BAB II
TINJAUAN PUSTAKA

2.1 Metode Pengembangan Perangkat Lunak

Dalam rekayasa perangkat lunak terdapat metode-metode pengembangan perangkat lunak yang bisa digunakan, diantaranya System Development Life Cycle (SDLC), Evolutionary Development, RAD, Extreme Programming dan lain – lain. Metode pengembangan perangkat lunak yang digunakan pada penelitian ini adalah System Development Life Cycle (SDLC).

2.1.1 System Development Life Cycle (SDLC)

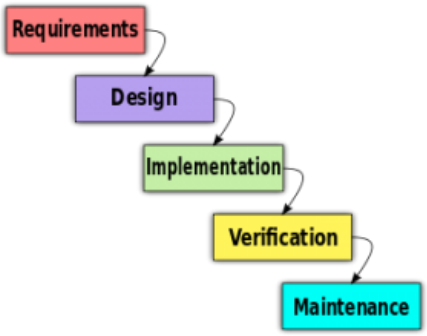
System Development Life Cycle (SDLC) adalah suatu kerangka yang menggambarkan kegiatan-kegiatan yang dilakukan pada setiap tahap pembuatan sebuah software (Fatta, 2007).

Model Waterfall

Model Waterfall adalah sebuah metode pengembangan software yang bersifat sekuensial. Metode ini dikenalkan oleh Royce pada tahun 1970 dan pada saat itu disebut sebagai isi klus klasik dan sekarang ini lebih dikenal dengan sekuensial linier. Selain itu Model ini merupakan model yang paling banyak dipakai oleh para pengembang software. Inti dari metodewaterfall adalah pengerjaan dari suatu system dilakukan secara berurutan atau secara linear. Jadi jika langkah satu belum dikerjakan maka tidak akan bisa melakukan pengerjaan langkah 2, 3 dan seterusnya. Secara otomatis tahapan ke-3 akan bisa dilakukan jika tahap ke-1 dan ke-2 sudah dilakukan.

Keterkaitan dan pengaruh antar tahap ini ada karena output sebuah tahap dalam Waterfall Model merupakan input bagi tahap berikutnya, dengan demikian ketidak sempurnaan hasil pelaksanaan tahap sebelumnya adalah awal ketidak sempurnaan tahap berikutnya. Memperhatikan karakteristik ini, sangat penting bagi tim pengembang dan perusahaan untuk secara bersama-sama melakukan analisa kebutuhan dan desain system sesempurna mungkin sebelum masuk kedalam tahap

penulisan kode program. Secara garis besar, berikut penjelasan mengenai fase-fase dalam waterfall pada gambar di bawah ini.



Gambar 2.1 Fase-Fase Waterfall

1. Analisa Kebutuhan (Requirement Analysis)

Langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa malakukan sebuah penelitian, wawancara atau studi literatur. Seorang sistem analis akan menggali informasi sebanyak-banyaknya dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen *user requirement* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen inilah yang akan menjadi acuan sistem analis untuk menterjemahkan ke dalam bahasa pemprogram.

2. Desain Sistem (System Design)

Proses desain akan menerjemahkan syarat kebutuhan kepada perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat pengkodean. Proses ini berfokus pada struktur data, arsitektur perangkat lunak, representasi *interface*, dan (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut *software requirment*. Dokumen inilah yang akan digunakan *programmer* untuk melakukan aktivitas pembuatan sistemnya.

3. Implementasi / Penulisan Kode Program (Implementation)

Penulisan kode program merupakan penerjemahan desain dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh *programmer* yang akan menterjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan pengujian terhadap sistem yang telah dibuat tadi. Tujuan pengujian adalah menemukan kesalahan-kesalahan terhadap sistem tersebut dan kemudian bisa diperbaiki.

4. Penerapan / Pengujian Program (Integration & Testing)

Tahapan ini bisa dikatakan akhir dalam pembuatan sebuah sistem. Setelah melakukan analisa, desain dan pengkodean maka sistem yang sudah jadi akan digunakan oleh *user*.

