lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Terdapat penelitian sebelumnya yang dilakukan oleh Korel dan Koutsogiannakis (2009) dengan membandingkan *test case prioritization* berbasis kode dan *test case prioritization* berbasis model, dengan tujuan mengevaluasi metode-metode yang berhubungan dengan efektivitas untuk mendeteksi kesalahan awal dalam sistem yang sudah dimodifikasi. Penelitian ini menunjukkan bahwa *test case prioritization* berbasis model dapat meningkat secara signifikan dalam mendeteksi kesalahan awal dan lebih murah dibandingkan dengan *test case prioritization* berbasis kode.

Test case prioritization berbasis model untuk memprioritaskan pengujian menggunakan UML *activity diagram*. Pada penelitian P.G. dan Mohanty, (2009) *activity diagram* digunakan untuk mewakili skenario yang terkait dengan *use case* pada *use case diagram* dan digunakan oleh stakeholder sistem untuk memahami fungsi sistem. Setiap *use case* dapat diwakili dengan satu atau lebih diagram aktivitas. *Activity diagram* merepresentasikan skenario yang terkait dengan *use case diagram*. Tujuan dari memprioritaskan skenario yang diambil dari diagram aktivitas adalah untuk mengidentifikasi kepentingan relatif dari skenario (Sapna & Hrushikesha, 2009).

Berdasarkan *test case prioritization* berbasis model menggunakan UML *activity diagram* untuk memprioritaskan skenario pengujian memerlukan perhitungan pada setiap jalur di *flow graph*. Bhuyan, Ray dan Das (2017) melakukan penelitian memprioritaskan skenario pengujian berdasarkan *test case prioritization* dengan UML *use case diagram* dan *activity diagram*, penelitian menggunakan faktor pembobotan dengan menghitung 5 faktor yaitu menghitung masing-masing jalur, kompleksitas jalur, cakupan node, decision node dan menghitung bobot berdasarkan jenis kopling. Akan tetapi, pada penelitian ini memiliki kekurangan yaitu perhitungan yang masih manual. Perhitungan dengan cara yang manual pada perangkat lunak dengan jumlah yang banyak akan menghabiskan sumber daya yang besar dan membutuhkan waktu yang banyak (Ubaidillah, Pradana & Priyambadha, 2017).

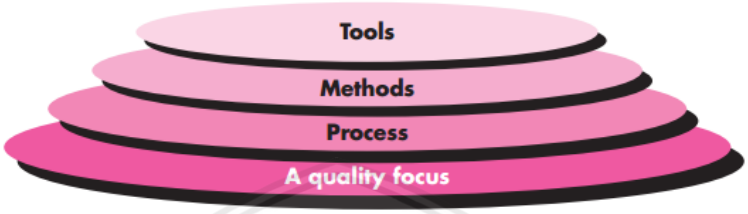
Kakas bantu pada penelitian yang dilakukan oleh Ubaidillah, Pradana dan Priyambadha (2017) pada perhitungan nilai kopling maka diperoleh hasil pengujian dengan tingkat akurasi sebesar 100% berdasarkan inputan lima kode sumber bahasa pemrograman java. Untuk itu penelitian ini mengimplementasikan kakas bantu untuk memprioritaskan skenario pengujian dengan *test case prioritization* berdasarkan UML *activity diagram* dengan perhitungan faktor pembobotan.

2.2 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah penerapan atau penggunaan dengan menggunakan dasar-dasar rekayasa agar dapat menghasilkan perangkat lunak

yang ekonomis, bekerja lebih efisien, serta handal pada mesin yang sebenarnya (Pressman, 2010). Dalam definisi lain, Rekayasa perangkat lunak merupakan penerapan pendekatan yang sistematis, disiplin, dapat terukur untuk pengembangan, operasi, dan pemeliharaan perangkat lunak (IEEE).

Secara umum pada rekayasa perangkat lunak adalah sebuah *technology* yang berlapis-lapis atau teknologi bertingkat, lapisan teknologi pada rekayasa perangkat lunak antara lain:



Gambar 2.1 *Software Engineering Layer*

(Sumber: Pressman, 2010)

Penjelasan pada Gambar 2.1 *Software Engineering Layer* adalah sebagai berikut :

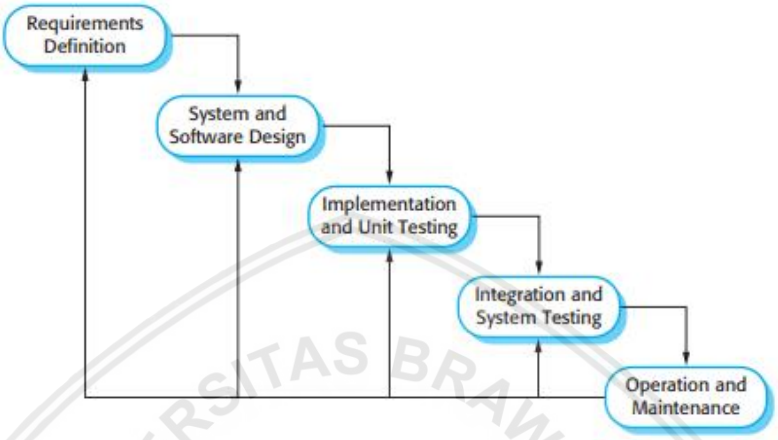
1. *A quality of focus* : manajemen kualitas secara menyeluruh, six sigma, serta filosofi yang sama akan menumbuhkan peningkatan proses yang berkelanjutan dan pada akhirnya akan menuju ke pendekatan yang lebih efektif untuk rekayasa perangkat lunak. *A quality of focus* merupakan lapisan dasar yang mendukung rekayasa perangkat lunak.
2. *Process* : lapisan yang menyatukan lapisan *technology* dan memungkinkan perangkat lunak computer berkembang yang rasional dan tepat waktu. Proses mengartikan *framework* yang ditentukan agar teknologi yang akan digunakan pada penyusunan program dimanfaatkan dengan efektif dan bentuk dasar untuk pengendalian manajemen proyek perangkat lunak.
3. *Methods* : lapisan yang menyediakan sarana teknis untuk mengembangkan sebuah perangkat lunak. Metode memiliki berbagai macam tugas beragam tugas yang meliputi *requirement analysis, communications*, *desaign, implementation, testing,* dan dukungan (*support*).
4. *Tools* : lapisan yang memberikan dukungan otomatis atau semi-otomatis untuk *process* dan *methods*. Ketika *tools* terintegrasi sehingga informasi yang diperoleh dari suatu alat (*tools*) dapat digunakan oleh yang lain, maka membentuk sebuah sistem yang disebut dengan *Computer Aided Software Engineering*.

2.3 Pengembangan Perangkat Lunak

Pada pengembangan perangkat lunak terdapat beberapa pembahasan, diantaranya model *waterfall,* pendekatan berorientasi objek (*Object Oriented*), pemodelan berorientasi objek, pemrograman berorientasi objek, dan pengujian perangkat lunak.

2.3.1 Model Waterfall

Model *waterfall* merupakan mencerminkan proses dasar spesifikasi, pengembangan, validasi (Sommerville, 2011). Model *waterfall* dianggap sesuai dengan pembangunan perangkat lunak ini, karena prosesnya yang bertahap (tiap tahapan) sehingga akan memperoleh kualitas yang baik. Pada model *waterfall* ada 5 tahapan seperti pada Gambar 2.2 diantaranya adalah :



Gambar 2.2 Model *Waterfall*

(Sumber : Sommerville, 2011)

Pada tahapan-tahapan model *waterfall* akan dijelaskan sebagai berikut:

1. *Requirement Analysis and Definition*
   Dalam *requirement analysis and definition* merupakan penentuan fitur, hambatan dan tujuan pada sistem yang diperoleh dari pengguna sistem, yang akan ditetapkan sebagai spesifikasi sistem.
2. *System and Software Design*
   Mengidentifikasi dasar sistem perangkat lunak serta hubungan-hubungan pada sistem perangkat lunak. Dan membentuk suatu arsitektur sistem.
3. *Implementation and Unit Testing*
   Implementation and unit testing merupakan tahapan sistem akan diimplementasikan sebagai suatu unit program. Setelah diimplementasikan maka akan dilakukan pengujian untuk mengetahui apakah sudah memenuhi spesifikasi sistemnya.
4. *Integration and System Testing*
   Pada pengujian sistem dan integrasi sistem setiap unit program akan diintegrasikan dengan unit program yang lain. Setelah sistem yang sudah diintegrasikan akan dikirim kepada pengguna sistem.
5. *Operation and Maintenance*
   Pada tahapan terakhir sistem akan diinstal dan sudah mulai digunakan. Ketika ada *error* pada sebuah sistem yang tidak diketahui pada waktu pembuatan maka dilakukan perbaikan. Dan jika ada penambahan fitur maka dilakukan pada tahapan ini.

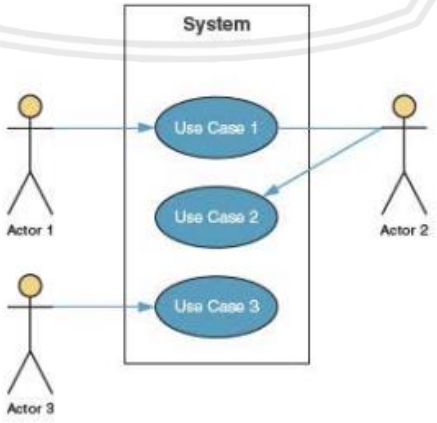
2.3.2 Pendekatan Berorientasi Objek

Pendekatan berorientasi objek atau *object oriented* adalah pendekatan yang dapat dilakukan dalam pengembangan perangkat lunak. Pada pendekatan perangkat lunak dimulai dari menganalisis kebutuhan yaitu dengan *Object Oriented Analysis* (OOA). Dengan menggunakan metode OOA untuk menemukan kebutuhan sistem, objek dan kelas diperoleh dari domain permasalahan. Selanjutnya yaitu tahap perancangan atau tahap *Object Oriented Design* (OOD) yang berdasarkan dari hasil analisis kebutuhan pada tahap OOA sebelumnya. Pada proses OOD yaitu perancangan arsitektur, komponen, data, dan antarmuka. Setelah melakukan perancangan pada sistem maka selanjutnya tahap merubah hasil perancangan menjadi bentuk kode program dengan menggunakan bahasa pemrograman yang dapat dibaca oleh mesin, pada membentuk kode program ini disebut *Object Oriented Programming* (OOP). Setelah tahapan OOA, OOD, dan OOP maka selanjutnya *Object Oriented Testing* (OOT) atau tahapan untuk melakukan pengujian. Dalam *Object Oriented Testing* dilakukan pengujian pada hasil perancangan arsitektur, antarmuka, dan komponen (Pressman, 2010).

2.3.3 Pemodelan Berorientasi Objek

Menurut Whitten dan Bentley (2007), UML adalah satu set dari ketentuan *modeling* yang digunakan untuk menspesifikasi atau mendeskripsikan sebuah sistem perangkat lunak dalam suatu kondisi dari objek. UML dibagi menjadi beberapa komponen :

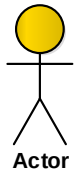
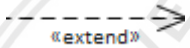
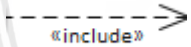
1. *Use Case Diagram*
   Use case diagram adalah diagram yang menggambarkan interaksi antara sistem dan pengguna. *Use case diagram* menggambarkan siapa yang akan menggunakan sistem dan dengan cara apa pengguna mengharapkan untuk berinteraksi dengan sistem (Whitten & Bentley, 2007). Contoh dari *use case diagram* ditunjukan pada Gambar 2.3 dan untuk penjelasan notasi pada *use case diagram* akan dijelaskan pada Tabel 2.1.



Gambar 2.3 *Use Case Diagram*

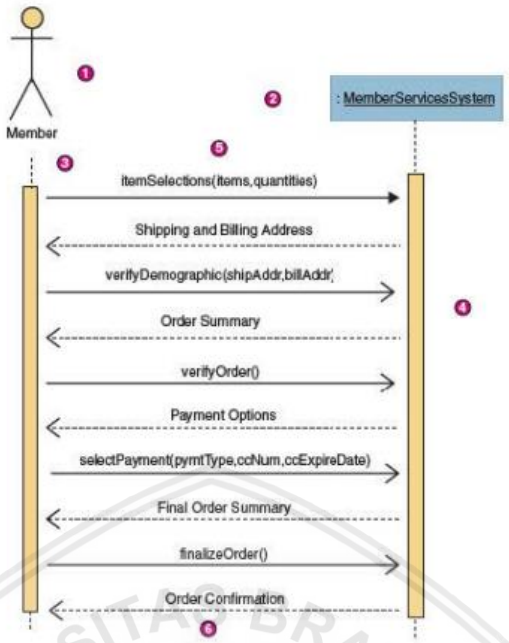
(Sumber : Whitten and Bentley, 2007)

Tabel 2.1 Notasi pada *Use Case Diagram*

Notasi	*Description*
Actor 	Aktor mewakili peran yang dipenuhi oleh pengguna yang berinteraksi dengan sistem untuk bertukar informasi dan untuk tujuan menyelesaikan beberapa tugas bisnis yang menghasilkan sesuatu yang dapat diukur nilainya.
Use Case 	Simbol *use case* memaparkan notasi fungsionalitas pada perangkat lunak yang akan dibangun dengan penamaan menggunakan kata kerja. *Use case* memiliki korelasi dengan aktor untuk bertukar data.
Assosiasi	Hubungan antara aktor dan *use case* dimana terjadi interaksi antara aktor dan *use case* tersebut
Extends 	Hubungan *use case* tambahan dimana use case yang ditambahkan dapat berdiri sendiri atau untuk menyederhanakan masalah orisinil use case yang kompleks
Includes 	Hubungan *use case* yang terikat terhadap *use case* lain untuk rangkaian *use case* yang perlu dikembangkan.
Generalisasi 	Hubungan antara yang memiliki arti umum ke khusus, pada generalisasi aktor yang satu adalah aktor yang lebih umum dari aktor yang lainnya.

2. *Sequence Diagram*

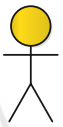
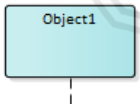
   Sequence diagram adalah urutan sistem diagram yang memvisualisasikan bagaimana objek yang satu dapat berhubungan atau dapat berinteraksi dengan objek lainnya. Interaksi pada objek ini dilakukan melalui pesan dan interaksi ini mendeskripsikan sebuah operasi dari sebuah fitur pada perangkat lunak (Whitten & Bentley, 2007). Contoh dari *sequence diagram* ditunjukkan pada gambar 2.4 dan untuk notasi pada *sequence diagram* akan dijelaskan pada Tabel 2.2 sampai Tabel 2.3.



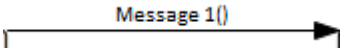
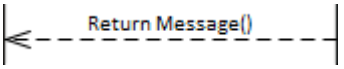
Gambar 2.4 *Sequence Diagram*

(Sumber : Whitten & Bentley, 2007)

Tabel 2.2 Notasi pada *Sequence Diagram*

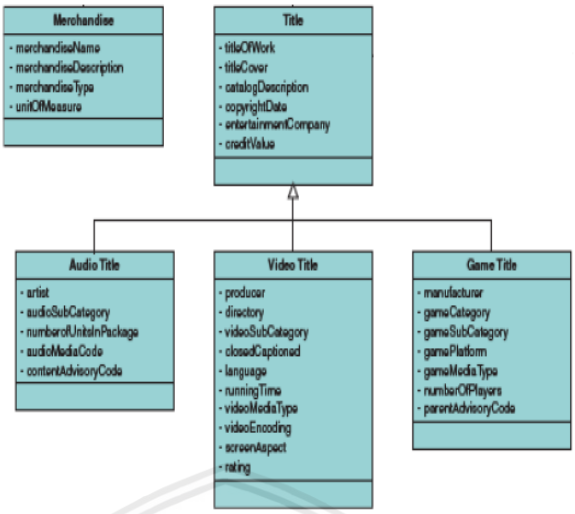
Notasi	*Description*
Actor 	Aktor mewakili peran yang dipenuhi oleh pengguna yang berinteraksi dengan perangkat lunak yang dibangun. Pengguna tidak hanya dapat diartikan dengan pengguna orang saja melainkan dapat menggambarkan sistem lain yang di luar sistem.
Object 	Notasi yang menggambarkan sebuah objek berinteraksi untuk bertukar pesan untuk menjalankan operasi pada sebuah fitur.
Lifeline	*Lifeline* divisualisasikan dalam garis yang putus-putus dengan bentuk vertikal memanjang ke bawah dari aktor dan simbol sistem, yang menunjukkan waktu aktivitas pada suatu objek.
Activation bar 	Bar yang terdapat pada jalur proses yang menunjukkan periode waktu ketika suatu objek aktif dalam interaksi.

Tabel 2.3 Notasi pada *Sequence Diagram (Lanjutan)*

Notasi	*Description*
Input messages 	Divisualisasikan seperti panah dengan bentuk horizontal dari aktor ke sistem yang menunjukkan adanya pesan atau memanggil operasi pada objek yang lain.
Output messages 	Divisualisasikan seperti panah dengan garis putus-putus berbentuk garis horizontal yang mendeskripsikan sebuah objek yang sudah melaksanakan operasinya dan memiliki data balikan pada objek yang memanggilnya.
Frame 	Divisualisasikan dalam bentuk kotak yang menunjukkan langkah loop, alternatif, atau opsional, serta mengandung satu atau lebih pesan agar dapat membagi *fragmen sequence.* dapat menyertakan satu atau lebih pesan untuk membagi fragmen *sequence*.
Boundary 	*Boundary* merupakan simbol yang mendeskripsikan kumpulan suatu kelas yang menjadi *interface* agar actor dengan sistem saling berinteraksi.
Control 	*Control* merupakan simbol yang mendeskripsikan kumpulan suatu kelas yang terdiri dari logika alur sistem.
Entity 	*Entity* merupakan simbol yang mendeskripsikan kumpulan suatu kelas untuk menyusun basis data dan menyimpan informasi pada basis data.

3. *Class* Diagram

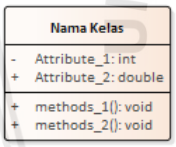
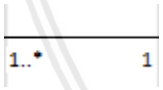
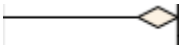
   Class diagram adalah penggambaran struktur dari suatu objek untuk menyusun sistem. *Class diagram* mempresentasikan objek dari class-class yang saling berhubungan (Whitten & Bentley, 2007). Contoh dari *class diagram* ditunjukkan pada gambar 2.5 dan penjelasan notasi pada *class diagram* dijelaskan pada Table 2.5.



Gambar 2.5 *Class Diagram*

(Sumber : Whitten & Bentley, 2007)

Tabel 2.4 Notasi pada *Class Diagram*

Notasi	*Description*
Class 	*Class* dideskripsikan sebuah kotak yang memiliki beberapa bagian yaitu nama class, atribut class, dan *methods* pada *class*. Pada bagian atribut *class* terdapat jenis tipe data atribut dan jenis *modifier*. Pada bagian methods yaitu jenis *return type* dan jenis *modifier*.
Association 	*Association* memaparkan hubungan antar dua kelas secara umum. Terdapat beberapa tipe-tipe hubungan pada notasi ini seperti *one-to-one, one-to-many,* dan *many-to-many*.
Composition 	*Composition* memaparkan hubungan antar dua kelas atau ketergantungan antar kelas dimana sebuah kelas tersebut tidak dapat berdiri sendiri.
Aggregations 	*Aggregations* memaparkan hubungan antar dua kelas dengan artian sebuah *class* dapat berdiri sendiri meskipun merupakan *class* tersebut merupakan bagian dari *class* yang lain.
Generalizations 	*Generalizations* memaparkan hubungan antar dua kelas dengan artian *class* yang memiliki bentuk umum ke khusus dari *class* yang lain.

