BAB II
LANDASAN TEORI

2.1 Citra

Prasetyo (2011,p.1) menguraikan bahwa sebuah citra dapat di definisikan sebagai fungsi dua demensi f(x,y),dimana x dan y adalah koordinat spasial,dan amplitudo dari f pada sembarang kordinat (x,y) disebut *intensity* (intensitas) atau *gray level* (level keabuan) dari citra pada titik tersebut. Ketika x,y dan nilai intensitas dari f adalah semua terbatas, *discrete quantities,* kita sebut citra tersebut *digital image* (citra digital). Citra digital terdiri dari sejumlah elemen tertentu, setiap elemen mempunyai lokasi dan nilai tertentu.

Lestari (2012,p.2) menguraikan bahwa Citra sebagai keluaran dari suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal *video* seperti gambar pada monitor televisi, dan bersifat digital yang dapat langsung disimpan pada suatu pita magnetik. Citra sebagai salah satu komponen multimedia memegang peranan penting sebagai bentuk informasi *Visual*. Citra mempunyai karakteristik yang tidak dimiliki oleh data *teks*, yaitu citra kaya dengan informasi.

Arifin (2009,p.15) menjelaskan bahwa citra merupakan suatu keluaran dari suatu sistem perekaman data yang bersifat optik, analog ataupun digital. Perekaman data citra dapat dibagi menjadi dua yaitu:

a) Citra Analog

Citra analog yaitu terdiri dari sinyal-sinyal elektromagnetik yang tidak dapat dibedakan sehingga pada umumnya tidak dapat ditentukan ukurannya. Citra analog mempunyai fungsi yang kontinu. Hasil perekaman citra analog dapat bersifat optik yakni berupa foto (film foto konvensional) dan bersifat sinyal video seperti gambar pada monitor televisi.

b) Citra Digital

Citra digital terdiri dari sinyal-sinyal yang dapat dibedakan dan mempunyai fungsi yang tidak kontinu yakni berupa titik-titik warna pembentuk citra. Hasil perekaman citra digital dapat disimpan pada suatu media magnetik.

2.1.1 Citra Warna

Arifin (2009,p.16) menguraikan bahwa citra warna adalah citra dengan sistem grafik yang memiliki satu set nilai tersusun (*aset of ordered values*) yang menyatakan berbagai tingkat warna. Citra warna bukanlah seperti citra *grayscale*. Dimana setiap set nilai tersusun mewakili satu *'scale*' warna atau '*hue*'.

Agus (2013,p.309) menjelaskan bahwa Citra yang memiliki warna *grayscale* cenderung kurang menarik utuk dilihat dibandingkan dengan citra berwarna, karena kamera pada jaman dahulu hanya mampu menghasilkan citra dengan format warna *grayscale*, sehingga hasil citra tersebut menjadi kurang menarik untuk dilihat. Padahal, banyak citra zaman dahulu memiliki nilai sejarah yang cukup tinggi yang semestinya disampaikan dari generasi kegenerasi.

Gonzales yang diterjemahkan oleh Arifin (2009,p.17) menguraikan bahwa sistem yang dipakai untuk mewakili warna yaitu sistem RGB (*Red, Green,Blue*). Sistem RGB adalah sistem penggabungan antara warna-warna primer (*additiveprimary colours*) yaitu merah (*Red*), hijau (*Green*) dan biru (*Blue*) untuk memperolehwarna tertentu. Misalnya warna putih diperoleh dari hasil gabungan warna merah =255, hijau = 255, dan biru = 255. Dalam sistem RGB, warna putih cerah dinyatakan dengan RGB (255, 255, 255). *Range* nilai dari setiap warna primer adalah 0 sampai 255. Sehingga kemungkinan warna yang dapat terbentuk dengan sistem RGB adalah 256 x 256 x 256 yakni kurang lebih 16.7 juta warna. Tabel kode warna seperti tertampil pada tabel 2.1 berikut diperlihatkan beberapa kode warna hasil gabungan warna RGB.