Keuntungan Model Waterfall

1. Kualitas dari sistem yang dihasilkan akan baik. Ini dikarenakan oleh pelaksanaannya secara bertahap. Sehingga tidak terfokus pada tahapan tertentu.
2. Dokumen pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya. Jadi setiap fase atau tahapan akan mempunyai dokumen tertentu.
3. Metode ini masih lebih baik digunakan walaupun sudah tergolong kuno, daripada menggunakan pendekatan asal-asalan. Selain itu, metode ini juga masih masuk akal jika kebutuhan sudah diketahui dengan baik.

Kelemahan Model Waterfall

1. Diperlukan majemen yang baik, karena proses pengembangan tidak dapat dilakukan secara berulang sebelum terjadinya suatu produk.
2. Kesalahan kecil akan menjadi masalah besar jika tidak diketahui sejak awal pengembangan yang berakibat pada tahapan selanjutnya.
3. Pelanggan sulit menyatakan kebutuhan secara eksplisit sehingga tidak dapat mengakomodasi ketidak pastian pada saat awal pengembangan.

2.2 Unified Modelling Language (UML)

Menurut Lethbridge (2011) Unified Modeling Language (UML) adalah Bahasa grafis standar untuk pembuatan/pemodelan perangkat lunak yang berorientasi pada objek. UML dikembangkan pada pertengahan 1990an oleh kolaborasi James Rumbaugh, Grady Booch dan Ivar Jacobson yang masin-masing telah memulai awal penelitiannya pada awal 1990an. Huruf 'U' dalam 'UML' adalah singkatan dari 'Unified' (terpadu) karena UML merupakan gabungan fitur-fitur terbaik yang pernah dibuat oleh tiga pengembang tersebut. Adapun beberapa diagram yang terdapat dalam UML yang akan digunakan antara lain:

2.2.1 Use Case Diagram

Menurut Bennett (2005), *Use Case Diagram* digunakan untuk menunjukan bahwa secara fungsional sistem akan menyediakan dan menunjukan mana *user* dan bagaimana berkomunikasi dengan sistem. Bagian – bagian dari *use case diagram*:

a. *Use Case*

Use case digunakan untuk menggambarkan deskripsi fungsional dari sistem dari perspektif *user* (*user*), yang berisi satu set perilaku terkait transaksi yang biasanya dilakukan bersama-sama untuk menghasilkan nilai bagi *user*.

b. *Actor*

Actor mewakili peran orang, sistem lain, perangkat lain, ketika berkomunkasi dengan kasus penggunaan tertentu dalam sistem.

c. *Communication Association*

Merupakan contoh koneksi logis antara *actor* dengan *use case*.

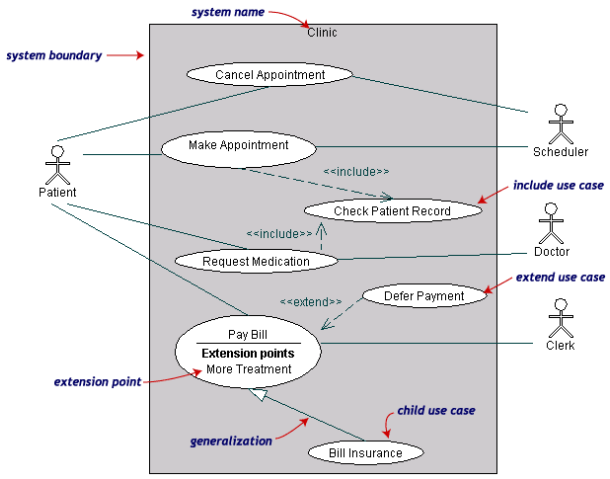
d. *Subsystem Boundary*

Boundary mendandakan batasan, pengklarifikasian komponen atau subsistem pada *use case*.

e. *Extend and Include Relationship*

Extend digunakan ketika anda ingin menunjukan bahwa *use case* memberikan tambahan fungsional yang mungkin diperlukan dalam penggunaan *use case* lain.