2.3.4 Pemrograman Berorientasi Objek (PBO)

Pemrograman berorientasi objek atau *Object Oriented Programming* merupakan proses yang dilakukan setelah perancangan pada sistem yang kemudian diimplementasikan menjadi kode program. Menurut (Baesens, Backiel, & Broucke 2015) terdapat beberapa karakteristik dalam PBO adalah:

a. *Object*
   Object adalah instansiasi dari sebuah class, di mana object tersebut merupakan suatu entitas yang memiliki *attribute* dan *method*.
b. *Inheritance*
   Inheritance adalah cara ketika membuat sebuah *class* baru dengan memanfaatkan *class* yang sudah ada atau sudah dibuat sebelumnya. Dalam hubungan inheritance, sebuah *child class* atau yang biasa disebut *class* turunan akan mewarisi atribut dan *method* yang ada pada *parent class* (*class* induk). Hal ini membuat program menjadi lebih mudah, karena memungkinkan *developer* untuk menggabungkan karakteristik umum kedalam objek *class* dan akan mewariskan karakteristik tersebut di dalam objek *class* turunan.
c. *Polymorphism*
   Polymorphism adalah teknik penamaan dalam pemrograman objek yang beberapa memiliki nama yang sama tetapi mempunyai fungsi yang berbeda. *Polymorphism* memiliki dua jenis yaitu *Overloading* dan *Overriding. Overloading* adalah penggunaan *method* dengan nama yang sama hanya pada beberapa *method* akan tetapi mempunyai fungsi dan parameter yang berbeda. *Overriding* adalah mengganti atau merubah method yang didapatkan atau diturunkan dari *parent class*.
d. Encapsulation
   Encapsulation adalah proses dimana tidak ada akses langsung yang diberikan untuk data, melainkan tersembunyi. Pada *encapsulation* akan mendefinisikan akses pada variabel maupun *method* yang berada pada suatu *class.* Untuk mendapatkan akses ke data, diperlukan interaksi dengan objek yang didefinisikan sebelumnya untuk data tersebut.

2.3.5 Pengujian Perangkat Lunak

a. *White-box Testing*

   White-box testing merupakan metode untuk melakukan pengujian program pada perangkat lunak dimana pengujian berdasarkan pengetahuan tentang struktur program dan komponennya (Pressman, 2010). Pada *white-box* testing pengujian dengan menentukan *flow graph* dari sebuah algoritme kemudian dihitung berdasarkan *cyclometic complexity* maka akan menghasilkan jalur independen sejumlah dengan jumlah cyclometic complexity. *Flow graph* digambarkan dengan sebuah node dengan bulatan berisi data yang biasanya berupa angka untuk merepresentasikan alur program, serta digambarkan dengan edge untuk menghubungkan node satu dengan node yang lainnya.

b. *Blackbox Testing*

Black-box testing merupakan pendekatan dalam pengujian yang berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan pengujian untuk memperoleh set kondisi input yang secara keseluruhan melakukan persyaratan fungsional untuk sebuah program (Pressman, 2010). Pengujian dengan *black-box testing* dilakukan dengan merancang sebanyak jumlah *test case* yang telah dibuat untuk menemukan *error* pada level validasi perangkat lunak (*software validation level*). Pada *black-box testing* pengujian dilakukan untuk menemukan kecacatan pada sistem seperti fungsi yang hilang atau fungsi yang tidak benar, kesalahan pada stuktur data, kesalahan pada antarmuka, kesalahan perilaku (behavior), dan kesalahan inisialisasi.

2.4 Test Case Prioritization

Test case merupakan prosedur yang dirancang untuk menguji fungsionalitas fitur dalam suatu sistem. Tujuan utama merancang *test case* adalah untuk mendeteksi kesalahan dalam suatu sistem. Sedangkan *test scenario* merupakan seperangkat kasus uji yang memastikan bahwa alur proses bisnis diuji dari ujung awal ke ujung akhir.

Test case prioritization merupakan aspek yang aman dari pengujian regresi. *Test case prioritization* mengatur ulang kasus pengujian dalam test suite T dan tidak membuang test case dalam T untuk eksekusi menuju tujuan pengujian yang dipilih (dilambangkan dengan G) (Jiang & Chan, 2015). Tujuan *test case prioritization* adalah untuk meningkatkan kemungkinan bahwa jika kasus uji diprioritaskan dalam beberapa urutan, mereka akan memenuhi tujuan yang ditentukan dalam waktu dan biaya yang ditetapkan daripada jika tidak diprioritaskan. *Test case prioritization* dapat mengatasi berbagai macam tujuan, misalnya sebagai berikut :

a. Pengembang perangkat lunak / penguji berniat meningkatkan laju deteksi kesalahan.
b. Mendeteksi kesalahan berisiko tinggi sebelumnya dalam pengujian siklus hidup.
c. Untuk meningkatkan kemungkinan kesalahan regresi terkait dengan perubahan kode khusus sangat awal dalam proses pengujian
d. Untuk meningkatkan cakupan kode yang dapat ditutup dengan lebih cepat.
e. Untuk membuat sistem lebih dapat diandalkan

2.4.1 Weighting Factors

Untuk memprioritaskan skenario pengujian, berikut faktor pembobotan dalam penetapan prioritas :

1. Weight of each test path
   Menghitung masing-masing jalur berdasarkan kompleksitas dari setiap node yang dilewati, dapat dilihat pada persamaan 2.1 sebagai berikut:

$$W(P_i) = \sum(Kompleksitas\ dari\ setiap\ node\ yang\ dilewati\ oleh\ P_i) \quad (2.1)$$

Untuk menghitung kompleksitas dari setiap node maka dihitung dengan menggunakan pada persamaan 2.2

$$IN(N_i)\ x\ OUT(N_i) \quad (2.2)$$

Keterangan :

$IN(N_i)$ adalah jumlah node dengan edge masuk menuju ke node N_i

$OUT(N_i)$ adalah jumlah node dengan edge keluar menuju ke node N_i

2. Complexity value of each path
   Nilai kompleksitas pada setiap jalur dihitung berdasarkan pada menemukan berapa banyak jumlah node yang dipengaruhi oleh node tertentu, misalnya node tertentu tersebut node yang merupakan aktivitas utama dalam suatu sistem yang akan dibuat. Dengan perhitungan pada persamaan 2.3 sebagai berikut:

$$CV(N_i) = \frac{Amount\ of\ node\ contribution}{Total\ number\ of\ affected\ nodes} \quad (2.3)$$

   Amount of node contribution : jumlah node yang terpengaruh oleh node tertentu
   Total number of affected nodes : jumlah node yang mempengaruhi

3. Node coverage
   Jumlah node yang dilalui dari masing-masing jalur independen.
4. Decision node coverage
   Jumlah keputusan atau pengkondisian dari masing-masing jalur independen.
5. *Weight of each path*
   Perhitungan berdasarkan jenis kopling. Perhitungan bobot didasarkan pada edge dengan mempertimbangkan jenis kopling seperti *data coupling*, *stamp coupling*, dan *controll coupling* antara node yang berhubungan dengan edge, pada persamaan 2.4

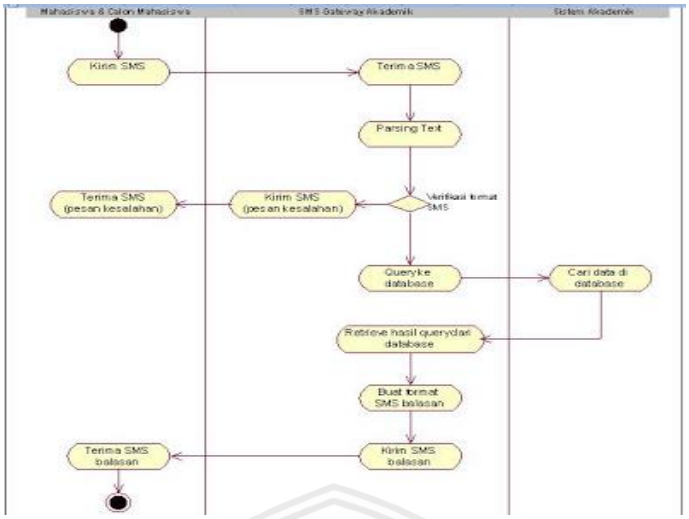
$$W1(P_i) = \sum(weight\ of\ the\ edges\ traversed\ by\ P_i) \quad (2.4)$$

6. Total *weight of each path,* seperti pada persamaan 2.5 sebagai berikut:

$$TW(P_i) = W(P_i) + CV(P_i) + N(P_i) + D(P_i) + W1(P_i) \quad (2.5)$$

2.4.2 UML Activity Diagram

Activity diagram menggambarkan alur yang berurutan dari aktivitas *use case*. Diagram ini juga dapat digunakan untuk memodelkan logika dengan suatu sistem (Whitten & Bentley, 2007). Contoh gambar *activity diagram* pada Gambar 2.6 dan notasi pada *activity diagram* dijelaskan pada Tabel 2.5.



Gambar 2.6 *Activity Diagram*

(Sumber : Whitten & Bentley, 2007)

Tabel 2.5 Notasi pada *Activity Diagram*

Notasi	*Description*
Initial node 	Divisualisasikan dalam bentuk bulatan yang menjelaskan awal pada sebuah proses.
Activity 	Divisualisasikan dalam bentuk segiempat dengan sudut tumpul yang menjelaskan tugas yang harus dikerjakan.
Panah (*control flow*) 	Divisualisasikan dengan panah membentuk garis horizontal yang menjelaskan sasaran untuk mengawali proses atau kegiatan.
Diamond (decision node) 	Divisualisasikan dalam bentuk belah ketupat yang menjelaskan sebuah keputusan pada *activity* seeprit *if* (pengkondisian).
Activity final node 	Divisualisasikan bentuk bulat dengan garis tepi yang menjelaskan akhir pada sebuah proses.

2.5 Teknologi Pengembangan Sistem

2.5.1 XML

eXtensible Markup Language (XML) yang dikembang oleh *World Wide Web Consortium* (W3C) pada tahun 1996. XML adalah bahasa yang digunakan digunakan untuk mempresentasikan dan memanipulasi data tanpa tergantung pada platform tertentu (Anshori, 2012). XML didefinisikan dengan menggunakan elemen yang ditandai dengan tag "<>" yaitu diawali dengan tag "<" dan diakhiri dengan tag ">". Dokumen XML memiliki beberapa jenis, diantaranya adalah sebagai berikut :

1. *Invalid document*, mendefinisikan struktur dokumen dengan tidak mengikuti aturan penulisan dari spesifikasi XML.
2. *Valid document*, mendefinisikan struktur dokumen dengan mengikuti aturan dari spesifikasi XML dan aturan DTD (*Document Type Definition*).
3. *Well-formed document*, mendefinisikan struktur dokumen dengan mengikuti dari spesifikasi XML tetapi tidak mengikuti aturan DTD (*Document Type Definition*)

XML memiliki kelebihan yaitu untuk menyederhanakan pertukaran data pada sistem dan format data yang berbeda. Terdapat suatu fungsi dari XML untuk menyediakan cara bertukar informasi tentang metadata yang dimana disebut dengan XMI (*XML Metadata Interchange*). XMI bertujuan untuk memudahkan programer menggunakan *Unified Modeling Language* (UML) agar dapat bertukar model data dengan yang lainnya (Putri, 2014).

2.5.2 *Parsing* XML

Umumnya parser adalah bagian kecil dari program yang mengambil input sebagai data dan menciptakan cara bagi program untuk menggunakan xml. Ini memainkan peran penting dalam membaca, mendeteksi pembentukannya dengan baik dan memvalidasi xml dengan bantuan skema xml (Tej *et al.*, 2011). *Parser* adalah sejenis compiler yang berguna untuk mengenali, memvalidasi atau meng-*compile* tag dalam suatu file. Parser mempunyai ukuran yang besar karena menampilkan banyak kode dan persyaratan runtime serta membutuhkan memori yang besar. Menurut Anshori (2012) ada tiga jenis parser yang mendasar dan jenis yang akan dipilih tergantung pada bagaimana aplikasi dibuat untuk berperilaku dan jenis dokumen apa yang akan di-*parsing*.

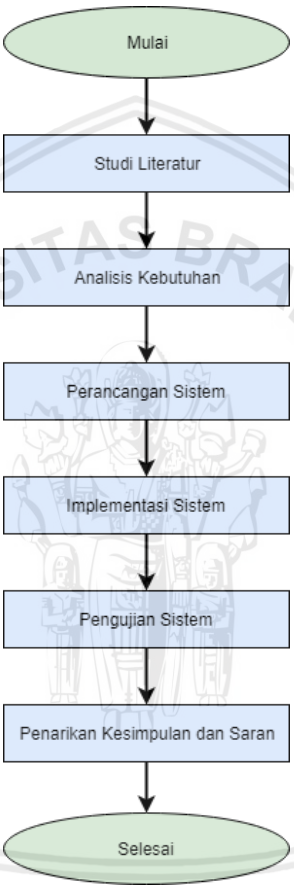
1. *Parser model* : membaca seluruh dokumen dan menciptakan representasi dari dokumen dalam memori. Model *parsing* menggunakan memori lebih besar dari jenis parser lainnya.
2. *Parser push* : membaca seluruh dokumen. Contoh : Operasi SAX API
3. *Parser pull* : membaca sedikit dokumen.

2.5.3 C#

C# (C Sharp) merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft dan bahasa C# juga telah distandarisasi secara internasional oleh ECMA (Jauhari, Hamidin & Rahmatuloh, 2017). C# adalah bahasa pemrograman multiparadigma yang sangat populer dan tersebar luas di bidang pengembangan karena kesederhanaan, fleksibilitas dan produktivitasnya. C# dipengaruhi oleh tiga bahasa pemrograman yang paling banyak digunakan yaitu C, C ++ dan Java. Menurut Alomari (2015) C# adalah bahasa pemrograman yang efisien meningkatkan produktivitas programmer dengan menyederhanakan proses pengembangan dan mengurangi kemungkinan kesalahan pemrograman yang paling sering terjadi. C# juga mempunyai fitur penerapan metadata yang dapat membantu dalam proses pemantauan dan memastikan pemetaan yang benar dari desain ke implementasi. C# memiliki sifat berorientasi komponen sehingga memudahkan proses pengelolaan dan pengembangan sistem kompleks sebagai sekelompok komponen yang saling berinteraksi.

BAB 3 METODOLOGI

Pada bab metodologi penelitian ini memaparkan metode yang akan digunakan pada pengembangan kakas bantu penentuan prioritas *test scenario*. Metodologi penelitian yang dilakukan dalam perancangan sistem menggunakan model *waterfall*. Diagram pada kakas bantu penentuan prioritas *test scenario* terdiri dari enam tingkatan yaitu studi literatur, analisis kebutuhan, perancangan sistem, implementasi sistem, pengujian sistem, serta penarikan kesimpulan dan saran seperti pada Gambar 3.1 sebagai berikut:



Gambar 3.1 Diagram Metodologi Penelitian

3.1 Studi Literatur

Pada penelitian ini memerlukan studi literatur dari dasar teori yang telah dibahas pada Bab 2 yang berhubungan dengan kakas bantu untuk penentuan prioritas *test scenario* berdasarkan UML *activity diagram*. Dasar teori pada penelitian yang diperoleh dari referensi-referensi buku, *journal, article, journal conference,* dan penelitian yang terkait. Dari referensi-referensi tersebut digunakan sebagai pengetahuan dasar melakukan analisis, perancangan, dan implementasi serta pengujian dalam tahap penelitian.

3.2 Analisis Kebutuhan

Analisis kebutuhan atau r*equirement analysis* mencakup penentuan kebutuhan atau kondisi yang harus dipenuhi dalam pembuatan perangkat lunak. Proses elisitasi kebutuhan pada penelitian kakas bantu untuk penentuan prioritas *test scenario* yaitu menganalisis sebuah penelitian Bhuyan, Ray, & Das (2017) yang berjudul *Test Scenario Prioritization Using UML Use Case and Activity Diagram*. Kemudian kebutuhan dideskripsikan menjadi kebutuhan yang lebih spesifik. Proses spesifikasi kebutuhan didapatkan yaitu kebutuhan fungsional. Pada setiap kebutuhan fungsional memiliki batasan-batasan pada sistem yang akan dikembangkan. Setelah menentukan kebutuhan maka akan dilakukan pemodelan dalam bentuk *use case diagram* dan dijabarkan dalam bentuk *use case scenario*.

3.3 Perancangan Sistem

Perancangan sistem akan membahas tentang rancangan sistem yang akan dikembangkan. Perancangan sistem dilakukan setelah semua kebutuhan didefinisikan pada tahapan rekayasa perangkat lunak dan sebagai pedoman untuk implementasi dan pengujian sistem. Perancangan sistem dibagi menjadi tiga perancangan yaitu sebagai berikut :

1. Perancangan Arsitektur
   Pada perancangan arsitektur dilakukan pemodelan menggunakan sequence diagram dan *class* diagram. Sequence diagram untuk menggambarkan bagaimana objek berinteraksi antara aktor dan sistem. Diagram *class* menggambarkan *class-class* yang digunakan dan menggambarkan hubungan antar *class* yang terlibat.
2. Perancangan Komponen
   Pada perancangan ini akan diberikan beberapa contoh algoritma utama pada sistem yang diambil dari *class*. Algoritma-algoritma tersebut akan dituliskan dalam bentuk pseudocode.
3. Perancangan Antarmuka
   Pada bagian perancangan ini mendeskripsikan rancangan antarmuka dalam sistem dalam bentuk tata letak fitur yang disediakan oleh sistem yang dibangun dan kemudahan pengguna dalam penggunaan sistem. Kemudian akan dari rancangan antarmuka akan menjadi implementasi antarmuka sistem.

3.4 Implementasi Sistem

Pada implementasi sistem merupakan tahapan yang mengacu pada perancangan sistem. Pada penelitian ini sistem akan diimplementasikan dengan menggunakan C#. Proses pada tahapan implementasi sistem diantaranya adalah:

1. Implementasi Spesifikasi Pengembangan Sistem
   Pada tahapan ini implementasi dilakukan pada spesifikasi sistem yang digunakan untuk membangun sebuah aplikasi ini.
2. Implementasi Logika Program
   Implementasi logika program mendeskripsikan algoritma-algoritma dalam sistem. Logika Program yang diimplementasikan pada sistem menggunakan bahasa pemrograman C#.
3. Implementasi Antarmuka
   Implementasi antarmuka berdasarkan rancangan antarmuka diimplementasikan menggunakan Unity3d.

3.5 Pengujian Sistem

Pengujian sistem memiliki tujuan untuk memastikan apakah sistem yang dibangun telah memenuhi dan apakah sudah sesuai dengan kebutuhan yang telah didefinisikan sebelumnya. Pengujian juga berfungsi untuk meminimalisir *bug* yang ada dalam sistem. Menurut Pressman (2010) terdapat beberapa strategi pengujian yang dilakukan diantaranya adalah:

1. Pengujian Unit
   Pengujian unit dilakukan untuk menguji unit dalam sistem yang dapat berupa komponen, *class*, atau objek dari perangkat lunak pada perancangan. Pengujian unit ini menggunakan teknik pengujian yaitu *whitebox testing* dengan metode pengujian *basis path testing*.
2. Pengujian Validasi
   Pengujian validasi dilakukan untuk melakukan pengecekan bahwa semua fungsi berjalan dengan benar dan tepat. Pengujian validasi menggunakan teknik *black-box testing*. Pengujian dilakukan untuk menemukan kesalahan-kesalahan terhadap sistem dan memastikan kebutuhan yang telah ditetapkan dengan masukan yang diberikan dengan keluaran yang dihasilkan tersebut apakah sudah memenuhi sesuai yang diharapkan apa tidak.

3.6 Penarikan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan perangkat lunak mulai dari tahapan studi literatur, analisis kebutuhan, perancangan sistem, implementasi sistem dan pengujian pada sistem. Sehingga akan diperoleh kesimpulan untuk menjawab rumusan masalah. Setelah kesimpulan kemudian ada saran dengan tujuan untuk memperbaiki kesalahan – kesalahan serta menyempurnakan penulisan.