Tabel 2.1 Kode Warna dalam Nilai Integer

Warna	Merah	Hijau	Biru
Hitam	0	0	0
Biru	0	0	255
Hijau	0	255	0
Cyan (Biru+Hijau)	0	255	255
Merah	255	0	0
Magenta (Merah+Biru)	255	0	255
Kuning (Merah+Hijau)	255	255	0
Putih (Merah+Hijau+Biru)	255	255	255
Abu-abu	128	128	128

2.1.2 Representasi Citra Digital

Prasetyo (2011,p.14) menguraikan bahwa sebuah citra dapat didefinisikan sebagai fungsi dua dimensi f(x,y), dimana x dan y adalah kordinat spesial, dan amplitudo dari f pada sembarang pasangan kordinat (x,y) disebut intensitas citra (level keabuan) pada titik tersbut. Warna citra di bentuk oleh kombinasi citra 2-D individual. Misalnya, dalam sistem warna RGB, warna citra terdiri dari tiga komponen individu (red, green, blue). Untuk alasan ini, banyak cara di kembangkan untuk citra monokrom dapat di perluas ke citra berwarna oleh pemorosesan tiga komponen citra.

Arifin (2009,p.21) menjelaskan bahwa resolusi gambar dikatakan sebagai jumlah *pixel* yang terkandung di dalam suatu citra. Pada resolusi rendah keterperincian dan kedalaman citra akan hilang sama sekali dimana *pixel-pixel* individu jelas kelihatan, pada resolusi tinggi keterperincian data lebih nyata dan tajam. *Aspect Ratio* adalah suatu bilangan yang dapat diperoleh bila bilangan *pixel* mendatar dibagi dengan bilangan *pixel* tegak. Dalam pembesaran resolusi citra *Aspect Ratio* perlu sama agar citra tidak kelihatan *distorted* (menyimpang) dan alami.

Eka (2012,p.2) menjelaskan bahwa pembesaran citra adalah suatu proses yang dilakukan untuk memperbesar suatu citra digital dari ukuran semula menjadi ukuran yang berbeda sesuai dengan faktor pembesaran yang dinginkan. Proses ini memiliki dua langkah yaitu pembuatan lokasi *píxel* yang baru dan penempatan warna yang berdasarkan kepada nilai *gray level* terhadap lokasi baru yang dibuat sebelumnya.

2.2 Dasar - Dasar Warna

Prestyo (2011,p.177) menjelaskan bahwa penggunaan warna dalam pemerosesan citra dimotivasi oleh dua faktor utama. Pertama, warna adalah deskriptor yang powerful yang sering menyederhanakan indentifikasi objek dan pengekstrakan dari *scene*. Kedua, manusia dapat melihat dengan jelas ribuan beentuk warna dan intensitas, dibandingkan dengan hanya dua lusin bentuk *gray*. Faktor kedua ini penting dalam tuntunan analisis citra. Walaupun proses diikuti oleh otak manusia dalam merasakan dan menginterprestasikan warna adalah fenomena *physiopsychological* yang belum sepenuhnya dimengerti, kelainan fisik warna dapat diekpresikan pada basis formal yang didukung oleh experimen dan teori.

2.2.1 Model Warna

Prasetyo (2011,p.181) menjelaskan bahwa tujuan dari model warna adalah untuk menfasilitasi spesifikasi warna dalam beberapa standar. Esensinya, model warna adalah representasi sistem kordinat dan *sub - space* di dalam sistem di mana setiap warna direpresentasikan oleh titik tunggal.

2.3 Digital

Digital berasal dari kata Digitus, dalam bahasa yunani yang berartikan jari – jemari. Apabila jari – jemari seseorang dihitung, maka akan berjumlah sepuluh (10). Nilai sepuluh tersebut terdiri dari 2 radix, yaitu 1 dan 0. Oleh karena itu digital merupakan penggambaran dari suatu kondisi bilangan yang terdiri dari angka 0 dan 1 atau *OFF* dan *ON* (*system* bilangan biner), dapat juga disebut dengan istilah bit, (*BINARY DIGIT)*. Semua *system* komputer menggunakan *system* digital sebagai basis data nya.