Include digunakan ketika ada urutan perilaku (*use case*) yang digunakan dalam sejumlah kasus, dan anda ingin menghindari menyalin deskripsi yang sama itu ke dalam setiap *use case* yang digunakan. Atau bisa dikatakan bahwa *include* menggambarkan *use case* yang dimasukkan didalamnya perilaku dari *use case* lain. Contoh use case diagram dapat dilihat pada gambar 2.2.



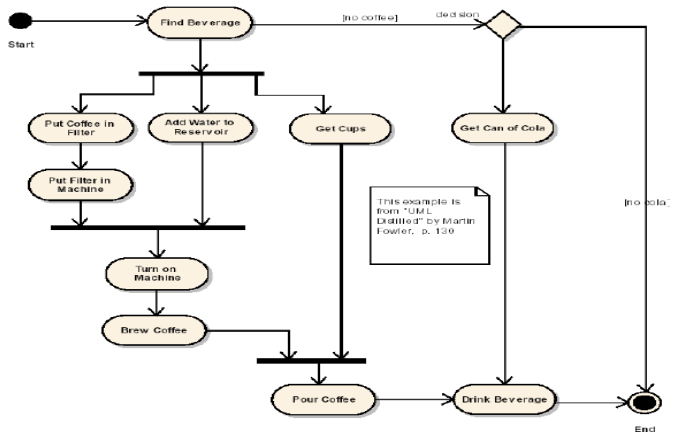
Gambar 2.2 Contoh *Use Case Diagram*

2.2.2 Activity Diagram

Menurut Bennet (2005), *Activity diagram* digunakan untuk model proses tugas dalam permodelan bisnis, juga untuk mendeskripsikan fungsi sistem yang diwakili oleh *use case*, dan dalam spesifikasi operasi untuk menggambarkan logika operasi.

Activity diagram terdiri dari *state* awal *(initial state)* dan *state* akhir *(final state)*, *state* awal menggambarkan bagaimana aktivitas sistem dimulai dan *state* akhir menggambarkan bagaimana aktivitas sistem berakhir. *Activity* digambarkan dengan persegi panjang bulat, yang mencerminkan aktivitas yang terjadi dan bersifat fisik. Untuk menunjukan aliran kontrol dari suatu tindakan ke tindakan yang berikutnya menggunakan *control flow*, yang digambarkan dengan garis serta mata panah. *Partition* ditampilkan baik sebagai *horizontal* maupun *vertikal*, partisi digunakan untuk tindakan terpisah dalam suatu kegiatan. *Decision* didefinisikan sebagai node keputusan, kontrol arus datang dari node keputusan memiliki kondisi yang memungkinkan mengalir apabila kondisi tersebut terpenuhi. Digambarkan

dengan bentuk berlian. *Forks* dan *Join node* memiliki notasi yang sama baik *vertikal* maupun *horizontal*, yang menunjukan awal dan akhir dari trit kontrol secara bersamaan.



Gambar 2.3 Contoh *Activity Diagram*

2.2.3 Class Diagram

Menurut Bennet (2005) *Class diagram* merupakan UML struktur diagram yang menunjukan kelas dengan atribut dan menjalankan operasi, bersama dengan asosiasi antar kelas. Bagian – bagian dari *class diagram*:

a. *Class*

Class merupakan kumpulan objek yang secara logis sama dalam hal perilaku dan struktur data mereka.

b. *Attributes*

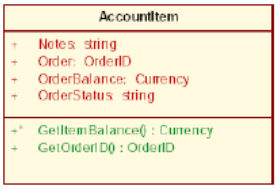
Atribut adalah struktur umum yang dapat *member class* ketahui, setiap objek akan memiliki atribut sendiri, mungkin unik. Bersama – sama dengan operasi untuk mendefinisikan kelas.

c. *Operations*

Merupakan apsek dari perilaku yang mendefinisikan sebuah kelas, untuk membuat spesifikasi dari unsur fungsionalitas sistem yang akan diimplementasikan sebagai metode objek.