BAB 4 ANALISIS KEBUTUHAN

Pada tahapan pertama yang harus dilakukan dalam proses pengembangan perangkat lunak adalah analisis kebutuhan. Tahapan analisis kebutuhan akan menentukan kebutuhan apa saja yang harus dipenuhi oleh sistem dalam pengembangan sistem sehingga menentukan sistem tersebut dapat dikatakan berhasil. Berhasil dengan arti sistem tersebut sudah dapat mencapai keinginan yang diharapkan sesuai dengan kebutuhan sistem. Pada analisis kebutuhan hal pertama yang dilakukan adalah mengidentifikasi aktor, menentukan kebutuhan fungsional sistem. Setelah aktor dan kebutuhan fungsional didapatkan maka selanjutnya melakukan pemodelan *use case diagram* dan dari setiap *use case* kemudian dijabarkan dalam bentuk *use case scenario.*

4.1 Gambaran Umum Sistem

Sistem kakas bantu penentuan prioritas *test scenario* berdasarkan UML *activity diagram* memiliki tujuan agar dapat membantu pengembang perangkat lunak dalam menentukan prioritas pengujian perangkat lunak. Berkas *activity diagram* dengan ekstensi xml adalah sebagai data masukan yang akan digunakan dalam sistem. Pada berkas *activity diagram* tersebut akan menghasilkan jalur independen yang kemudian akan dihitung dari setiap jalur independen.

Sistem mulai bekerja pada saat melakukan *parsing* terhadap berkas *activity diagram* dengan ekstensi XML yang telah dimasukkan oleh pengguna. *Parsing* digunakan untuk memperoleh data dari *activity diagram* berupa *control flow graph* dan jalur independen. Dari setiap jalur independen akan dihitung dengan berdasarkan *weighting factors*. Kemudian akan diurutkan berdasarkan nilai tertinggi dari hasil perhitungan. Keluaran yang akan ditampilkan sistem antara lain jalur independen dengan masing-masing nilai yang sudah dilakukan perhitungan dan jenis dari *main scenario* atau *alternative scenario*.

4.2 Identifikasi Aktor

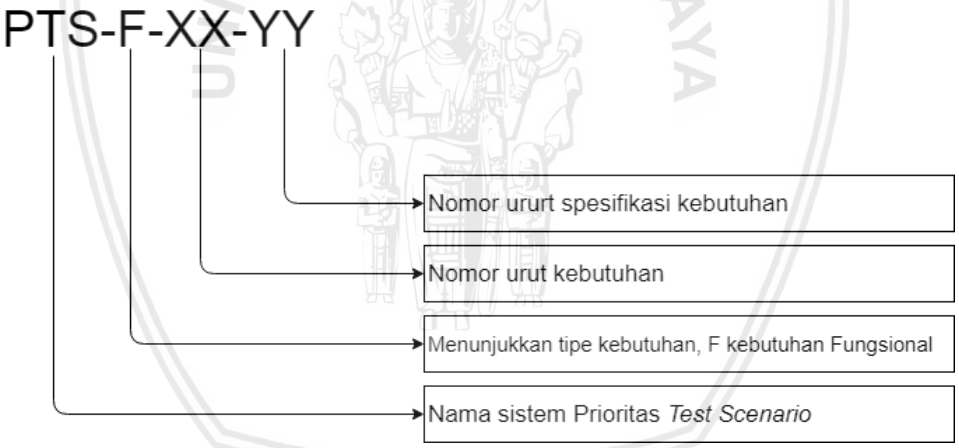
Aktor adalah orang atau sistem yang berinteraksi dengan sistem. Interaksi aktor merupakan sebuah masukan informasi dari pengguna ke dalam sistem ataupun aktor disini bisa menerima hasil keluaran informasi dari sebuah sistem. Pada tahapan identifikasi aktor adalah tahap untuk menentukan aktor yang menjalankan sistem. Pada Tabel 4.1 terdapat aktor yang akan terlibat dalam sistem yang akan dibangun beserta deskripsinya.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi Aktor
Pengguna	Pengguna merupakan aktor yang menjalankan sistem dan dapat menjalankan semua fitur yang ada pada sistem. Pengguna adalah seorang developer yang membuat atau mempunyai berkas *activity diagram* dan mengerti apa yang ada di dalam berkas *activity diagram* tersebut.

4.3 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan tentang layanan sistem yang harus disediakan, bagaimana sistem harus bereaksi terhadap masukan dan bagaimana sistem harus dalam situasi tertentu (Sommerville, 2011). Setiap kebutuhan akan diberikan kode yang berbeda dengan aturan penomoran PTS-F-XX-YY, dimana PTS menandakan nama sistem, F memiliki arti Fungsional, XX menandakan nomor urut kebutuhan dan YY menandakan nomor urut spesifikasi kebutuhan seperti pada Gambar 4.1 dan kebutuhan fungsional dijelaskan pada Tabel 4.2 sampai Tabel 4.4 sebagai berikut:



Gambar 4.1 Penomoran Kebutuhan Fungsional

Tabel 4.2 Kebutuhan Fungsional

No	Kode Kebutuhan	Kebutuhan	Nama Use Case
1	PTS-F-01	Sistem harus dapat menerima masukan berkas *activity diagram*.	*Input* berkas *activity diagram*
1.1	PTS-F-01-01	Berkas *activity diagram* hanya berkas dengan ekstensi ".xml"	

Tabel 4.3 Kebutuhan Fungsional (Lanjutan)

No	Kode Kebutuhan	Kebutuhan	Nama Use Case
2	PTS-F-02	Sistem harus dapat mem-*parsing* berkas *activity diagram* yang telah dimasukkan	*Parsing* XML
2.1	PTS-F-02-01	Sistem harus dapat menampilkan *control flow graph* dari berkas *activity diagram.*	
2.2	PTS-F-02-02	Sistem harus dapat memperlihatkan jalur independen dari berkas *activity diagram.*	
3	PTS-F-03	Sistem harus dapat melakukan perhitungan pada hasil *parsing* berkas *activity diagram* berdasarkan *weighting factors.*	Hitung Prioritas
3.1	PTS-F-03-01	Sistem harus dapat memberikan pilihan untuk menentukan main activity atau bukan pada setiap node.	
3.2	PTS-F-03-02	Sistem harus dapat memberikan pilihan untuk menentukan tipe kopling dari satu node ke node yang lain.	
3.3	PTS-F-03-03	Sistem harus dapat menghitung *weight of each path* pada setiap jalur independen.	
3.4	PTS-F-03-04	Sistem harus dapat menghitung *complexity value of each path* pada setiap jalur independen.	
3.5	PTS-F-03-05	Sistem harus dapat menghitung *node coverage by each path* pada setiap jalur independen.	
3.6	PTS-F-03-06	Sistem harus dapat menghitung *decision node coverage by each path* pada setiap jalur independen.	
3.7	PTS-F-03-07	Sistem harus dapat menghitung *weight of each path* berdasarkan tipe kopling pada setiap jalur independen.	
3.8	PTS-F-03-08	Sistem harus dapat menghitung *total weight of each path* dan menampilkan hasil prioritas.	

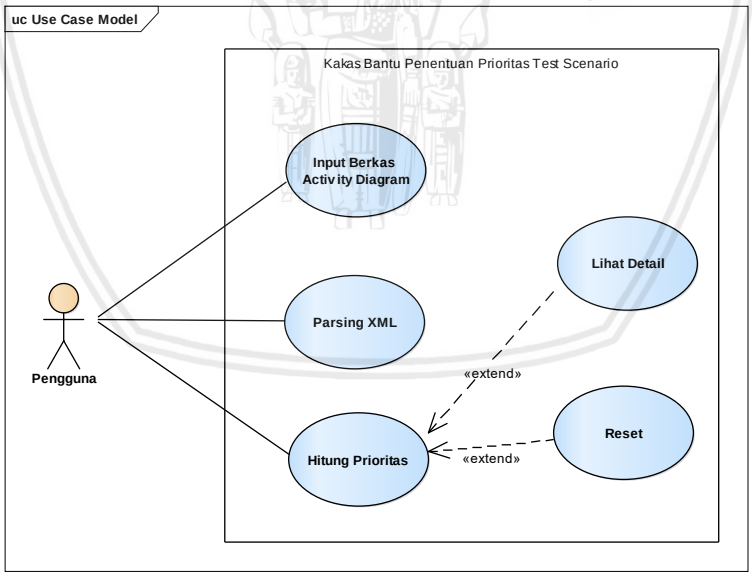
Tabel 4.4 Kebutuhan Fungsional (Lanjutan)

No	Kode Kebutuhan	Kebutuhan	Nama Use Case
4	PTS-F-04	Sistem harus dapat menampilkan informasi detail dari hasil perhitungan berdasarkan *weighting factors.*	Lihat Detail
5	PTS-F-05	Sistem harus dapat menghapus seluruh data hasil perhitungan dan menampilkan halaman utama.	Reset

4.4 Pemodelan Kebutuhan

4.4.1 Use Case Diagram

Use case diagram merupakan diagram yang memvisualisasikan bagaimana sistem berinteraksi dengan lingkungannya, atau dapat memvisualisasikan interaksi aktor eksternal sistem dengan sistem (Sommerville, 2011). Use case diagram digunakan untuk menggambarkan kebutuhan dari sistem dan dibuat berdasarkan hasil analisis kebutuhan. Pada Gambar 4.3 menjelaskan bahwa pengguna menjalankan seluruh fungsionalitas sistem kakas bantu penentu prioritas test scenario yaitu *Input* Berkas *Activity Diagram*, *Parsing* XML, Hitung Prioritas, Lihat Detail, dan Reset.



Gambar 4.2 *Use Case Diagram* Kakas Bantu Penentuan Prioritas *Test Scenario*

4.4.2 Use Case Scenario

Use case scenario merupakan penjabaran atau penjelasan yang lebih detail dari setiap *use case* pada *use case diagram*.

4.4.2.1 Use Case Scenario Input Berkas Activity Diagram

Use case scenario input berkas *activity diagram* adalah alur untuk memasukkan berkas *activity diagram* kedalam sistem dan alur tersebut akan dijelaskan pada Tabel 4.5.

Tabel 4.5 *Use Case Scenario* Input Berkas *Activity Diagram*

Item	Deskripsi
Nomor *Use Case*	PTS-F-01
Nama *Use Case*	Input Berkas *Activity Diagram*
Objektif	Pengguna memasukkan berkas dengan ekstensi ".xml"
Aktor	Pengguna
Kondisi sebelum	Sistem menampilkan halaman utama sistem
Alur utama	1. Pengguna menekan tombol Choose a File 2. Sistem menampilkan *window* pengelola berkas 3. Pengguna memilih berkas *activity diagram* 4. Pengguna menekan tombol open pada *window* pengelola berkas Sistem memindai berkas yang dimasukkan.
Alur alternatif	1. Jika pengguna menekan tombol cancel pada *pop up* pengelola berkas maka berkas tidak dimasukkan dalam sistem Jika pada *pop up* pengelola berkas tidak terdapat berkas berekstensi ".xml" maka akan menampilkan tidak ada data yang dicari
Kondisi sesudah	Sistem menampilkan nama berkas dengan ekstensi xml yang telah dimasukkan dan menampilkan tombol Parsing XML

4.4.2.2 Use Case Scenario Parsing XML

Use case scenario parsing XML adalah alur untuk melakukan parsing agar dapat menampilkan control flow graph dan jalur independen dan alur tersebut akan dijelaskan pada Tabel 4.6 sampai Tabel 4.7.

Tabel 4.6 *Use Case Scenario Parsing* XML

Item	Deskripsi
Nomor *Use Case*	PTS-F-02
Nama *Use Case*	*Parsing* XML
Objektif	Pengguna melakukan *parsing* XML
Aktor	Pengguna
Kondisi sebelum	Sistem memasukkan berkas activity diagram dengan ekstensi ".xml"

Tabel 4.7 *Use Case Scenario Parsing* XML (Lanjutan)

Item	Deskripsi
Alur utama	1. Pengguna menekan tombol Parsing XML 2. Sistem melakukan *parsing* terhadap berkas *activity diagram* dengan ekstensi ".xml"
Alur alternatif	Jika berkas yang dimasukkan dengan ekstensi xml tetapi bukan activity diagram maka sistem tidak dapat melakukan parsing
Kondisi sesudah	Sistem menampilkan *control flow graph* dan jalur independen

4.4.2.3 *Use Case Scenario* Hitung Prioritas

Use case scenario hitung prioritas adalah alur untuk melakukan perhitungan pada sistem dan alur tersebut akan dijelaskan pada Tabel 4.8.

Tabel 4.8 *Use Case Scenario* Hitung Prioritas

Item	Deskripsi
Nomor *Use Case*	PTS-F-03
Nama *Use Case*	Hitung Prioritas
Objektif	Pengguna melakukan perhitungan pada bekas *activity diagram*
Aktor	Pengguna
Kondisi sebelum	Sistem melakukan *parsing* terhadap berkas *activity diagram,* serta menampilkan *control flow graph* dan jalur independen
Alur utama	1. Pengguna menekan tombol Hitung Prioritas 2. Sistem akan menampilkan halaman untuk menentukan *main activity* atau tidak pada setiap node 3. Pengguna menentukan *main activity* pada sebuah node 4. Pengguna menekan tombol *save* 5. Sistem menampilkan halaman untuk menentukan tipe kopling 6. Pengguna menentukan tipe kopling dari salah satu node ke node yang lain 7. Pengguna menekan tombol *save* 8. Sistem melakukan perhitungan pada setiap faktor pembobotan
Alur alternatif	1. Pengguna menekan tombol close pada saat menentukan *main activity* 2. Sistem menampilkan halaman parsing XML 3. Pengguna menekan tombol close pada saat menentukan tipe kopling 4. Sistem menampilkan halaman parsing XML
Kondisi sesudah	Sistem menampilkan hasil perhitungan dengan menggunakan metode *weighting factors* atau faktor pembobotan

4.4.2.4 *Use Case Scenario* Lihat Detail

Use case scenario lihat detail adalah alur untuk melihat secara detail hasil perhitungan dari lima faktor pembobotan pada sistem dan alur tersebut akan dijelaskan pada Tabel 4.9.

Tabel 4.9 *Use Case Scenario* Lihat Detail

Item	Deskripsi
Nomor *Use Case*	PTS-F-04
Nama *Use Case*	Lihat Detail
Objektif	Pengguna melihat detail perhitungan pada setiap faktor dari lima faktor pembobotan
Aktor	Pengguna
Kondisi sebelum	Sistem menampilkan hasil perhitungan skenario prioritas serta menampilkan tombol detail dan reset
Alur utama	1. Pengguna menekan tombol Detail 2. Sistem memproses detail hasil perhitungan
Alur alternatif	-
Kondisi sesudah	Sistem menampilkan hasil perhitungan dari setiap faktor pembobotan dalam bentuk table

4.4.2.5 *Use Case Scenario* Reset

Use case scenario reset adalah alur untuk mengembalikan halaman ke halaman awal awal pada sistem dan alur tersebut akan dijelaskan pada Tabel 4.10.

Tabel 4.10 *Use Case Scenario* Reset

Item	Deskripsi
Nomor *Use Case*	PTS-F-05
Nama *Use Case*	Reset
Objektif	Menghapus semua data yang sudah dilakukan perhitungan dan menampilkan ke halaman utama
Aktor	Pengguna
Kondisi sebelum	Sistem menampilkan hasil perhitungan skenario prioritas serta menampilkan tombol detail dan reset
Alur utama	1. Pengguna menekan tombol Cancel 2. Sistem akan melakukan proses penghapusan data dan melakukan proses untuk menampilkan halaman utama.
Alur alternatif	-
Kondisi sesudah	Sistem menghapus seluruh data hasil perhitungan dan menampilkan halaman utama

BAB 5 PERANCANGAN DAN IMPLEMENTASI SISTEM

5.1 Perancangan Sistem

Perancangan sistem merupakan tahapan pengembangan perangkat lunak setelah menganalisis kebutuhan pengguna. Dimana setelah melakukan analisis kebutuhan kemudian kebutuhan yang didapatkan akan dimodelkan dalam bentuk perancangan. Pada proses pengembangan kakas bantu penentuan prioritas *test scenario* berdasarkan *activity diagram* tahap perancangan dibagi menjadi beberapa bagian, yaitu perancangan arsitektur, perancangan komponen, dan perancangan antarmuka.

5.1.1 Perancangan Arsitektur

Perancangan arsitektur dilakukan dengan membuat *sequence diagram* dan *class diagram*. Pada perancangan ini akan dijelaskan lebih detail tentang sequence diagram dan class diagram, sequence diagram yang akan dijelaskan yaitu input berkas *activity diagram*, parsing XML, dan hitung prioritas, sedangkan class diagram akan diigambarkan *class-class* sebagai penyusun sistem.

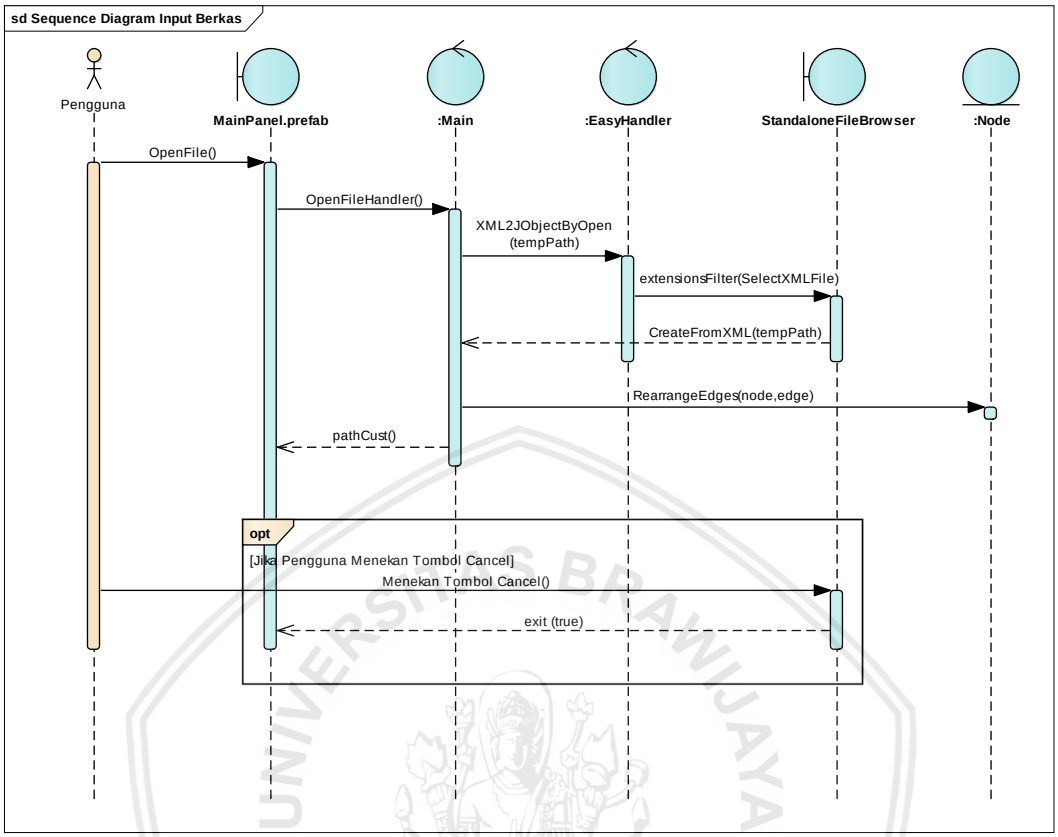
5.1.1.1 Sequence Diagram

Pada kakas bantu penentuan prioritas *test scenario* berdasarkan *activity diagram* dijelaskan dengan tiga fitur utama pada *sequence diagram* yaitu *input* berkas *activity diagram*, parsing XML, dan Hitung Prioritas.