2.4 *Format File* Gambar

Andreswari (2001,p.22) menjelaskan bahwa pada umumnya *file* gambar digunakan untuk menyimpan gambar yang ditampilkan dilayar ke dalam suatu media penyimpanan data. Untuk menyimpan sebuah *file* gambar ini digunakan salah satu format *file*. Ada banyak format *file* gambar yang dapat digunakan untuk menyimpan *file* gambar, salah satunya adalah *BMP*.

2.4.1 *Format File* BMP *(Bitmap)*

Format file BMP merupakan format *grafis* yang fleksibel untuk *platform Windows* sehingga dapat dibaca oleh program *grafis* manapun. Hal tersebut didukung kuat oleh Christian (2012,p.24) dengan berpendapat bahwa, *format* ini mampu menyimpan informasi dengan kualitas tingkat 1 bit sampai 24 bit. *Format file* ini mampu menyimpan gambar dalam mode warna RGB, *Grayscale, Indexed Color,* dan *Bitmap*.

Kelemahan *format file* ini adalah tidak mampu menyimpan *alpha channel* serta ada kendala dalam pertukaran *platform*. Kelebihan *tipe file* BMP adalah dapat dibuka oleh hampir semua program pengolah gambar. Baik *file* BMP yang terkompresi maupun tidak terkompresi, *file* BMP memiliki ukuran yang jauh lebih besar daripada tipe-tipe yang lain.

2.4.2 *Format File JPG/JPEG (Joint Photographic Experts Group)*

Christian (2012 , p.25) menjelaskan bahwa *Joint Photograpic Experts* (*JPEG* , dibaca jay-peg) di rancang untuk kompresi beberapa *full-color* atau *gray-scale* dari suatu gambar yang asli, seperti pemandangan asli di dunia ini. *JPEG* bekerja dengan baik pada *continous tone images* seperi *photographs* tetapi tidak terlalu bagus pada ketajaman gambar dan seni pewarnaan seperti penulisan, kartun yang sederhana atau gambar yang mengunakan banyak garis. *JPEG* sudah mendukung untuk *24-bit color depth* atau sama dengan 16,7 juta warna (224 = 16.777.216 warna).

2.5 Area Penggunaan Pengolahan Citra Digital

Prasetyo (2011,P.2) menjelaskan bahwa cara paling mudah untuk mengembangkan pemahaman dasar tingkat aplikasi pengolahan citra digital adalah dengan mengklasifikasikan citra menurut sumbernya (misalnya, *Visual*, dan *x-ray*). Sumber energi citra yang digunakan saat ini adalah energi spektrum elektromagnetik.

Achmad (2013,p.4) menjelaskan bahwa suatu citra digital melalui pengolahan citra digital (*digital image processing*) dapat menghasilkan citra digital yang baru, termasuk didalamnya perbaikan citra (*image restoration*) dan peningkatan kualitas citra (*image enhancement*).

2.5.1 Citra Berskala Keabuan

Kadir (2013, p.23) menjelaskan bahwa sesuai dengan nama yang melekat, citra jenis ini menangani gradasi warna hitam dan putih, yang tentu saja menghasilkan efek warna abu-abu. Pada jenis gambar ini, warna dinyatakan dengan intensitas. Dalam hal ini, intensitas berkisar antara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih.

2.5.2 Citra Biner

Kadir (2013,p.23) menjelaskan bahwa citra biner adalah citra dengan setiap *pixel* hanya dinyatakan dengan sebuah nilai dari dua kemungkinan (yaitu nilai 0 dan 1). Nilai 0 menyatakan warna hitam dan nilai 1 menyatakan warna putih. Citra jenis ini banyak dipakai dalam pemrosesan citra, misalnya untuk kepentingan memperoleh tepi bentuk suatu objek.