Atribut dan *method* (Gambar 2.4) dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar class yang bersangkutan
b. *Protected*, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
c. *Public*, dapat dipanggil oleh siapa saja

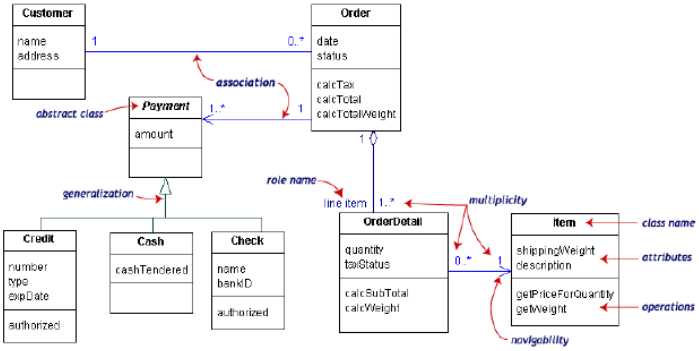


Gambar 2.4 *Atribut* dan *Method* pada *Class Diagram*

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki *method*. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi *method* pada saat *run-time*. Sesuai dengan perkembangan *class model*, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

Hubungan antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua *atribut* dan *method class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain.



Gambar 2.5 Contoh *Class Diagram*

2.2.4 Entity Relationship Diagram

Model *Entity Relationship* diperkenalkan pertama kali oleh P.P. Chen pada tahun 1976. Model ini dirancang untuk menggambarkan persepsi dari pemakai dan berisi obyek-obyek dasar yang disebut entity dan hubungan antar entity-entity tersebut yang disebut relationship. Pada model ER ini semesta data yang ada dalam dunia nyata ditransformasikan dengan memanfaatkan perangkat konseptual menjadi sebuah diagram, yaitu diagram ER (*Entity Relationship*). Diagram *Entity-Relationship* melengkapi penggambaran grafik dari struktur logika. Dengan kata lain Diagram E-R menggambarkan arti dari aspek data seperti bagaimana entity-entity, atribut-atribut dan relationship-relationship disajikan. Sebelum membuat Diagram E-R, tentunya kita harus memahami betul data yang diperlukan dan ruang lingkupnya. Di dalam pembuatan diagram E-R perlu diperhatikan penentuan sesuatu konsep apakah merupakan suatu entity, atribut atau relationship.

Tipe Entity

Entity adalah obyek yang dapat dibedakan dengan yang lain dalam dunia nyata. *Entity* dapat berupa obyek secara fisik seperti orang, rumah, atau kendaraan. *Entity* dapat pula berupa obyek secara konsep seperti pekerjaan, perusahaan, dan sebagainya. *Entity* disajikan dalam bentuk persegi panjang, entity kuat disajikan dengan perseg panjang dengan satu garis, sedangkan entity lemah disajikan dengan persegi panjang dobel.

Atribut

Atribut adalah karakteristik dari entity atau relationship, yang menyediakan penjelasan detail tentang entity atau relationship tersebut. Nilai Atribut merupakan suatu data aktual atau informasi yang disimpan pada suatu atribut di dalam suatu entity atau relationship. Atribut digambarkan dalam bentuk oval.

Relationship

Relationship adalah hubungan yang terjadi antara satu atau lebih entity. *Relationship set* adalah kumpulan *relationship* yang sejenis. Pada *relationship* terdapat derajat dari *relationship* menjelaskan jumlah *entity* yang berpartisipasi dalam suatu *relationship*. Terdapat tiga jenis derajat dari *relationship*, *unary degree* (derajat satu), *binary degree* (derajat dua) dan *ternary degree* (derajat tiga).