1. *Sequence Diagram Input* Berkas Activity Diagram

Pada Gambar 5.1 merupakan perancangan *sequence diagram* input berkas *activity diagram*. Pada sequence diagram input berkas *activity diagram* menunjukkan interaksi objek didalam sistem untuk memasukkan berkas activity diagram. *Sequence diagram* input berkas *activity diagram* terdapat lima objek diantaranya terdapat 2 boundary yaitu MainPanel dan StandaloneFileBrowser, dua controller yaitu Main dan EasyHandler, dan satu objek model yaitu Node.

Pada Sequence diagram input berkas activity diagram menjelaskan bahwa pengguna akan berinteraksi dengan objek boundary StandaloneFileBrowser untuk memilih berkas activity diagram dimana pada saat menampilkan File Browser sudah di filter hanya menampilkan berkas dengan ekstensi xml saja. Setelah memilih berkas activity diagram maka struktur-struktur pada berkas activity diagram tersebut akan disimpan pada objek Node. Sistem akan mengembalikan path *activity diagram* yang dipilih. Jika pengguna tidak ingin memilih berkas maka ada pilihan alternatif yaitu pengguna dapat menekan tombol cancel.

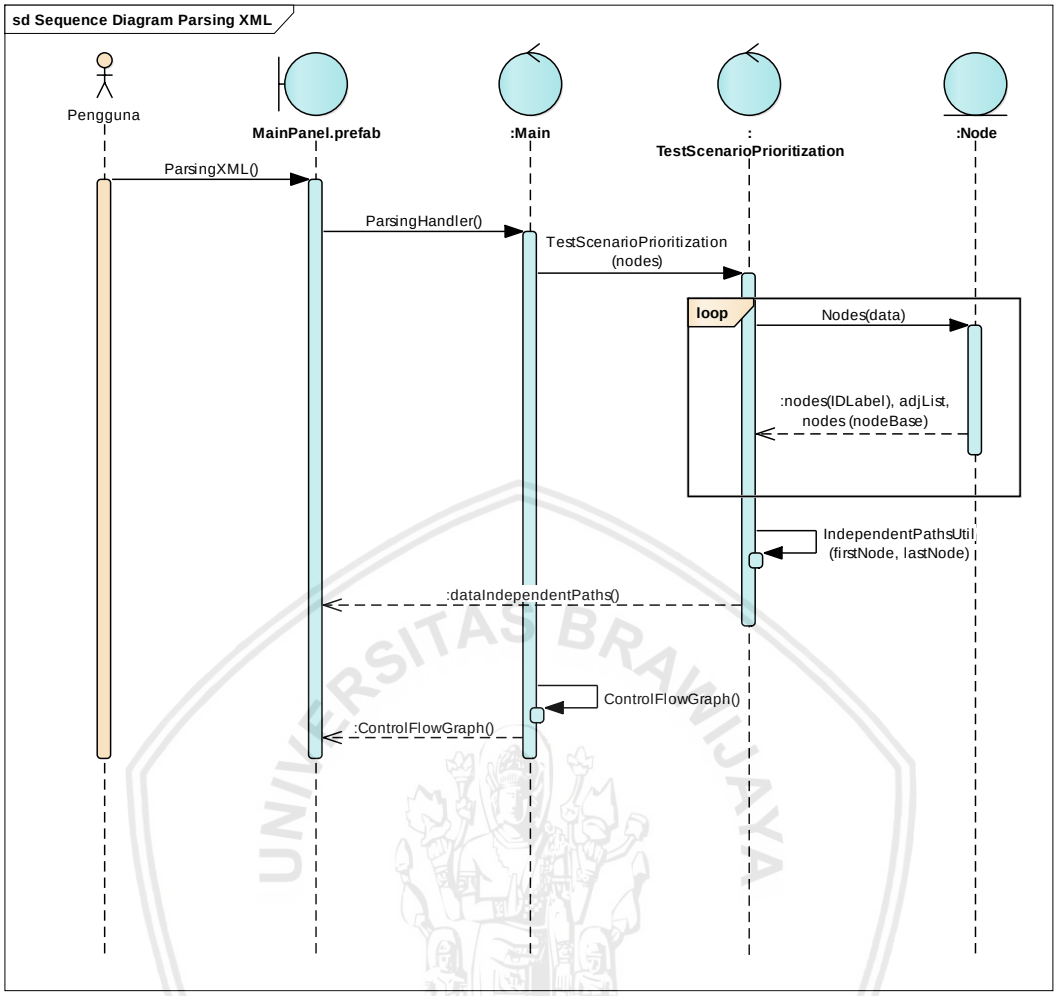


Gambar 5.1 Sequence Diagram Input Berkas *Activity Diagram*

2. *Sequence Diagram Parsing XML*

Pada Gambar 5.2 merupakan perancangan sequence diagram parsing xml. Pada sequence diagram parsing xml menunjukkan interaksi objek di dalam sistem untuk me-generate activity diagram dengan ekstensi xml diubah menjadi *control flow graph* dan jalur independen. *Sequence diagram* parsing xml terdapat empat objek diantaranya terdapat 1 boundary yaitu MainPanel, dua controller yaitu Main dan TestScenarioPrioritization, dan satu objek model yaitu Node.

Pada Sequence diagram parsing xml menjelaskan bahwa pengguna akan berinteraksi dengan objek boundary MainPanel. Objek TestScenarioPrioritization akan mengambil node dengan parameter IDLabel, adjList, dan mengambil nodeBase out dan nodeBase in. Kemudian objek TestScenarioPrioritization akan memanggil UpdateIndependentPaths untuk menampilkan jalur independen. Untuk menampilkan *control flow graph* maka objek main memanggil ControlFlowGraph. Jalur independen dan *control flow graph* akan ditampilkan pada boundary MainPanel.

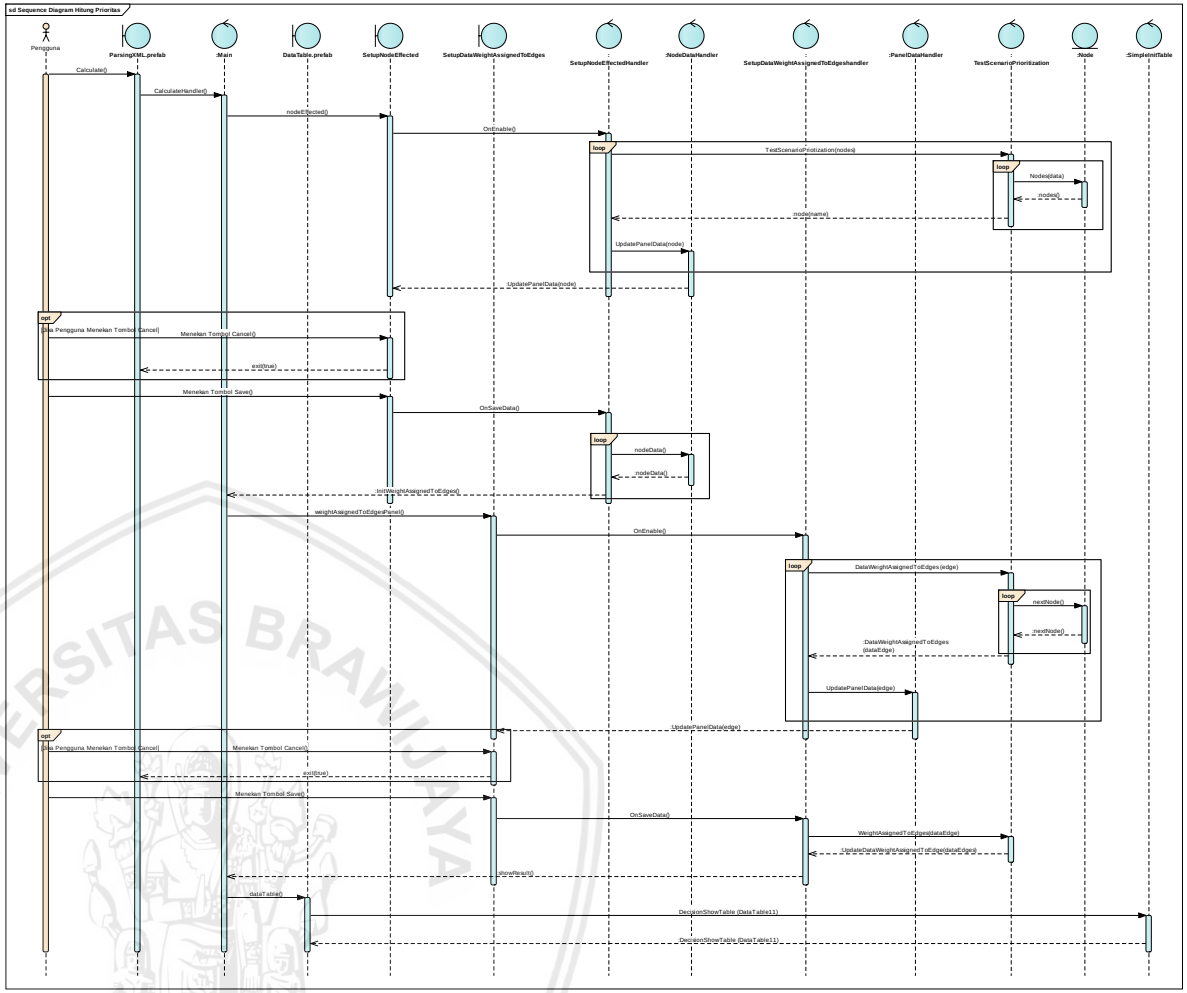


Gambar 5.2 Sequence Diagram Parsing XML

3. *Sequence Diagram* Hitung Prioritas

Pada Gambar 5.3 merupakan perancangan *sequence diagram* hitung prioritas. Pada *sequence diagram* hitung prioritas menunjukkan perhitungan pada berkas *activity diagram* yang sudah ditentukan jalur independennya kemudian dihitung pada setiap jalurnya. *Sequence diagram* hitung prioritas terdapat tiga belas objek yang terdiri dari empat boundary yaitu ParsingXML, SetupNodeEffected, SetupDataWeightAssignedToEdges, dan DataTable, tujuh controller yaitu Main, SetupNodeEffectedHandler, NodeDataHandler, SetupDataWeightAssignedToEdgeshandler, PanelDataHandler, SimpleInitTable, dan TestScenarioPrioritization, dan satu objek model yaitu Node.

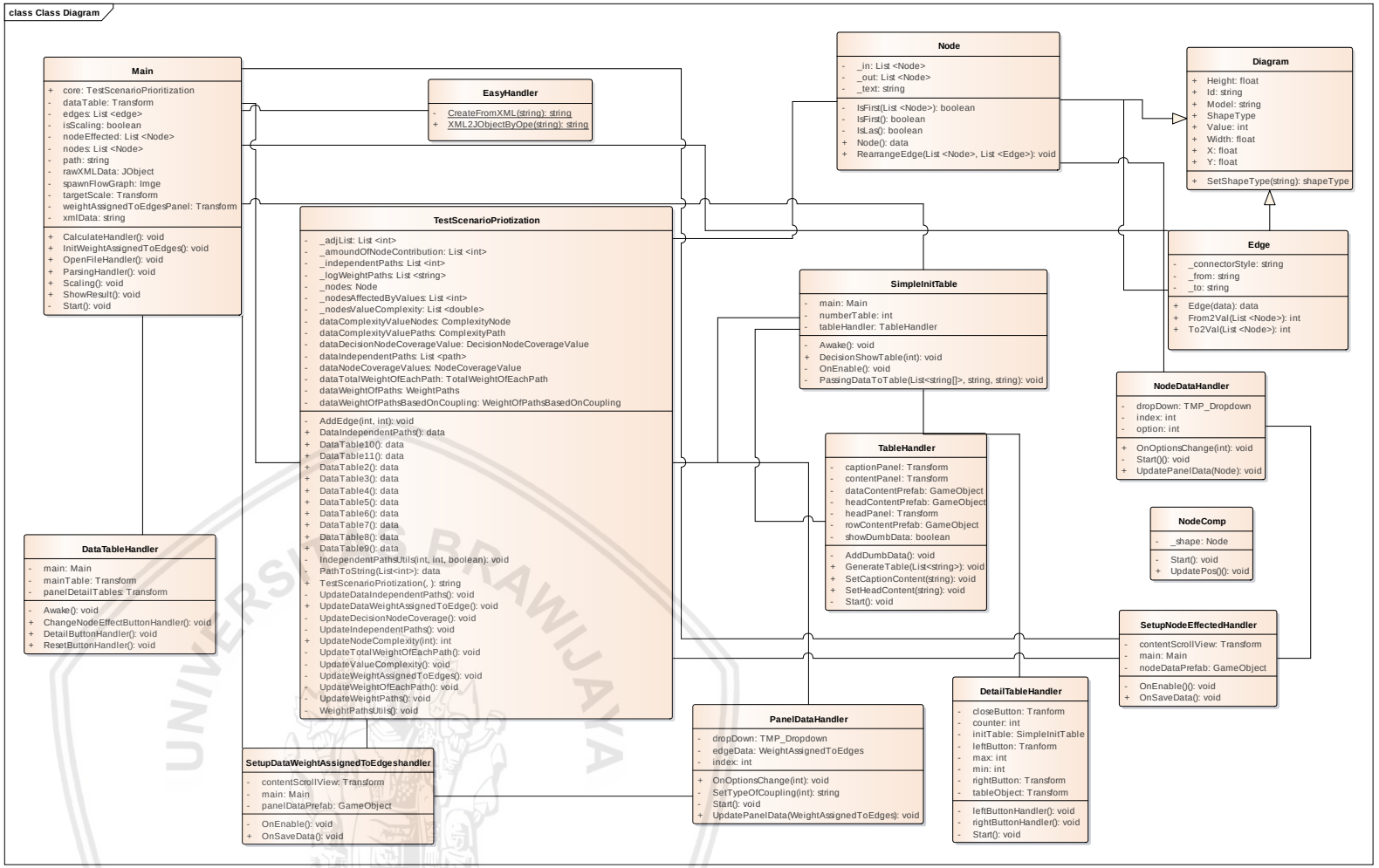
Pada *sequence diagram* hitung prioritas menjelaskan bahwa pengguna akan berinteraksi dengan objek boundary MainPanel, SetupNodeEffected, SetupDataWeightAssignedToEdges. Pada *sequence diagram* hitung prioritas objek SimpleInitTable akan menampilkan hasil perhitungan penentuan prioritas test scenario dalam bentuk tabel.



Gambar 5.3 Sequence Diagram Hitung Prioritas

5.1.1.2 Class Diagram

Class diagram pada kakas bantu penentuan prioritas *test scenario* ditunjukkan pada Gambar 5.4, *class diagram* menjelaskan struktur objek yaitu *class* serta keseluruhan *attribute* dan *method*. *Class diagram* dalam penelitian ini yaitu terdapat lima belas *class* yang saling berinteraksi, terdapat tiga *class* model dan dua belas *class controller*. Pada class model terdapat class dengan nama diagram yang merupakan induk dari *class* node dan *class* edge. Pada class *controller* berfungsi sebagai alur logika dalam menjalankan sistem yang terdiri dari class TestScenarioPriotization, SetupDataWeightAssignedToEdgeshandler, Main, EasyHandler, DataTableHandler, SimpleInitTable, TableHandler, NodeComp, DetailTableHandler, SetupNodeEffectedHandler, PanelDataHandler, NodeDataHandler.



Gambar 5.4 Class Diagram Penentuan Prioritas *Test Scenario*

5.1.2 Perancangan Komponen

Perancangan komponen akan menjelaskan rincian sub-sistem dari setiap komponen perangkat lunak, mendefinisikan rincian algoritma proses yang terjadi di dalam komponen. Pada perancangan komponen 3 fitur utama pada sistem akan dipilih 3 algoritma utama yaitu method "XML2JObjectByOpen, "DecisionShowTable", dan "UpdateTotalWeightOfEachPath".

5.1.2.1 Perancangan Komponen Method "*XML2JObjectByOpen*"

Pada Algoritme 1 menjelaskan perancangan komponen algoritme method XML2JObjectByOpen. Method XML2JObjectByOpen yang digunakan untuk menginputkan berkas activity diagram dengan ekstensi xml. Pada algoritme method XML2JObjectByOpen melakukan seleksi apabila berkas yang akan dimasukkan tidak terdapat berkas dengan ekstensi xml maka akan menampilkan hasil null atau menampilkan halaman direktori kosong, jika terdapat berkas dengan ekstensi xml maka akan menampilkan berkas terdapat berkas xml.

Nama class: EasyHandler

Nama Operasi: XML2JObjectByOpen

Algoritme1 : Pseudocode method XML2JObjectByOpen	
1	If tempPath sama dengan null atau tempPath sama dengan kosong
2	Instansiasi variabel extensions dengan parameter xml
3	tempPath <- StandaloneFileBrowser dan memanggil OpenFilePanel
4	Endif
5	Mengembalikan nilai CreateFromXML parameter tempPath

5.1.2.2 Perancangan Komponen Method "*DecisionShowTable*"

Pada Algoritme 2 menjelaskan perancangan komponen algoritme method DecisionShowTable. Method DecisionShowTable yang digunakan untuk memanggil hasil perhitungan dan menampilkan hasil perhitungan. Pada algoritme method DecisionShowTable melakukan seleksi jika case mempunyai nilai 2 maka akan memanggil method DataTable2 dan menampilkan hasil perhitungan dari method DataTable2 dalam bentuk tabel, jika case mempunyai nilai 3 maka akan memanggil method DataTable3 dan menampilkan hasil perhitungan dari method DataTable3 dalam bentuk tabel sampai pada case mempunyai nilai 11 maka akan memanggil method DataTable11 dan menampilkan hasil perhitungan dari method DataTable11 dalam bentuk tabel, jika case bernilai selain angka 2 sampai 11 maka sistem berakhir.

Nama class: SimpleInitTable

Nama Operasi: DecisionShowTable

Algoritme2 : Pseudocode method DecisionShowTable	
1	Switch dengan parameter index
2	Case bernilai 2
3	Memanggil method PassingDataToTable
4	Memanggil method DataTable2
5	Menampilkan "Table 1. Information complexity table"
6	Inisialisasi array dengan nilai 4

```
7                Membuat kolom dan Menampilkan “Node”
8                Membuat kolom dan Menampilkan “IN”
9                Membuat kolom dan Menampilkan “OUT”
10               Membuat kolom dan Menampilkan “IN x OUT”
11         Endcase
12      Case bernilai 3
13         Memanggil method PassingDataToTable
14            Memanggil method DataTable3
               Menampilkan “Table 2. Weight of paths based on
15    information complexity"
               Inisialisasi array dengan nilai 3
16               Membuat kolom dan Menampilkan “Independent paths”
17               Membuat kolom dan Menampilkan “Path description”
18               Membuat kolom dan Menampilkan “Weight of each path”
19         Endcase
20      Case bernilai 4
21         Memanggil method PassingDataToTable
22            Memanggil method DataTable4
23            Menampilkan “Table 3. Complexity value of nodes"
24            Inisialisasi array dengan nilai 3
25               Membuat kolom dan Menampilkan “Node”
26               Membuat kolom dan Menampilkan “Amount of node
27    contribution”
                 Membuat kolom dan Menampilkan “Complexity value of
28    each node”
         Endcase
29      Case bernilai 5
30         Memanggil method PassingDataToTable
31            Memanggil method DataTable5
32            Menampilkan “Table 4. Path complexity based on node
33    complexity value "
               Inisialisasi array dengan nilai 3
34               Membuat kolom dan Menampilkan “Independent paths”
35               Membuat kolom dan Menampilkan “Path description”
36               Membuat kolom dan Menampilkan “Complexity value of
37    each path”
         Endcase
38      Case bernilai 6
39         Memanggil method PassingDataToTable
40            Memanggil method DataTable6
41            Menampilkan “Table 5. Node coverage value"
42            Inisialisasi array dengan nilai 3
43               Membuat kolom dan Menampilkan “Independent paths”
44               Membuat kolom dan Menampilkan “Path description”
45               Membuat kolom dan Menampilkan “Node coverage of each
46    path”
         Endcase
47      Case bernilai 7
48         Memanggil method PassingDataToTable
49            Memanggil method DataTable7
50            Menampilkan “Table 6. Decision node coverage values"
51            Inisialisasi array dengan nilai 3
52               Membuat kolom dan Menampilkan “Independent paths”
53               Membuat kolom dan Menampilkan “Path description”
54               Membuat kolom dan Menampilkan “Decision node
55    coverage of each path”
         Endcase
56      Case bernilai 8
57         Memanggil method PassingDataToTable
58            Memanggil method DataTable8
59            Menampilkan “Table 7. Weight assigned to edges"
60            Inisialisasi array dengan nilai 3
61               Membuat kolom dan Menampilkan “Edge”
62               Membuat kolom dan Menampilkan “Type of coupling”
63               Membuat kolom dan Menampilkan “Weight”
```

```
64          Endcase
65       Case bernilai 9
66          Memanggil method PassingDataToTable
67             Memanggil method DataTable9
68             Menampilkan "Table 8. Weight of paths based on coupling
69    factor"
               Inisialisasi array dengan nilai 3
70                Membuat kolom dan Menampilkan "Independent paths"
71                Membuat kolom dan Menampilkan "Path description"
72                Membuat kolom dan Menampilkan "Weight of each path"
73          Endcase
74       Case bernilai 10
75          Memanggil method PassingDataToTable
76             Memanggil method DataTable10
77             Menampilkan "Table 9. Total weight of each path"
78             Inisialisasi array dengan nilai 3
79                Membuat kolom dan Menampilkan "Independent paths"
80                Membuat kolom dan Menampilkan "Path description"
81                Membuat kolom dan Menampilkan "Total weight of each
82    path"
          Endcase
83       Case bernilai 11
84          Memanggil method PassingDataToTable
85             Memanggil method DataTable11
86             Menampilkan "Table 10. Prioritized scenario"
87             Inisialisasi array dengan nilai 4
88                Membuat kolom dan Menampilkan "Scenario"
89                Membuat kolom dan Menampilkan "Path"
90                Membuat kolom dan Menampilkan "Path description"
91                Membuat kolom dan Menampilkan "Total weight of each
92    path"
          Endcase
93    End Switch
94
```

5.1.2.3 Perancangan Komponen Method "*UpdateTotalWeightOfEachPath*"

Pada Algoritme 3 menjelaskan perancangan komponen algoritme method UpdateTotalWeightOfEachPath. Method UpdateTotalWeightOfEachPath yang digunakan untuk menjumlahkan perhitungan dengan *weighting factors*. Pada algoritme method UpdateTotalWeightOfEachPath melakukan perulangan apabila nilai variabel i kurang dari independentPath maka menjumlahkan semua perhitungan dan disimpan pada variabel totalWeight. Sedangkan apabila variabel i lebih besar dari independentPath maka proses peulangan berakhir.

Nama class: TestScenarioPriotization

Nama Operasi: UpdateTotalWeightOfEachPath

```
Algoritme3 : Pseudocode method UpdateTotalWeightOfEachPath
1     Instansiasi dataTotalWeightOfEachPath
2     For variabel i = 0, i kurang dari _independentPaths
3        DataTotalWeightOfEachPath ditambah memanggil
      TotalWeightOfEachPath
4           Inisialisasi path = _independentPaths ke-i
5           totalWeight = dataWeightOfPaths ke-i memanggil weight
              +dataComplexityValueNodeske-i memanggil
      complexityValueNode
              +dataNodeCoverageValue ke-i memanggil nodeCoverage
              +dataDecisionNodeCoverageValue ke-i decisionoNode
              +dataWeightOfPathsBasedOnCoupling ke-i memanggil
      weightOfEachPath
```

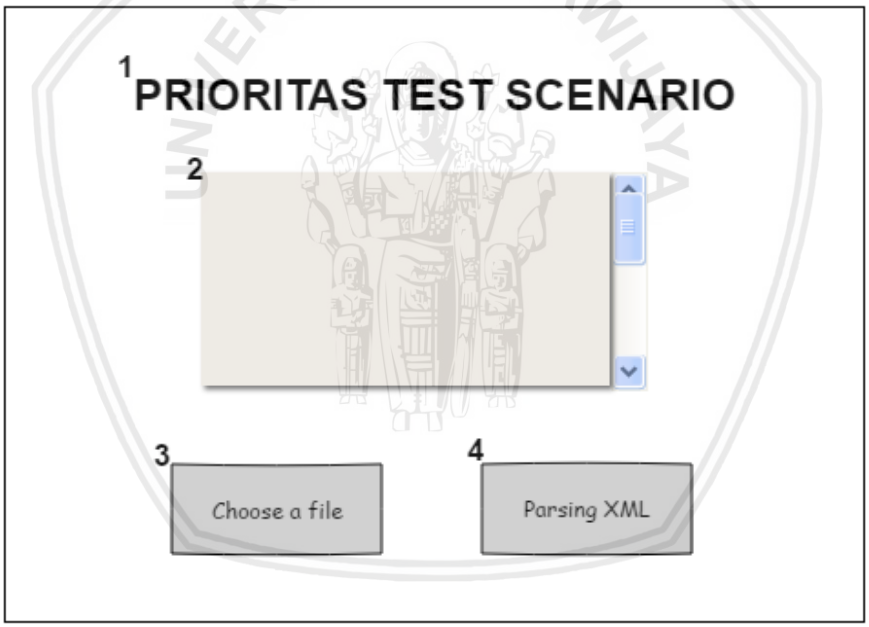
6	Endfor

5.1.3 Perancangan Antarmuka

Antarmuka pengguna merupakan sebuah tampilan dari sistem yang akan digunakan untuk berinteraksi oleh pengguna dengan sistem. Dalam perancangan antarmuka dilakukan dengan membuat sebuah tampilan kerangka yang akan diimplementasikan pada sistem dan untuk mendeskripsikan tata letak pada sistem.

5.1.3.1 Perancangan Antarmuka *Input* Berkas Activity Diagram

Pada Gambar 5.5 merupakan perancangan antarmuka untuk input berkas dan untuk penjelasan komponen dijelaskan pada Tabel 5.1 sampai Tabel 5.2. Pada Gambar 5.5 pengguna berhadapan dengan halaman antarmuka untuk memasukkan berkas yang berupa ekstensi xml yang selanjutnya akan di parsing. Pengguna akan menekan tombol Choose a file kemudian akan muncul pop up menu browse dan pengguna memilih berkas yang akan di parsing. Dan kemudian tampil untuk parsing berkas yang berekstensi xml tersebut.



Gambar 5.5 Perancangan Antarmuka Input Berkas *Activity Diagram*

Tabel 5.1 Penjelasan perancangan antarmuka input berkas *activity diagram*

No.	Nama Komponen	Tipe	Deskripsi
1	Judul Sistem	Label	Teks yang menampilkan judul
2	Nama Berkas	Pane	Untuk menampilkan nama Berkas *activity diagram* yang dimasukkan

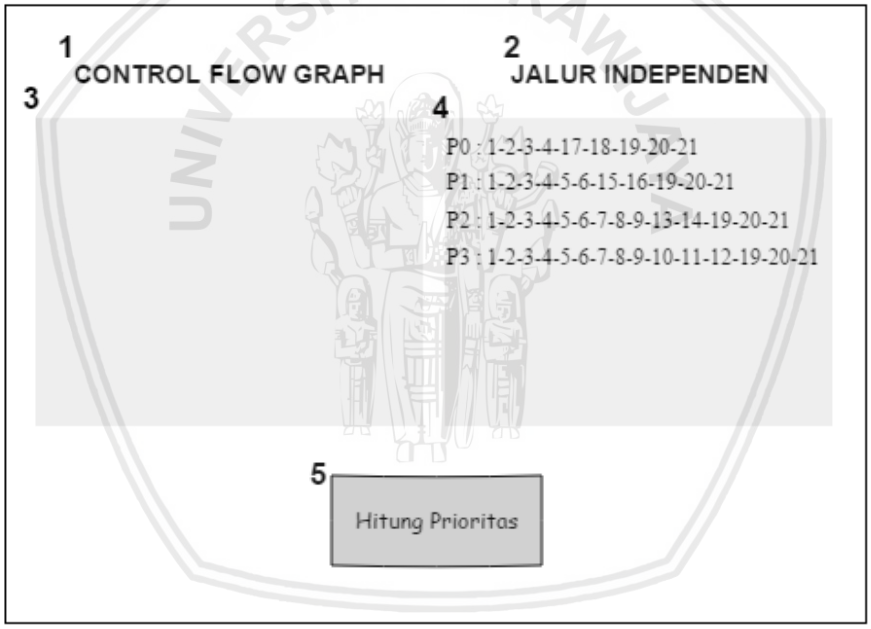
Tabel 5.2 Penjelasan perancangan antarmuka input berkas *activity diagram*

(Lanjutan)

No.	Nama Komponen	Tipe	Deskripsi
3	Tombol Choose a file	Button	Tombol yang berfungsi untuk menampilkan menu browse
4	Tombol Parsing XML	Button	Tombol yang berfungsi untuk melakukan parsing xml

5.1.3.2 Perancangan Antarmuka Parsing XML

Pada Gambar 5.6 merupakan perancangan antarmuka untuk parsing XML dan untuk penjelasan komponen dijelaskan pada Tabel 5.3 sampai Tabel 5.4. Pada Gambar 5.6 pengguna akan dihadapkan dengan halaman antarmuka untuk menampilkan hasil parsing XML dengan menekan tombol parsing pada antarmuka sebelumnya, kemudian ada tombol untuk melakukan perhitungan.



Gambar 5.6 Perancangan Antarmuka Parsing XML

Tabel 5.3 Penjelasan perancangan antarmuka parsing XML

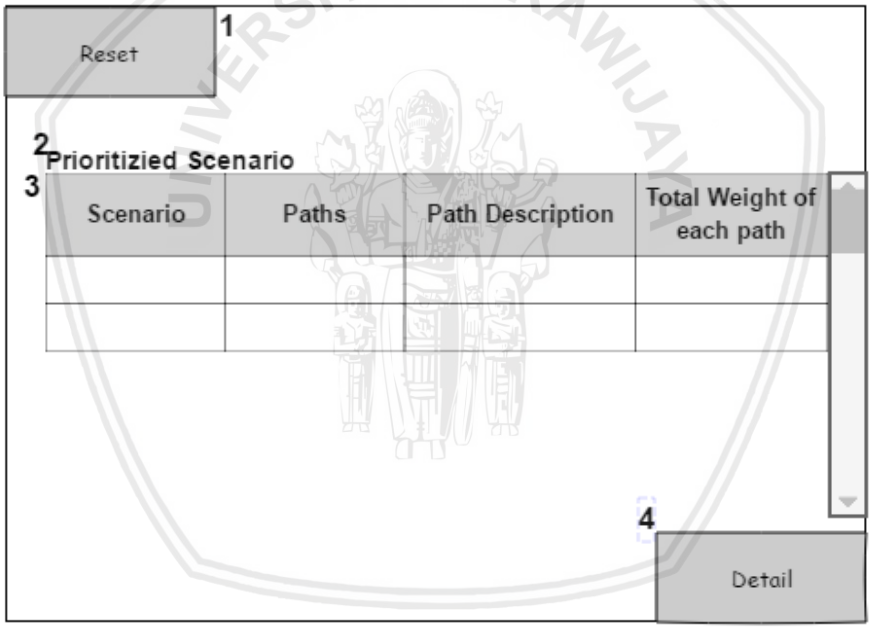
No.	Nama Komponen	Tipe	Deskripsi
1	Nama Flow Graph	Label	Teks yang menampilkan control flow graph
2	Nama Jalur Independent	Label	Teks yang menampilkan jalur independen

Tabel 5.4 Penjelasan perancangan antarmuka parsing XML (Lanjutan)

No.	Nama Komponen	Tipe	Deskripsi
3	Gambar Control Flow Graph	Pane	Menampilkan hasil dari activity diagram menjadi control flow graph
4	Jalur Independent	Pane	Menampilkan hasil jalur independen
5	Tombol Hitung Prioritas	Button	Tombol yang berfungsi untuk menghitung hasil parsing xml

5.1.3.3 Perancangan Antarmuka Hitung Prioritas

Pada Gambar 5.7 merupakan perancangan antarmuka untuk Hitung Prioritas dan untuk penjelasan komponen dijelaskan pada Tabel 5.5 sampai Tabel 5.6. Pada Gambar 5.7 pengguna akan dihadapkan dengan halaman antarmuka untuk menampilkan hasil perhitungan dan menampilkan prioritas pada pengujian skenario.



Gambar 5.7 Perancangan Antarmuka Hitung Prioritas

Tabel 5.5 Penjelasan perancangan antarmuka hitung prioritas

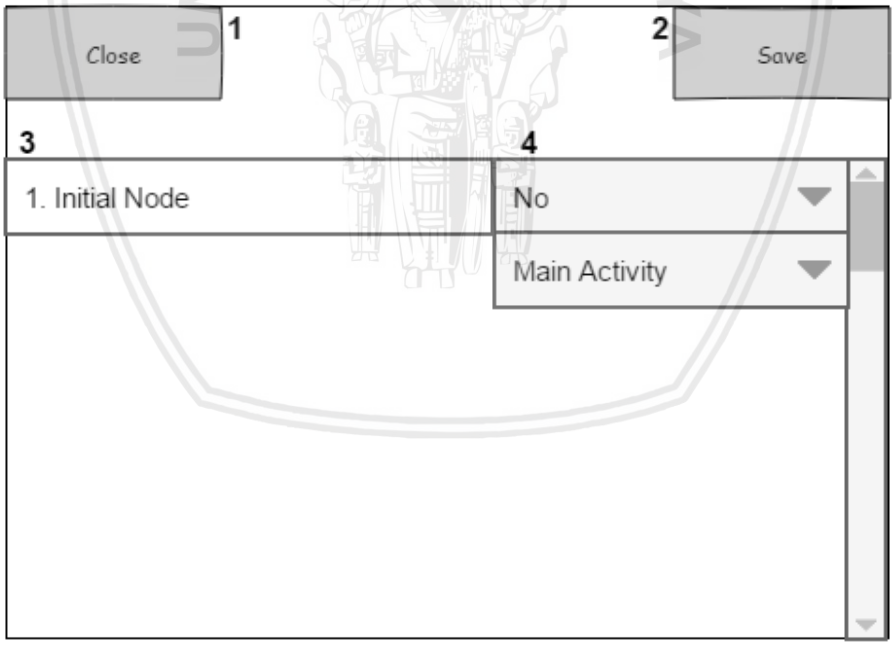
No.	Nama Komponen	Tipe	Deskripsi
1	Tombol Reset	Button	Tombol untuk menghapus hasil perhitungan dan kembali ke halaman utama
2	Teks Prioritizied Scenario	Label	Judul dari tabel hasil perhitungan

Tabel 5.6 Penjelasan perancangan antarmuka hitung prioritas (Lanjutan)

No.	Nama Komponen	Tipe	Deskripsi
3	Tabel Prioritizied Scenario	Table	Tabel untuk menampilkan hasil perhitungan dan menampilkan prioritas utama
4	Tombol Detail	Button	Untuk melihat lebih detail hasil perhitungan

5.1.3.4 Perancangan Antarmuka Menentukan Main Activity

Pada Gambar 5.3 merupakan perancangan antarmuka untuk menentukan main activity dan untuk penjelasan komponen dijelaskan pada Tabel 5.7. Pada Gambar 5.3 pengguna akan dihadapkan dengan halaman antarmuka untuk menentukan main activity pada setiap node sebelum sistem melakukan perhitungan secara otomatis. Pada perancangan antarmuka menentukan main activity ada 2 button, 1 panel, dan 1 combo box.



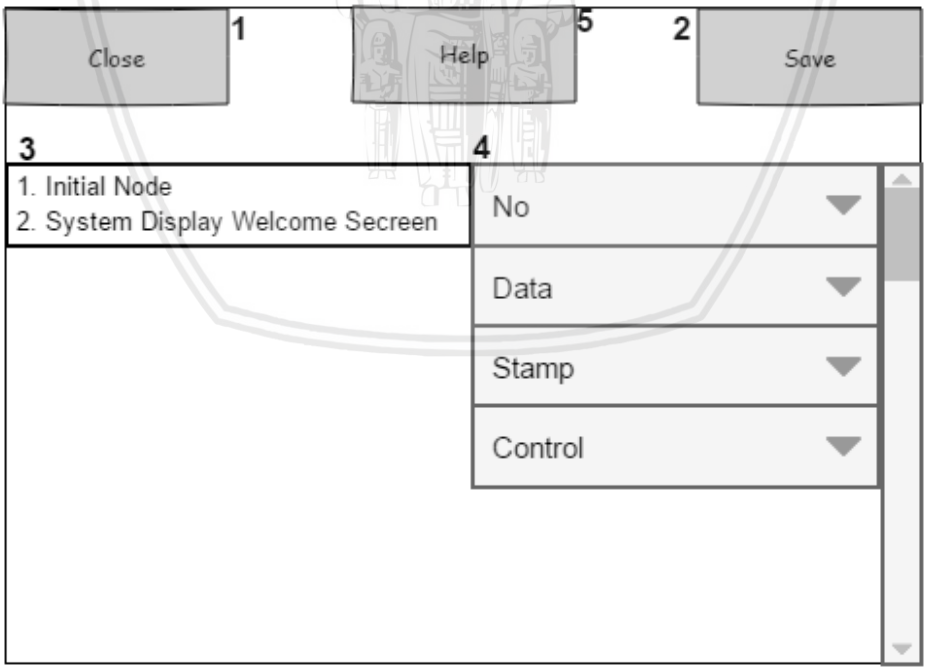
Gambar 5.8 Perancangan Antarmuka Menentukan Main Activity

Tabel 5.7 Penjelasan perancangan antarmuka menentukan main activity

No.	Nama Komponen	Tipe	Deskripsi
1	Tombol Close	Button	Tombol untuk kembali ke halaman Hitung Prioritas
2	Tombol Save	Button	Tombol untuk menyimpan hasil setelah menentukan main activity
3	Nama Node	Pane	Untuk menampilkan nama dari setiap node
4	Pilihan main activity	Combo Box	Untuk menampilkan pilihan antara main activity atau no

5.1.3.5 Perancangan Antarmuka Menentukan Tipe Kopling

Pada Gambar 5.3 merupakan perancangan antarmuka untuk menentukan tipe kopling dan untuk penjelasan komponen dijelaskan pada Tabel 5.8. Pada Gambar 5.3 pengguna akan dihadapkan dengan halaman antarmuka untuk menetukan tipe kopling pada setiap node yang satu dengan node yang lain yang saling berhubungan sebelum sistem melakukan perhitungan secara otomatis. Pada perancangan antarmuka menentukan main activity ada 2 button, 1 panel, dan 1 combo box.