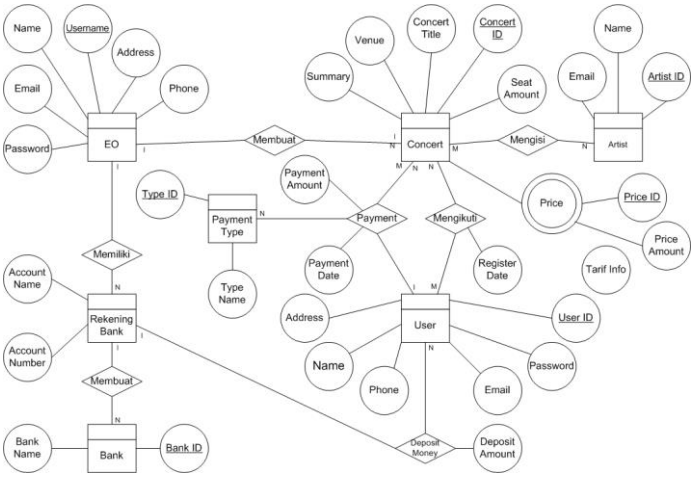
Cardinality Ratio Contraint

Cardinality ratio contraint merupakan menjelaskan batasan jumlah keterhubungan satu *entity* dengan *entity* lainnya. Terdapat tiga Jenis *cardinality ratio contraints*, satu pada satu (1:1), satu pada banyak (1:N/ N:1) dan banyak pada banyak (M:N).

Participation constraint

Participation constraint merupakan batasan yang menjelaskan apakah keberadaan suatu *entity* tergantung pada hubungannya dengan *entity* lain .Terdapat 2 macam *participation constraint*, *total participation contraints* dan *partial participation constraints*.

Berikut ini adalah contoh dari ERD ditunjukkan pada gambar 2.6.



Gambar 2.6 Contoh *Entity Relationship Diagram*

2.3 Metode Simple Additive Weighting (SAW)

Salah satu metode yang digunakan untuk menyelesaikan masalah dari Fuzzy Multiple Attribute Decision Making (FMADM) adalah metode Simple Additive Weighting (SAW) yaitu suatu metode yang digunakan untuk mencari alternatif optimal dari sejumlah alternatif dengan kriteria tertentu.

Definisi Metode Simple Additive Weighting (SAW) sering juga dikenal istilah metode penjumlahan terbobot. Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut (Pahlevy. 2010). Metode SAW membutuhkan proses normalisasi matriks keputusan X ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada. Formula untuk melakukan normalisasi tersebut adalah sebagai berikut (Kusumadewi, dkk, 2006):

Formula Normalisasi

$$r_{ij} = \begin{cases} \dfrac{x_{ij}}{\text{Max}_i\, x_{ij}} & \text{jika j adalah atribut keuntungan (benefit)} \\ \dfrac{\text{Min}_i\, x_{ij}}{x_{ij}} & \text{jika j adalah atribut biaya (cost)} \end{cases}$$

Dimana r_{ij} adalah rating kinerja ternormalisasi dari alternatif Ai pada atribut Cj; i=1,2,...,m dan j=1,2,...,n.

Nilai preferensi untuk setiap alternatif (Vi) diberikan sebagai:

$$V_i = \sum_{j=1}^{n} w_j \, r_{ij}$$

Dimana :

V_i = nilai prefensi

w_j = bobot ranking

r_{ij} = rating kinerja ternormalisasi

Nilai Vi yang lebih besar mengindikasikan bahwa alternatif Ai lebih terpilih.

Langkah-langkah dari metode SAW adalah:

1. Menentukan kriteria-kriteria (C) yang akan dijadikan acuan dalam pengambilan
   keputusan.
2. Menentukan rating kecocokan setiap alternatif pada setiap kriteria.
3. Membuat matriks keputusan berdasarkan kriteria (C), kemudian melakukan normalisasi matriks berdasarkan persamaan yang disesuaikan dengan jenis atribut (atribut keuntungan ataupun atribut biaya) sehingga diperoleh matriks ternormalisasi R.
4. Hasil akhir diperoleh dari proses perankingan yaitu penjumlahan dari perkalian matriks ternormalisasi R dengan vector bobot sehingga diperoleh nilai terbesar yang dipilih sebagai alternatif terbaik (A) sebagai solusi (Kusumadewi, dkk. 2006).