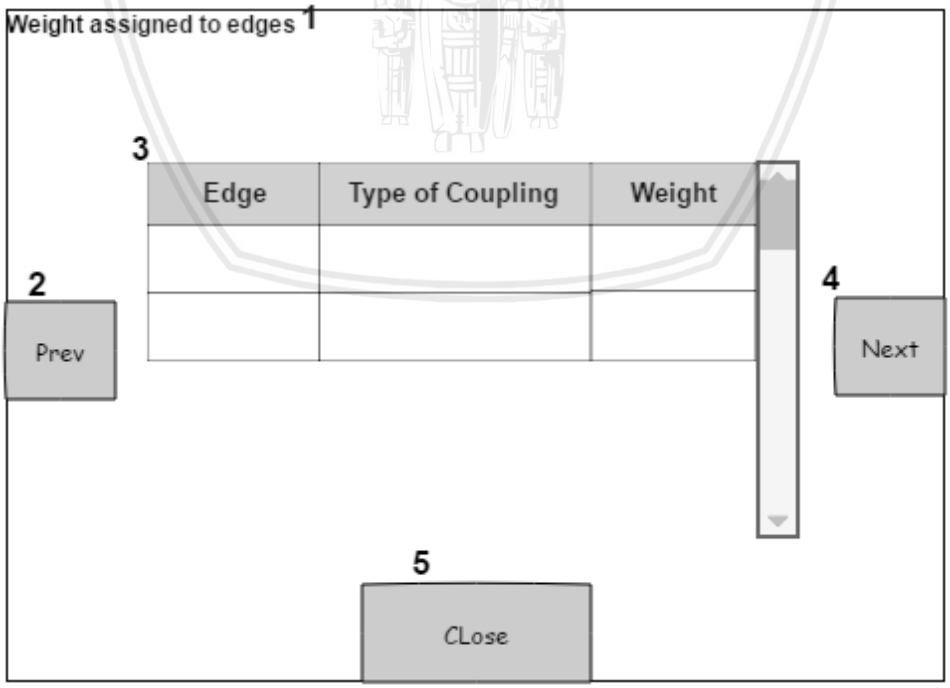
Gambar 5.9 Perancangan Antarmuka Menentukan Tipe Kopling

Tabel 5.8 Penjelasan perancangan antarmuka menentukan tipe kopling

No.	Nama Komponen	Tipe	Deskripsi
1	Tombol Close	Button	Tombol untuk kembali ke halaman Hitung Prioritas
2	Tombol Save	Button	Tombol untuk menyimpan hasil setelah menentukan tipe kopling
3	Nama Node	Pane	Untuk menampilkan nama dari setiap node satu ke node yang lain yang saling berhubungan
4	Pilihan main activity	Combo Box	Untuk menampilkan pilihan antara no, data, stamp, ataupun control
5	Tombol Help	Button	Untuk menampilkan penjelasan tentang tipe kopling yaitu data, stamp, dan control

5.1.3.6 Perancangan Antarmuka Lihat Detail

Pada Gambar 5.4 merupakan perancangan antarmuka untuk lihat detail dan untuk penjelasan komponen dijelaskan pada Tabel 5.9. Pada Gambar 5.4 pengguna akan dihadapkan dengan halaman antarmuka untuk melihat detail dari perhitungan dari setiap faktor yang berdasarkan pada *weighting factors*.



Gambar 5.10 Perancangan Antarmuka Lihat Detail

Tabel 5.9 Penjelasan perancangan antarmuka lihat detail

No.	Nama Komponen	Tipe	Deskripsi
1	Judul Tabel	Label	Teks untuk menampilkan judul pada perhitungan setiap faktor pembobotan
2	Tombol Prev	Button	Tombol untuk kembali pada halaman sebelumnya
3	Tabel	Table	Tabel yang menampilkan hasil perhitungan pada setiap faktor pembobotan
4	Tombol Next	Button	Tombol untuk kehalaman selanjutnya
5	Tombol Close	Button	Tombol untuk kembali pada halaman Hitung Prioritas

5.2 Implementasi Sistem

Implementasi sistem merupakan tahapan yang dikerjakan setelah melakukan perancangan pada sistem. Tahap implementasi dilakukan berdasarkan tahapan-tahapan sebelumnya yaitu tahap analisis dan perancangan. Pada implementasi sistem akan membahas spesifikasi sistem, mengimplementasikan perancangan komponen menjadi implementasi kode program, serta mengimplementasikan antarmuka dari rancangan antarmuka.

5.2.1 Spesifikasi Sistem

Pada bagian spesifikasi sistem ini akan menjelaskan tentang spesifikasi perangkat keras dan spesifikasi perangkat lunak yang digunakan untuk mendukung pengembangan kakas bantu penentuan prioritas *test scenario*. Pada Tabel 5.10 merupakan penjelasan spesifikasi perangkat keras yang meliputi penjelasan processor, secondary memory, dan memory (RAM). Sedangkan pada Tabel 5.11 merupakan penjelasan spesifikasi untuk perangkat lunak yang meliputi untuk spesifikasi perangkat lunak dipaparkan mengenai sistem operasi, editor dokumentasi, editor perancangan, bahasa pemrograman, dan lain sebagainya.

Tabel 5.10 Spesifikasi Perangkat Keras

Component	*Spesification*
Processor	Intel® Core ™ i7-4720HQ @ 2.60GHz
Memory (RAM)	4.00 GB
Hardisk	1024 GB

Tabel 5.11 Spesifikasi Perangkat Lunak

Component	*Spesification*
Operating System	Windows 10
Editor Dokumentasi	Microsoft® Word 2013
Editor Perancangan	Enterprise Architect 13.0.13.10
Bahasa Pemrograman	C#
Text Editor	Visual Studio 2019
User Interface Builder	Unity3D

5.2.2 Implementasi Kode Program

Perancangan komponen pada perancangan sistem selanjutnya akan diimplementasikan pada kode program. Pada implementasi kode program algoritme akan dirubah menjadi kode program. Pada penelitian kakas bantu penentuan prioritas *test scenario* bahasa pemrograman C#. Penelitian ini akan memaparkan 3 kode program yaitu method XML2JObjectByOpen, method DecisionShowTable, dan method UpdateTotalWeightOfEachPath. Pada penelitian ini implementasi kode program akan dijelaskan pada sub bab 5.1.2.1, sub bab 5.1.2.2, dan sub bab 5.1.2.3.

5.2.2.1 Implementasi Kode Program Method "*XML2JObjectByOpen*"

Pada Algoritme 4 merupakan method XML2JObjectByOpen, method XML2JObjectByOpen menjelaskan bahwa sistem melakukan filter terhadap berkas yang akan dimasukkan yaitu hanya dengan ekstensi xml. Method XML2JObjectByOpen melakukan seleksi apabila tempPath bernilai null atau kosong maka berkas yang akan dimasukkan tidak terdapat berkas dengan ekstensi xml, jika terdapat berkas dengan xml maka akan menampilkan berkas terdapat berkas xml.

Nama class: EasyHandler

Nama Operasi: XML2JObjectByOpen

```
Algoritme4 : Pseudocode method XML2JObjectByOpen
1   if (tempPath == null || tempPath == "") {
2      var extensions = new[] { new ExtensionFilter("Select XML
    Files", "xml") };
3      tempPath = @StandaloneFileBrowser.OpenFilePanel("Open File",
    "", extensions, true)[0];
4   }
5   return CreateFromXML(tempPath);
```

5.2.2.2 Implementasi Kode Program Method "*DecisionShowTable*"

Pada Algoritme 5 merupakan method DecisionShowTable, method DecisionShowTable digunakan untuk memanggil hasil perhitungan dan

menampilkan hasil perhitungan. Pada algoritme method DecisionShowTable melakukan seleksi jika case mempunyai nilai 2 maka akan memanggil method DataTable2 dan menampilkan hasil perhitungan dari method DataTable2 dalam bentuk tabel, jika case mempunyai nilai 3 maka akan memanggil method DataTable3 dan menampilkan hasil perhitungan dari method DataTable3 dalam bentuk tabel sampai pada case mempunyai nilai 11 maka akan memanggil method DataTable11 dan menampilkan hasil perhitungan dari method DataTable11 dalam bentuk tabel, jika case bernilai selain angka 2 sampai 11 maka sistem berakhir.

Nama class: SimpleInitTable

Nama Operasi: DecisionShowTable

Tabel 5.12 Kode Program Method DecisionShowTable

Algoritme5 : Pseudocode method DecisionShowTable

```
1   switch (index)
2           {
3               case 2:
4                   PassingDataToTable(
5                       main.core.DataTable2(),
6                       "Table 1. Information complexity table",
7                       new string[4]
8                       {
9                           "Node",
10                          "IN",
11                          "OUT",
12                          "IN x OUT"
13                      });
14                  break;
15              case 3:
16                  PassingDataToTable(
17                      main.core.DataTable3(),
18                      "Table 2. Weight of paths based on
    information complexity",
19                      new string[3]
20                      {
21                          "Independent paths",
22                          "Path description",
23                          "Weight of each path"
24                      });
25                  break;
26              case 4:
27                  PassingDataToTable(
28                      main.core.DataTable4(),
29                      "Table 3. Complexity value of nodes",
30                      new string[3]
31                      {
32                          "Node",
33                          "Amount of node contribution",
34                          "Complexity value of each node"
35                      });
36                  break;
37              case 5:
38                  PassingDataToTable(
39                      main.core.DataTable5(),
40                      "Table 4. Path complexity based on node
    complexity value",
41                      new string[3]
42                      {
43                          "Independent paths",
44                          "Path description",
```

```
45                              "Complexity value of each path"
46                          });
47                      break;
48                  case 6:
49                      PassingDataToTable(
50                          main.core.DataTable6(),
51                          "Table 5. Node coverage values",
52                          new string[3]
53                          {
54                              "Independent paths",
55                              "Path description",
56                              "Node coverage of each path"
57                          });
58                      break;
59                  case 7:
60                      PassingDataToTable(
61                          main.core.DataTable7(),
62                          "Table 6. Decision node coverage values",
63                          new string[3]
64                          {
65                              "Independent paths",
66                              "Path description",
67                              "Decision node coverage of each path"
68                          });
69                      break;
70                  case 8:
71                      PassingDataToTable(
72                          main.core.DataTable8(),
73                          "Table 7. Weight assigned to edges",
74                          new string[3]
75                          {
76                              "Edge",
77                              "Type of coupling",
78                              "Weight"
79                          });
80                      break;
81                  case 9:
82                      PassingDataToTable(
83                          main.core.DataTable9(),
84                          "Table 8. Weight of paths based on coupling
    factor",
85                          new string[3]
86                          {
87                              "Independent paths",
88                              "Path description",
89                              "Weight of each path"
90                          });
91                      break;
92                  case 10:
93                      PassingDataToTable(
94                          main.core.DataTable10(),
95                          "Table 9. Total weight of each path",
96                          new string[3]
97                          {
98                              "Independent paths",
99                              "Path description",
100                             "Total weight of each path"
101                         });
102                     break;
103                 case 11:
104                     PassingDataToTable(
105                         main.core.DataTable11(),
106                         "Table 10. Prioritized scenario",
107                         new string[4]
108                         {
```

```
109                             "Scenario",
110                             "Paths",
111                             "Path description",
112                             "Total weight of each path"
113                         });
114                     break;
115                 default:
116                     break;
117             }
```

5.2.2.3 Implementasi Kode Program Method "*UpdateTotalWeightOfEachPath*"

Pada Algoritme 6 merupakan method UpdateTotalWeightOfEachPath, method UpdateTotalWeightOfEachPath digunakan untuk menjumlahkan perhitungan dengan *weighting factors*. Pada algoritme method UpdateTotalWeightOfEachPath melakukan perulangan apabila nilai variabel i kurang dari independentPath maka menjumlahkan semua perhitungan dan disimpan pada variabel totalWeight. Sedangkan apabila variabel i lebih besar dari independentPath maka proses perulangan berakhir.

Nama class: TestScenarioPriotization

Nama Operasi: UpdateTotalWeightOfEachPath

Tabel 5.13 Kode Program Method UpdateTotalWeightOfEachPath

```
Algoritme6 : Pseudocode method UpdateTotalWeightOfEachPath
1   dataTotalWeightOfEachPath = new List<TotalWeightOfEachPath>();
2   for (int i = 0; i < _independentPaths.Count; i++)
3   {
4      dataTotalWeightOfEachPath.Add(new TotalWeightOfEachPath()
5      {
6         path = _independentPaths[i],
7         totalWeight =
8             dataWeightOfPaths[i].weight
9             + dataComplexityValueNodes[i].complexityValueNode
10            + dataNodeCoverageValues[i].nodeCoverage
11            + dataDecisionNodeCoverageValue[i].decisionoNode
12            + dataWeightOfPathsBasedOnCoupling[i].weightOfEachPath
13     });
14  }
```

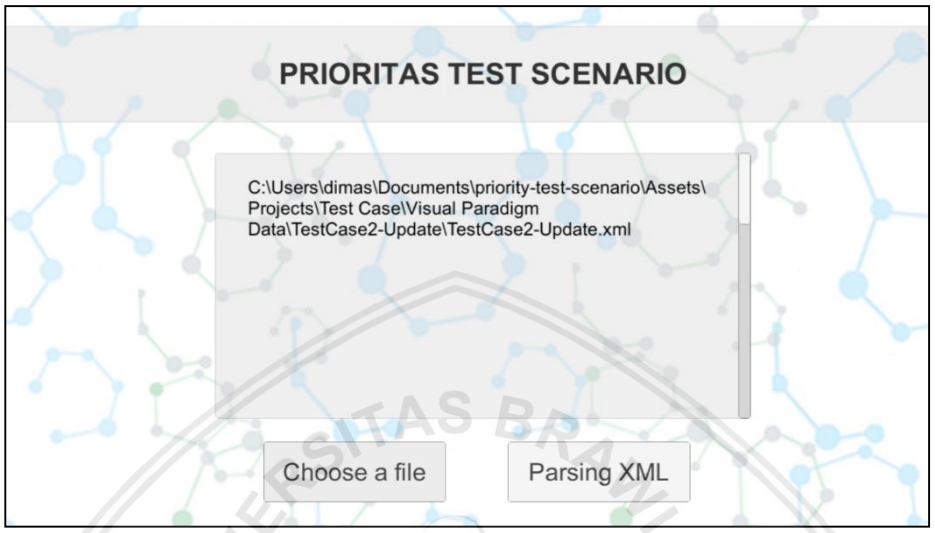
5.2.3 Implementasi Antarmuka

Implementasi antarmuka diperoleh dari hasil rancangan antarmuka yang dijelaskan sebelum pada perancangan sistem. Implementasi antarmuka diimplementasikan dengan Unity3D dan akan dijelaskan enam antarmuka yaitu antarmuka input berkas *activity diagram,* parsing xml, hitung prioritas, menetukan *main activity,* menentukan tipe kopling, dan lihat detail yang dijelaskan pada sub bab 5.2.3.1 sampai 5.2.3.6.

5.2.3.1 Implementasi Antarmuka Input Berkas *Activity Diagram*

Pada Gambar 5.11 merupakan gambar implementasi antarmuka input berkas *activity diagram*. Pada gambar implementasi antarmuka input berkas *activity diagram* tersebut terdapat 2 tombol yaitu choose a file untuk menampilkan

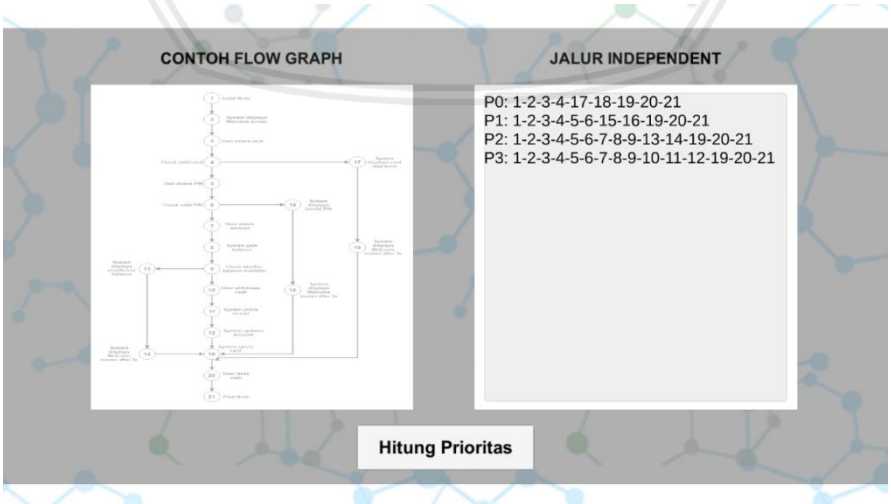
window pengelola file menu browse dan pengguna memilih berkas dengan ekstensi ".xml" yang akan di parsing. Setelah memilih berkas tersebut akan menampilkan nama berkas yang dipilih. Sedangkan untuk tombol parsing XML adalah tombol yang digunakan untuk merubah berkas *activity diagram* dengan ekstensi ".xml" menjadi *control flow graph* dan jalur independen.



Gambar 5.11 Implementasi Antarmuka Input Berkas Activity Diagram

5.2.3.2 Implementasi Antarmuka Parsing XML

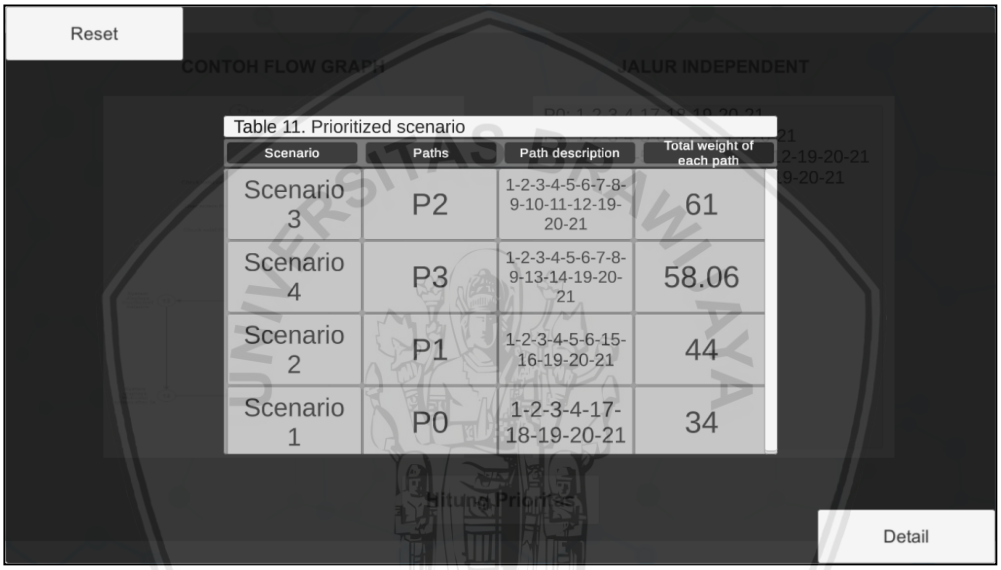
Pada Gambar 5.12 merupakan gambar implementasi antarmuka parsing XML. Pada gambar implementasi antarmuka parsing XML yaitu merubah berkas activity diagram yang sebelumnya sudah dipilih menjadi control flow graph dan jalur independen dan kemudian ada tombol hitung prioritas. Tombol prioritas merupakan tombol yang digunakan untuk menghitung berkas activity diagram tersebut sesuai dengan setiap jalur independennya menggunakan *weighting actors.*



Gambar 5.12 Implementasi Antarmuka Parsing XML

5.2.3.3 Implementasi Antarmuka Hitung Prioritas

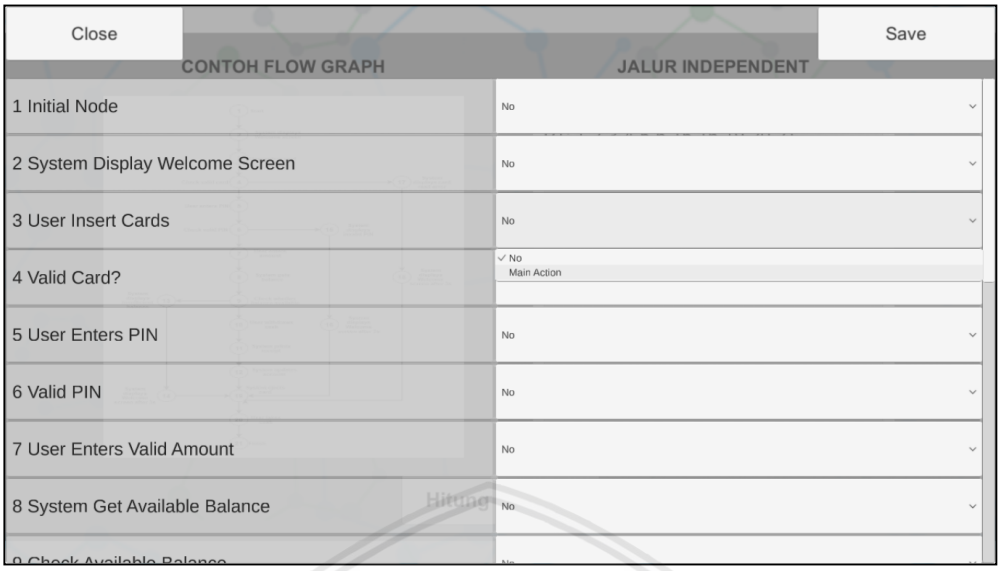
Pada gambar implementasi antarmuka hitung prioritas yaitu implementasi antarmuka yang menampilkan hasil perhitungan pada berkas activity diagram berekstensi ".xml" dengan menggunakan *weighting factors* pada setiap jalur independen kemudian memprioritaskan jalur independen dengan total hasil perhitungan tertinggi. Pada Gambar 5.13 terdapat 2 tombol reset dan detail. Tombol detail digunakan untuk melihat detail perhitungan dari setiap faktor dari beberapa faktor pembobotan (*weighting factors*). Sedangkan tombol reset digunakan untuk menghapus hasil perhitungan dan kembali pada halaman menu utama.



Gambar 5.13 Implementasi Antarmuka Hitung Prioritas

5.2.3.4 Implementasi Antarmuka Menentukan Main Activity

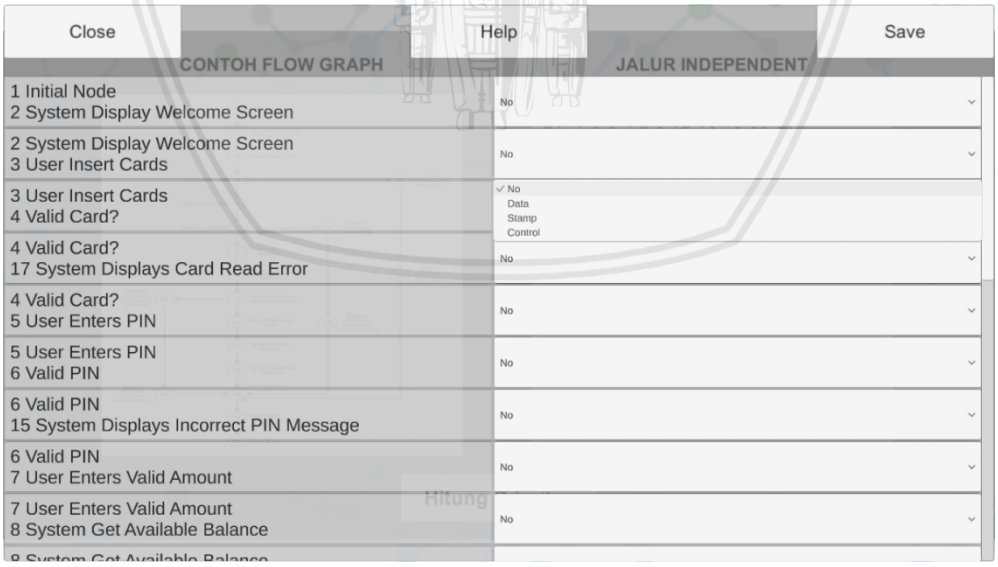
Pada Gambar 5.14 yaitu terdapat sebuah halaman untuk menentukan main activity atau no (bukan main activity) pada setiap node sebelum sistem menghitung *calculations of path complexity* pada *weighting factors*. Implementasi antarmuka menentukan main activity ini terdapat 2 tombol *button* yaitu *button* save dan *button* close. *Button* save digunakan untuk menyimpan hasil inputan pengguna, sedangkan *button* close digunakan untuk menutup halaman dan kembali ke halaman hitung prioritas.



Gambar 5.14 Implementasi Antarmuka Menentukan Main Activity

5.2.3.5 Implementasi Antarmuka Menentukan Tipe Kopling

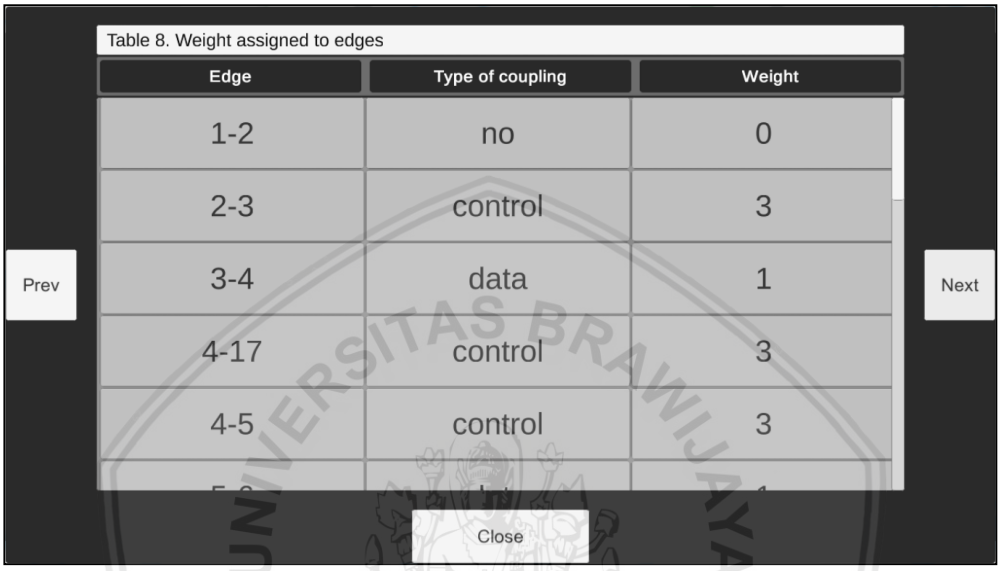
Pada Gambar 5.15 yaitu terdapat sebuah halaman untuk menentukan tipe kopling yaitu no, data, stamp, dan control dari node yang satu ke node yang lain yang saling berhubungan sebelum sistem menghitung *calculation of weight of each path* pada *weighting factors*. Implementasi antarmuka menentukan tipe kopling ini terdapat 2 tombol *button* yaitu *button* save dan *button* close. *Button s*ave digunakan untuk menyimpan hasil inputan pengguna, sedangkan *button* close digunakan untuk menutup halaman dan kembali ke halaman hitung prioritas.



Gambar 5.15 Implementasi Antarmuka Menentukan Tipe Kopling

5.2.3.6 Implementasi Antarmuka Lihat Detail

Pada Gambar 5.16 merupakan halaman detail dari perhitungan pada setiap jalur menggunakan *weighting factors.* Implementasi antarmuka lihat detail ini menampilkan setiap perhitungan dalam bentuk tabel kemudian terdapat tombol next untuk melihat tabel perhitungan selanjutnya, tombol prev untuk melihat tabel perhitungan sebelumnya, dan terdapat tombol close untuk kembali menampilkan halaman hitung prioritas.



Gambar 5.16 Implementasi Antarmuka Lihat Detail

BAB 6 PENGUJIAN SISTEM

6.1 Pengujian Unit

Pengujian unit adalah pengujian dengan metode *whitebox testing* yang menggunakan teknik basis path testing. Pengujian unit akan menguji dari setiap komponen pada perangkat lunak dan pada penelitian ini akan menguji tiga method yaitu XML2JObjectByOpen, DecisionShowTable, dan UpdateTotalWeightOfEachPath.

6.1.1 Pengujian Unit Method *"*XML2JObjectByOpen*"*

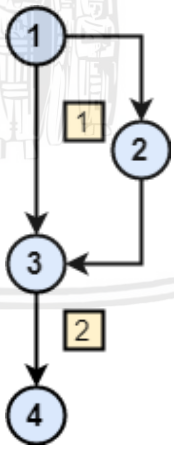
1. *Pseudocodes*

Algoritme7 : Pseudocode method XML2JObjectByOpen		
1	If tempPath sama dengan null atau tempPath sama dengan kosong	**1**
2	Instansiasi variabel extensions	
3	tempPath <- StandaloneFileBrowser dan memanggil OpenFilePanel	**2**
4	Endif **3**	
5	Mengembalikan nilai CreateFromXML parameter tempPath	**4**

2. Basis Path Testing

   a. Flow Graph

   Setelah melakukan penjabaran pada algoritme maka selanjutnya yaitu membuat flow graph pada algoritme method XML2JObjectByOpen seperti pada Gambar 6.1.



Gambar 6.1 Flow graph method XML2JObjectByOpen

   b. Cyclomatic Complexity

   Setelah membuat flow graph maka akan didapatkan jalur independen melalui perhitungan cyclomatic complexity. Hasil perhitungan cyclomatic complexity pada method XML2JObjectByOpen adalah:

   - V(G) = jumlah region = 2
   - V(G) = Edge (E) – Node (N) + 2 = 4 edge – 4 node + 2 = 2

- V(G) = Predicate Node (P) + 1 = 1 + 1 = 2

c. Independent Path

- Jalur 1: 1-3-4
- Jalur 2: 1-2-3-4

Jalur independen yang diperoleh selanjutnya akan dilakukan untuk membuat kasus uji (*test* case). Pada method XML2JObjectByOpen kasus uji dan hasil pengujian dijelaskan pada Tabel 6.1

Tabel 6.1 Hasil pengujian method XML2JObjectByOpen

Jalur	Prosedur pengujian	Expected Result	Actual Result	Status
1	Memanggil *method* XML2JObjectByOpen dengan nilai tempPath = " " atau null	*Method* XML2JObjectByOpen akan menampilkan "*Tidak ada Berkas dengan Ekstensi xml*"	*Method* XML2JObjectByOpen berhasil menampilkan "*Tidak ada Berkas dengan Ekstensi xml*"	Valid
2	Memanggil *method* XML2JObjectByOpen dengan nilai tempPath = "xml"	*Method* XML2JObjectByOpen akan menampilkan "*Terdapat Berkas xml*"	*Method* XML2JObjectByOpen berhasil menampilkan "*Terdapat Berkas xml*"	Valid

6.1.2 Pengujian Unit Method *"*DecisionShowTable*"*

1. Pseudocode

```
Algoritme8 : Pseudocode method DecisionShowTable

Switch denan parameter index
   Case berlnilai 2 2
      Memanggil method PassingDataToTable
         Memanggil method DataTable2
         Menampilkan "Table 1. Information complexity table"
         Inisialisasi array dengan nilai 4
            Membuat kolom dan Menampilkan "Node"
            Membuat kolom dan Menampilkan "IN"
            Membuat kolom dan Menampilkan "OUT"
            Membuat kolom dan Menampilkan "IN x OUT"
      Endcase 4
   Case belnilai 3  5
      Memanggil method PassingDataToTable
         Memanggil method DataTable3
         Menampilkan "Table 2. Weight of paths based on
information complexity"
         Inisialisasi array dengan nilai 3
            Membuat kolom dan Menampilkan "Independent paths"
            Membuat kolom dan Menampilkan "Path description"
            Membuat kolom dan Menampilkan "Weight of each path"
      Endcase 7
   Case bernilai 4 8
      Memanggil method PassingDataToTable
         Memanggil method DataTable4
         Menampilkan "Table 3. Complexity value of nodes"
```

3

6

```
25          Inisialisasi array dengan nilai 3
26             Membuat kolom dan Menampilkan "Node"
27             Membuat kolom dan Menampilkan "Amount of node
    contribution"
28             Membuat kolom dan Menampilkan "Complexity value of
    each node"
29       Endcase 10
30    Case bernilai 5  11
31       Memanggil method PassingDataToTable
32          Memanggil method DataTable5
33          Menampilkan "Table 4. Path complexity based on node
    complexity value "
34          Inisialisasi array dengan nilai 3
35             Membuat kolom dan Menampilkan "Independent paths"
36             Membuat kolom dan Menampilkan "Path description"
37             Membuat kolom dan Menampilkan "Complexity value of
    each path"
38       Endcase 13
39    Case bernilai 6  14
40       Memanggil method PassingDataToTable
41          Memanggil method DataTable6
42          Menampilkan "Table 5. Node coverage value"
43          Insialisasi array dengan nilai 3
44             Membuat kolom dan Menampilkan "Independent paths"
45             Membuat kolom dan Menampilkan "Path description"
46             Membuat kolom dan Menampilkan "Node coverage of each
    path"
47       Endcase 16
48    Case bernilai 7  17
49       Memanggil method PassingDataToTable
50          Memanggil method DataTable7
51          Menampilkan "Table 6. Decision node coverage values"
52          Inisialisasi array dengan nilai 3
53             Membuat kolom dan Menampilkan "Independent paths"
54             Membuat kolom dan Menampilkan "Path description"
55             Membuat kolom dan Menampilkan "Decision node
    coverage of each path"
56       Endcase 19
57    Case bernilai 8  20
58       Memanggil method PassingDataToTable
59          Memanggil method DataTable8
60          Menampilkan "Table 7. Weight assigned to edges"
61          Inisialisasi array dengan nilai 3
62             Membuat kolom dan Menampilkan "Edge"
63             Membuat kolom dan Menampilkan "Type of coupling"
64             Membuat kolom dan Menampilkan "Weight"
65       Endcase 22
66    Case bernilai 9  23
67       Memanggil method PassingDataToTable
68          Memanggil method DataTable9
69          Menampilkan "Table 8. Weight of paths based on coupling
    factor"
70          Inisialisasi array dengan nilai 3
71             Membuat kolom dan Menampilkan "Independent paths"
72             Membuat kolom dan Menampilkan "Path description"
73             Membuat kolom dan Menampilkan "Weight of each path"
74       Endcase 25
75    Case bernilai 10  26
76       Memanggil method PassingDataToTable
77          Memanggil method DataTable10
78          Menampilkan "Table 9. Total weight of each path"
79          Inisialisasi array dengan nilai 3
80             Membuat kolom dan Menampilkan "Independent paths"
81             Membuat kolom dan Menampilkan "Path description"
82
```

9 12 15 18 21 24 27

```
                Membuat kolom dan Menampilkan "Total weight of each
83  path"
84          Endcase 28
85      Case bernilai 11  29
86          Memanggil method PassingDataToTable
87              Memanggil method DataTable11
88              Menampilkan "Table 10. Prioritized scenario"
89              Inisialisasi array dengan nilai 4
90                  Membuat kolom dan Menampilkan "Scenario"
91                  Membuat kolom dan Menampilkan "Path"
92                  Membuat kolom dan Menampilkan "Path description"
                    Membuat kolom dan Menampilkan "Total weight of each
93  path"
94          Endcase 31
    End Switch 32
```

30

2. Basis Path Testing

   a. Flow Graph

   Setelah melakukan penjabaran pada algoritme maka selanjutnya yaitu membuat flow graph pada algoritme method DecisionShowTable seperti pada Gambar 6.2.

   b. Cyclomatic Complexity

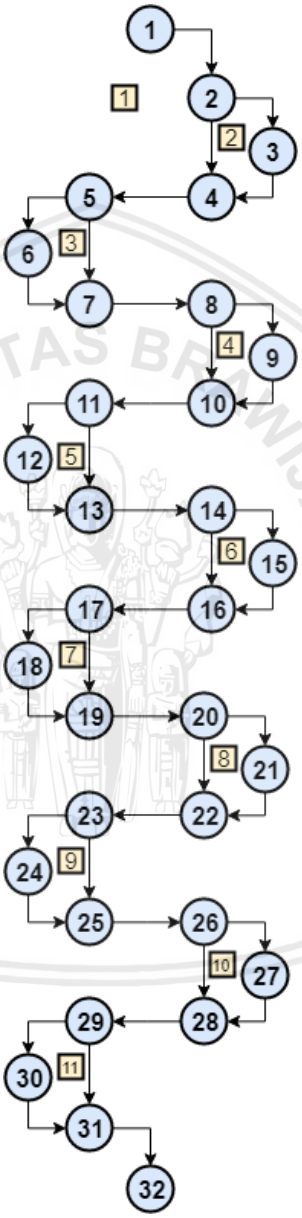
   Setelah membuat flow graph maka akan didapatkan jalur independen melalui perhitungan cyclomatic complexity. Hasil perhitungan cyclomatic complexity pada method DecisionShowTable adalah:

   - V(G) = jumlah region = 11
   - V(G) = Edge (E) – Node (N) + 2 = 41edge – 31node + 2 = 11
   - V(G) = Predicate Node (P) + 1 = 10 + 1 = 11

   c. Independent Path

   - Jalur 1: 1-2-4-5-7-8-10-11-13-14-16-17-19-20-22-23-25-26-28-29-31-32
   - Jalur 2: 1-2-3-4-5-7-8-10-11-13-14-16-17-19-20-22-23-25-26-28-29-31-32
   - Jalur 3: 1-2-3-4-5-6-7-8-10-11-13-14-16-17-19-20-22-23-25-26-28-29-31-32
   - Jalur 4: 1-2-3-4-5-6-7-8-9-10-11-13-14-16-17-19-20-22-23-25-26-28-29-31-32
   - Jalur 5: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-16-17-19-20-22-23-25-26-28-29-31-32
   - Jalur 6: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-19-20-22-23-25-26-28-29-31-32
   - Jalur 7: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-22-23-25-26-28-29-31-32
   - Jalur 8: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-25-26-28-29-31-32

- Jalur 9: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-28-29-31-32
- Jalur 10: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-31-32
- Jalur 11: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32



Gambar 6.2 Flow Grpah method DecisionShowTable

Jalur Jalur independen yang diperoleh selanjutnya akan dilakukan untuk membuat kasus uji (*test* case). Pada method DecisionShowTable kasus uji dan hasil pengujian dijelaskan pada Tabel 6.2 sampai Tabel 6.3

Tabel 6.2 Hasil pengujian method DecisionShowTable

Jalur	Prosedur pengujian	Expected Result	Actual Result	Status
1	Memanggil *method* DecisionShowTable dengan nilai case = 2	*Method* DecisionShowTable akan menampilkan "Table 1. Information complexity table"	*Method* DecisionShowTable berhasil menampilkan "Table 1. Information complexity table"	Valid
2	Memanggil *method* DecisionShowTable dengan nilai case = 3	*Method* DecisionShowTable akan menampilkan "Table 2. Weight of paths based on information complexity"	*Method* DecisionShowTable berhasil menampilkan "Table 2. Weight of paths based on information complexity"	Valid
3	Memanggil *method* DecisionShowTable dengan nilai case = 4	*Method* DecisionShowTable akan menampilkan "Table 3. Complexity value of nodes"	*Method* DecisionShowTable berhasil menampilkan "Table 3. Complexity value of nodes"	Valid
4	Memanggil *method* DecisionShowTable dengan nilai case = 5	*Method* DecisionShowTable akan menampilkan "Table 4. Path complexity based on node complexity value"	*Method* DecisionShowTable berhasil menampilkan "Table 4. Path complexity based on node complexity value"	Valid
5	Memanggil *method* DecisionShowTable dengan nilai case = 6	*Method* DecisionShowTable akan menampilkan "Table 5. Node coverage values"	*Method* DecisionShowTable berhasil menampilkan "Table 5. Node coverage values"	Valid
6	Memanggil *method* DecisionShowTable dengan nilai case = 7	*Method* DecisionShowTable akan menampilkan "Table 6. Decision node coverage values"	*Method* DecisionShowTable berhasil menampilkan "Table 6. Decision node coverage values"	Valid

Tabel 6.3 Hasil pengujian method DecisionShowTable (Lanjutan)

Jalur	Prosedur uji	Hasil pengujian yang diharapkan	Hasil pengujian	Status
7	Memanggil *method* DecisionShowTable dengan nilai case = 8	*Method* DecisionShowTable akan menampilkan "Table 7. Weight assigned to edges"	*Method* DecisionShowTable berhasil menampilkan "Table 7. Weight assigned to edges"	Valid
8	Memanggil *method* DecisionShowTable dengan nilai case = 9	*Method* DecisionShowTable akan menampilkan "Table 8. Weight of paths 9 based on coupling factor"	*Method* DecisionShowTable berhasil menampilkan "Table 8. Weight of paths based on coupling factor"	Valid
9	Memanggil *method* DecisionShowTable dengan nilai case = 10	*Method* DecisionShowTable akan menampilkan "Table 9. Total weight of each path"	*Method* DecisionShowTable berhasil menampilkan "Table 9. Total weight of each path"	Valid
10	Memanggil *method* DecisionShowTable dengan nilai case = 11	*Method* DecisionShowTable akan menampilkan "Table 10. Prioritized scenario"	*Method* DecisionShowTable berhasil menampilkan "Table 10. Prioritized scenario"	Valid
11	Memanggil *method* DecisionShowTable dengan nilai case = 12	*Method* DecisionShowTable akan menampilkan "Tidak Terdapat Tabel"	*Method* DecisionShowTable berhasil menampilkan "Tidak Terdapat Tabel"	Valid

6.1.3 Pengujian Unit Method *"*UpdateTotalWeightOfEachPath*"*

1. *Pseudocodes*

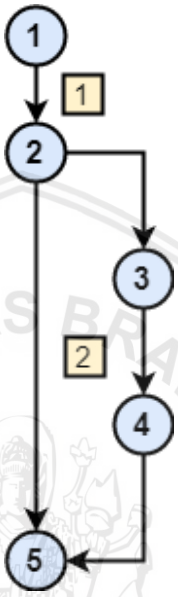
```
Algoritme9 : Pseudocode method UpdateTotalWeightOfEachPath
1   Instansiasi dataTotalWeightOfEachPath 1
2   For variabel i = 0, i kurang dari _independentPaths 2
3      DataTotalWeightOfEachPath dengan parameter 3
    TotalWeightOfEachPath
4         Inisialisasi path = _independentPaths ke-i
5         totalWeight = dataWeightOfPaths ke-i memanggil weight
            +dataComplexityValueNodeske-i memanggil
    complexityValueNode
            +dataNodeCoverageValue ke-i memanggil nodeCoverage
            +dataDecisionNodeCoverageValue ke-i decisionoNode
            +dataWeightOfPathsBasedOnCoupling ke-i memanggil
    weightOfEachPath                                          4
```

6	Endfor **5**

2. Basis Path Testing

   a. Flow Graph

   Setelah melakukan penjabaran pada algoritme maka selanjutnya yaitu membuat flow graph pada algoritme method UpdateTotalWeightOfEachPath seperti pada Gambar 6.3.