Kelebihan Metode SAW

Kelebihan dari metode simple additive weighting dibanding dengan model pengambil keputusan lainnya terletak pada kemampuannya untuk melakukan penilaian secara lebih tepat karena didasarkan pada nilai kriteria dan bobot preferensi yang sudah ditentukan, selain itu SAW juga dapat menyeleksi alternatif terbaik dari sejumlah alternatif yang ada karena adanya proses perankingan setelah menentukan bobot untuk setiap atribut (Kusumadewi, dkk. 2006).

2.4 Analisis dan Perancangan Sistem Berorientasi Objek

Analisis berorientasi objek adalah cara baru dalam memikirkan sebuah masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan penggabungan antar struktur data dan perilaku dalam sebuah entitas. Analisa berorientasi objek dimulai denganmenyatakan suatu masalah, analisis menggambarkan model situasi dari dunia nyata, menggambarkan sifat yang penting (Pressman R. S., 2010)

Komponen utama pada analisa berorientasi objek adalah :

a. Kelas yaitu definisi abstrak dari sebuah objek, dimana dijelaskan bahwa struktur dan perilaku dari tiap objek tergabung dalam satu kelas.
b. Objek : Merepresentasikan sebuah entitas, baik secara fisik, konsep ataupun secara software.
c. Atribut : Nama-nama property dari sebuah kelas yang menjelaskan batasan nilainya dari properti yang dimiliki oleh sebuah kelas tersebut.

Dalam melakukan analisa dan perancangan sistem berorientasi obyek penulis menggunakan UML (Unified Modelling Language) untuk memodelkannya.

2.5 Tinjauan Studi

1. **Penentuan Distribusi Air Bersih Di Kabupaten X Menggunakan Metode SAW (Agustina & Bakti, 2015).**

Penulis akan membuat “Model Sistem Pendukung Keputusan Penentuan Prioritas Penanganan Air Bersih Pada Dinas Pekerjaan Umum Cipta Karya dan Tata Ruang Kabupaten Lahat Menggunakan Metode SAW ini diharapkan dapat membantu Pekerjaan Umum Cipta Karya dan Tata Ruang Kabupaten Lahat dalam merekap data hasil survei dan mampu mendukung pengambilan suatu keputusan untuk menentukan prioritas penanganan air bersih. Dengan merumuskan kriteria yanokg terdiri atas jumlah penduduk, kebutuhan air, debit air, jarak pipanisasi, beda tinggi, gesekan dan tekanan.

2. ***Flood-prone Areas Mapping at Semarang City By Using Simple Additive Weighting Method*** **(Setyani & Saputra, 2015).**

Penelitian ini mengembangkan Pemetaan Daerah Rawan Banjir yang dianalisis sebagai alasan banjir lokal. Banjir lokal hanya terjadi di tempat-tempat tertentu dimana hujan turun. Kriteria yang digunakan adalah hujan turun, topografi, drainase, dan penggunaan lahan. Ada banyak aspek yang harus dianalisis, sehingga penggunaannya Simple Additive Weighting untuk menentukan daerah rawan banjir. Kelebihan pemetaan daerah rawan banjir ini pengguna dapat dengan mudah mengakses informasi tentang daerah rawan banjir yang akan ditampilkan dalam bentuk peta, grafik, dan tabel.

3. **Aplikasi Pencarian Informasi Dan Lokasi Tempat Makan Pada Perangkat Mobile Berbasis Android (Layona & Yulianto, 2016).**

Hasil dari penelitian ini berupa aplikasi bernama "Nomnom" pada perangkat mobile berbasis Android yang dapat menampilkan hasil pencarian informasi dan lokasi tempat makan yang diinginkan konsumen. Simpulan dari penelitian ini adalah aplikasi "Nomnom" dapat membantu konsumen dalam mendapatkan informasi dan lokasi tempat makan.