Gambar 6.3 Flow graph method UpdateTotalWeightOfEachPath

   b. Cyclomatic Complexity

   Dari Setelah membuat flow graph maka akan didapatkan jalur independen melalui perhitungan cyclomatic complexity. Hasil perhitungan cyclomatic complexity pada method UpdateTotalWeightOfEachPath adalah:

   - V(G) = jumlah region = 2
   - V(G) = Edge (E) – Node (N) + 2 = 5edge – 5node + 2 = 2
   - V(G) = Predicate Node (P) + 1 = 1+1= 2

   c. Independent Path

   - Jalur 1: 1-2-5
   - Jalur 2: 1-2-3-4-5

   Jalur independen yang diperoleh selanjutnya akan dilakukan untuk membuat kasus uji (*test* case). Pada method UpdateTotalWeightOfEachPath kasus uji dan hasil pengujian dijelaskan pada Tabel 6.4.

Tabel 6.4 Hasil pengujian method UpdateTotalWeightOfEachPath

Jalur	Prosedur pengujian	Expected Result	Actual Result	Status
1	Memanggil *method* UpdateTotalWeightOfEachPath dengan nilai variabel i = 2, _independentPaths = 4	Berhasil menampilkan "P2 dengan Total = 58"	Berhasil menampilkan "P2 dengan Total = 58"	Valid
2	Memanggil *method* UpdateTotalWeightOfEachPath*()* dengan nilai variabel i = 3, _independentPaths = 4, maka dataTotalWeightOfEachPath =	Berhasil menampilkan "P3 dengan Total = 24"	Berhasil menampilkan "P3 dengan Total = 24"	Valid

6.2 Pengujian Validasi

Pengujian validasi menguji seluruh kebutuhan fungsional dengan *blackbox testing*, pengujian ini dilakukan untuk memastikan apakah sudah selaras dengan *use case scenario* yang didefinisikan sebelumnya. Pengujian validasi berhasil apabila sistem yang telah diimplementasikan sesuai dengan dengan semua kebutuhan dan alur sesuai dengan skenario yang telah ditetapkan.

6.2.1 Pengujian Validasi Input Berkas Activity Diagram

Pengujian validasi input berkas *activity diagram* dengan kode TS-F-01 menguji jalur utama bahwa sistem dapat memasukkan berkas *activity diagram* dengan ekstensi xml kedalam sistem. Pengujian validasi jalur alternatif menguji bahwa sistem harus kembali pada halaman utama jika menekan tombol cancel dengan tidak memasukkan berkas ke dalam sistem dan menguji sistem tidak dapat menampilkan berkas atau tidak dapat menemukan berkas jika tidak terdapat berkas dengan ekstensi xml. Pengujian validasi jalur utama input berkas *activity diagram* dijabarkan dalam Tabel 6.5 dan Tabel 6.6, pengujian validasi jalur alternatif input berkas *activity diagram* dijabarkan dalam Tabel 6.7 dan Tabel 6.8.

Tabel 6.5 Pengujian validasi input berkas activity diagram

Kode kebutuhan	TS-F-01
Prosedur pengujian	1. Pengguna menekan tombol Choose a file 2. Sistem menampilkan *window* pengelola berkas 3. Pengguna memilih berkas *activity diagram* kemudian menekan tombol Open

Tabel 6.6 Pengujian validasi input berkas activity diagram (Lanjutan)

Expected Result	Berkas activity diagram berhasil dimasukkan dan menampilkan nama file serta alamat berkas
Actual Result	Berkas activity diagram berhasil dimasukkan dan menampilkan nama file serta alamat berkas
Status	Valid

Tabel 6.7 Pengujian validasi input berkas activity diagram jalur alternatif (1)

Kode kebutuhan	TS-F-01
Prosedur pengujian	1. Pengguna menekan tombol Choose a file 2. Sistem menampilkan *window* pengelola berkas 3. Pengguna menekan tombol cancel pada *window* pengelola berkas
Expected Result	Sistem dapat menampilkan halaman awal dan tidak menampilkan berkas activity diagram dengan ekstensi xml
Actual Result	Sistem dapat menampilkan halaman awal dan tidak menampilkan berkas activity diagram dengan ekstensi xml
Status	Valid

Tabel 6.8 Pengujian validasi input berkas activity diagram jalur alternatif (2)

Kode kebutuhan	TS-F-01
Prosedur pengujian	1. Pengguna menekan tombol Choose a file 2. Sistem menampilkan *window* pengelola berkas 3. Pengguna memilih direktori yang tidak terdapat bekas dengan ekstensi xml
Expected Result	Sistem tidak dapat menampilkan berkas atau tidak dapat menemukan berkas
Actual Result	Sistem tidak dapat menampilkan berkas atau tidak dapat menemukan berkas
Status	Valid

6.2.2 Pengujian Validasi Parsing XML

Pengujian validasi parsing XML dengan kode TS-F-02 menguji jalur utama bahwa sistem dapat memparsing berkas *activity diagram* dengan ekstensi xml menjadi *control flow graph* dan jalur independen. Pengujian validasi jalur alternatif menguji bahwa sistem tidak dapat melakukan parsing pada berkas dengan ekstensi xml yang bukan berkas *activity diagram*. Pengujian validasi jalur utama parsing XML dijabarkan dalam Tabel 6.9, pengujian validasi jalur alternatif parsing XML dijabarkan dalam Tabel 6.10.

Tabel 6.9 Pengujian validasi parsing xml

Kode kebutuhan	TS-F-02
Prosedur pengujian	1. Pengguna menekan tombol Parsing XML 2. Sistem menampilkan gambar control flow graph dan menampilkan jalur independen
Expected Result	Sistem menampilkan gambar control flow graph dan jalur independen
Actual Result	Sistem menampilkan gambar control flow graph dan jalur independen
Status	Valid

Tabel 6.10 Pengujian validasi parsing xml jalur alternatif

Kode kebutuhan	TS-F-02
Prosedur pengujian	1. Pengguna menekan tombol Parsing XML 2. Sistem tidak dapat melakukan parsing
Expected Result	Sistem tidak dapat melakukan parsing atau tombol parsing tidak dapat digunakan
Actual Result	Sistem tidak dapat melakukan parsing atau tombol parsing tidak dapat digunakan
Status	Valid

6.2.3 Pengujian Validasi Hitung Prioritas

Pengujian validasi hitung prioritas dengan kode TS-F-03 menguji jalur utama bahwa sistem dapat menghitung berkas activity diagram pada setiap jalur independen dengan *weighting factors* (faktor pembobotan) dan menampilkan urutan prioritas perhitungan dalam bentuk tabel dengan urutan nilai *total weight of each path* paling besar. Pengujian validasi jalur alternatif menguji bahwa sistem tidak dapat melakukan parsing pada berkas dengan ekstensi xml yang bukan berkas *activity diagram*. Pengujian validasi jalur utama hitung prioritas dijabarkan dalam Tabel 6.11 dan Tabel 6.12.

Tabel 6.11 Pengujian validasi hitung prioritas

Kode kebutuhan	TS-F-03
Prosedur pengujia	1. Pengguna menekan tombol Hitung Prioritas 2. Sistem menampilkan halaman hasil perhitungan dan menampilkan urutan prioritas berdasarkan nilai paling besar
Expected Result	Sistem menampilkan tabel hasil perhitungan dan urutan prioritas perhitungan dengan urutan nilai *total weight of each path* paling besar

Tabel 6.12 Pengujian validasi hitung prioritas (Lanjutan)

Actual Result	Sistem menampilkan tabel hasil perhitungan dan urutan prioritas perhitungan dengan urutan nilai *total weight of each path* paling besar
Status	Valid

6.2.4 Pengujian Validasi Lihat Detail

Pengujian validasi lihat detail dengan kode TS-F-04 menguji jalur utama bahwa sistem dapat menunjukkan hasil perhitungan setiap faktor dari faktor pembobotan (*weighting factors)* pada masing-masing jalur independen. Pengujian validasi jalur utama lihat detail dijabarkan dalam Tabel 6.13.

Tabel 6.13 Pengujian validasi lihat detail

Kode kebutuhan	TS-F-04
Prosedur pengujian	1. Pengguna menekan tombol Lihat Detail 2. Sistem menampilkan halaman hasil perhitungan pada setiap faktor dalam bentuk table
Expected Result	Sistem menampilkan tabel detail hasil perhitungan dari faktor pembobotan Weight of each test path, Complexity value of each path, Node coverage, Decision node coverage, dan *Weight of each path.*
Actual Result	Sistem menampilkan tabel detail hasil perhitungan dari faktor pembobotan Weight of each test path, Complexity value of each path, Node coverage, Decision node coverage, dan *Weight of each path.*
Status	Valid

6.2.5 Pengujian Validasi Reset

Pengujian validasi lihat detail dengan kode TS-F-05 menguji jalur utama bahwa sistem dapat menghapus hasil perhitungan dan kembali kehalaman awal sistem. Pengujian validasi jalur utama reset dijabarkan dalam Tabel 6.9.

Tabel 6.14 Pengujian validasi reset

Kode kebutuhan	TS-F-05
Prosedur pengujian	3. Pengguna menekan tombol Reset 4. Sistem menghapus hasil perhitungan dan kembali kehalaman awal sistem
Expected Result	Hasil perhitungan dapat dihapus dan kembali kehalaman awal sistem
Actual Result	Hasil perhitungan dapat dihapus dan kembali kehalaman awal sistem
Status	Valid

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil analisis kebutuhan sampai pada pengujian sistem maka dapat disimpulkan bahwa:

1. Pada tahap analisis kebutuhan sistem kakas bantu untuk penentuan prioritas test scenario berdasarkan UML Activity Diagram, diperoleh lima kebutuhan fungsional yang digunakan untuk melakukan perhitungan penentuan prioritas *test scenario* pada berkas activity diagram dengan ekstensi xml. Pada kebutuhan fungsional yang sudah ditentukan sebelumnya yaitu input berkas activity diagram, parsing xml, hitung prioritas, lihat detail, dan reset. Berdasarkan kebutuhan fungsional yang sudah didefinisikan maka akan dimodelkan dalam bentuk use case diagram dan use case scenario.
2. Pada tahap perancangan diperoleh hasil perancangan arsitektur yaitu sequence diagram dan class diagram. Kemudian perancangan komponen didapatkan rancangan algoritme. Dan selanjutnya rancangan antarmuka diperoleh tata letak didalam sistem. Setelah melakukan perancangan maka tahap implementasi menghasilkan spesifikasi sistem, implementasi algoritme menjadi kode program, dan menghasilkan antarmuka yang diimplementasikan dengan Unity3D.
3. Pada tahap pengujian diperoleh hasil pengujian unit dan pengujian validasi yang dilakukan pada tiga method mendapatkan nilai 100% valid pada setiap kasus uji.

7.2 Saran

Saran yang diberikan untuk pengembangan lebih lanjut sistem kakas bantu untuk penentuan prioritas *test scenario* berdasarkan UML activity diagram, adalah sebagai berikut:

1. Dapat menentukan main activity setiap node dan type of coupling (tipe kopling) antar node secara otomatis. Saat ini belum ada sistem yang dapat menentukan main activity dan *type of coupling* (tipe kopling) secara otomatis, masih menggunakan inputan dari pengguna.
2. Dapat menghitung penentuan prioritas *test scenario* dengan bahasa pemrograman lain selain C#.
3. Dapat menambahkan fitur visualisasi dalam bentuk flow graph secara otomatis dari hasil parsing berkas activity diagram.

DAFTAR PUSTAKA

Alomari, Z. (2015) 'Comparative Studies of Six Programming Languages Comparative Studies of Six Programming Languages', (April).

Ammar, et al. 2016. *Enhanced weighted method for test case prioritization in regression testing using unique priority value.* Tersedia di: <https://ieeexplore.ieee.org/> [Diakses 2 Agustus 2018]

Ansari, et al., 2016. *Optimized regression test using test case prioritization.* Tersedia di: <https://www.sciencedirect.com> [Diakses 19 Agustus 2018]

Anshori, Y. (2012) '*Menggunakan teknologi java dan XML untuk perangkat seluler*, 2(2), pp. 178–183.

Arwan, A. & Rusdianto, D. S. (2018) 'Automation of Independent Path Searching using Depth First Search', 3(1), pp. 104–112.

Baesens, B., Backiel, A. and Broucke, S. vanden (2015) *Beginning java® programming: The Object-Oriented Approach*. Crosspoint Boulevard Indianapolis, IN: John Wiley & Sons, Inc.

Bhuyan, P., Ray, A. & Das, M. (2017) 'Test scenario prioritization using UML use case and activity diagram', *Advances in Intelligent Systems and Computing*, pp. 499–512. doi: 10.1007/978-981-10-3874-7_47.

Caytiles, Ronnie D. & Lee, Sunguk., 2014. *A Review of an MVC Framework based Software Development*

Jauhari, M., Hamidin, D. & Rahmatuloh, M. (2017) 'Komparasi Stabilitas Eksekusi Kode Bahasa Pemrogrman .Net C# Versi 4.0.3019 Dengan Google Golang Versi 1.4.2 Menggunakan Algoritma Bubble Sort dan Insertion Sort', 9(1), pp. 13–20.

Jiang, B. & Chan.W.K., 2015. *Input-based adaptive randomized test case prioritization: A local beam search approach.* Tersedia di: <https://www.sciencedirect.com> [Diakses 19 Agustus 2018]

Khandelwal, E. & Bhadauria, M. (2013) 'Various Techniques Used For Prioritization of Test Cases', *International Journal of Scientific and Research Publications*, 3(6), pp. 3–6.

Korel, B. & Koutsogiannakis, G. (2009) 'Experimental comparison of code-based and model-based test prioritization', *IEEE International Conference on Software Testing, Verification, and Validation Workshops, ICSTW 2009*, pp. 77–84. doi: 10.1109/ICSTW.2009.45.

Mahali, P. & Acharya, A. A. (2013) 'Model Based Test Case Prioritization Using Uml Activity Diagram and Evolutionary Algorithm', *International Journal of Computer Science and Informatics*, (PRINT), pp. 2231–5292. Available at: https://pdfs.semanticscholar.org/e263/de716d26500be67881b13e33579ad5f746e4.pdf.

Maulana, Y. I. (2017) 'Perancangan Perangkat Lunak Sistem Informasi Pendataan Guru Dan Sekolah (Sindaru) Pada Dinas Pendidikan Kota Tangerang Selatan', *Jurnal Pilar Nusa Mandiri*, 13(1), pp. 21–27. Available at: http://ejournal.nusamandiri.ac.id/ejurnal/index.php/pilar/article/view/331.

P.G., S. & Mohanty, H. (2009) 'Prioritization of Scenarios Based on UML Activity Diagrams', *2009 First International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 271–276. doi: 10.1109/CICSYN.2009.74.

Pressman, R., S. 2010. *Software Engineering A Practitioner's Approach*. 7th Ed. New York: McGraw-Hill.

Putra, T. R., Siahaan, D. & Yuhana, U. L. (2014). Klasifikasi Severity Dari Bug Untuk Proyek Perangkat Lunak. Tersedia di: <https://www.researchgate.net/publication/266873248_KLASIFIKASI_SEVERITY_DARI_BUG_UNTUK_PROYEK_PERANGKAT_LUNAK> [Diakses 7 Agustus 2018]

Rathi, P. & Mehra, V. (2015) 'Analysis of Automation and Manual Testing Using Software Testing Tool', *International Journal of Innovations & Advancement in Computer Science*, 4(March), pp. 709–713.

Rava, M. & Wan-Kadir, W. M. N. (2016) 'A Review on Automated Regression Testing', *International Journal of Software Engineering and Its Applications*, 10(1), pp. 221–232. doi: 10.14257/ijseia.2016.10.1.21.

Sagar, G.P. & Prasad, P.V.R.D., *A survey on test case prioritization techniques for regression testing*. Tersedia di: <http://www.indjst.org> [Diakses 2 Agustus 2018]

Sapna, P. G. & Hrushikesha, M. (2009) 'Prioritizing use cases to aid ordering of scenarios', *EMS 2009 - UKSim 3rd European Modelling Symposium on Computer Modelling and Simulation*, pp. 136–141. doi: 10.1109/EMS.2009.78.

Sommerville, I. 2011. *Software engineering*. 9th ed. London: Addison-Wesley.

Sultan, Z. *et al.* (2017) 'Analytical Review on Test Cases Prioritization Techniques: An Empirical Study', *IJACSA) International Journal of Advanced Computer Science and Applications*, 8(2), pp. 293–302. doi: 10.14569/IJACSA.2017.080239.

Sumalatha, V.M. & Raju, G.S.V.P., 2013. *Model based test case generation from UML Activity Diagrams.* Tersedia di: <https://pdfs.semanticscholar.org/> [Diakses 8 Agustus 2018]

Tej, M. V. U. *et al.* (2011) 'Analyzing XML Parsers Performance for Android Platform', 2(3), pp. 973–976.

Ubaidillah, M., Pradana. F. & Priyambadha. B. (2017) 'Kakas Bantu Perhitungan Nilai Kopling Menggunakan Metrik Cognitive Weighted Coupling Between Object', 6(1), pp. 90–95.

Whitten, J. L. & Lonnie D. Bentley (2007) *Systems Analysis and Design Methods*, *McGraw-Hill*. doi: 10.1007/s13398-014-0173-7.2.

Widodo, W. *et al.* (2016) 'Evaluasi Proses Pengembangan Perangkat Lunak', 10(1), pp. 1140–1148.