2.6 Pengertian *Template Matching*

Template matching adalah sebuah teknik dalam pengolahan citra digital untuk menemukan bagian-bagian kecil dari gambar yang cocok dengan *template* gambar. *Template matching* merupakan salah satu ide yang digunakan untuk menjelaskan bagaimana otak kita mengenali kembali bentuk-bentuk atau pola-pola. *Template* dalam konteks rekognisi pola menunjuk pada konstruk internal yang jika cocok (*match*) dengan stimulus penginderaan mengantar pada rekognisi

suatu objek. Atau pengenalan pola terjadi jika terjadi kesesuaian antara stimulus indera dengan bentuk mental internal. Gagasan ini mendukung bahwa sejumlah besar *template* telah tercipta melalui pengalaman hidup kita. Tiap-tiap *template* berhubungan dengan suatu makna tertentu.

Teori *Template matching* memiliki keunggulan dan kelemahan, yaitu :
Keunggulan :

(1) Jelas bahwa untuk mengenal bentuk, huruf atau bentuk-bentuk *Visual* lainnya diperlukan kontak dengan bentuk-bentuk internal.
(2) *Template matching* adalah prosedur pengenalan pola yang sederhana yang didasarkan pada ketepatan konfigurasi informasi penginderaan dengan "konfigurasi" pada otak. (Contohnya : kecacatan papan PCB)

Kelemahan :
Jika perbandingan eksternal objek dgn internal objek 1:1, maka objek yang berbeda sedikit saja dengan *template* tidak akan dikenali. Oleh karena itu, jutaan *template* yang spesifik perlu dibuat agar cocok dengan berbagai bentuk *geometri* yang kita lihat dan kenal. Jika memang penyimpanan memori di otak seperti ini, otak tentu seharusnya sangat kewalahan dan pencarian informasi akan memakan waktu, padahal pada kenyataannya tidak demikian.

Template Matching dapat dibagi antara dua pendekatan, yaitu : pendekatan berbasis fitur dan pendekatan berbasis *template*. Pendekatan berbasis fitur menggunakan fitur pencarian dan *template* gambar seperti tepi atau sudut, sebagai pembanding pengukuran matrik untuk menemukan lokasi *template matching* yang terbagus di sumber gambar. Contoh Aplikasi "PENGGUNAAN METODE *TEMPLETE MATCHING* UNTUK IDENTIFIKASI KECACATAN PADA PCB" Penerapan metode *template matching* pada identifikasi kecacatan PCB dapat dilakukan langkah utama sebagai berikut :

1. Pengepasan posisi: Dilakukan dengan mencuplik 80% area citra untuk mendapatkan posisi ideal.

2. Hitung nilai korelasi silang : Untuk mengklasifikasikan suatu citra PCB adalah baik dan tanpa cacat sedikitpun, maka nilai korelasi adalah 1 dan cacat total maka nilai korelasinya adalah -1. Rumus yang digunakan dapat dilihat pada gambar 2.1.

$$r = \frac{\sum_{i=0}^{N-1}(x_i - \bar{x}).(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{N-1}(x_i - \bar{x})^2 . \sum_{i=0}^{N-1}(y_i - \bar{y})^2 .}}$$

Gambar 2.1 Rumus Menghitung Nilai Korelasi Silang

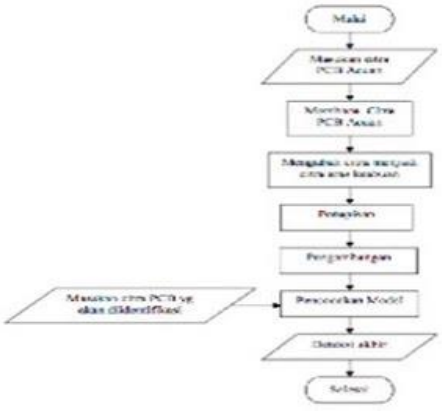
3. Deteksi akhir : Dari nilai korelasi yang didapat, nilai tersebut kemudian di konversikan dalam rentang 0 sampai 255 pada channel red untuk digambarkan dalam bentuk segiempat pada titik koordinat citra PCB yang mengalami cacat. Prinsip metode ini adalah membandingkan antara image objek yang akan dikenali dengan image *template* yang ada. *Image* objek yang akan dikenali mempunyai tingkat kemiripan sendiri terhadap masing-masing *image template*. Pengenalan dilakukan dengan melihat nilai tingkat kemiripan tertinggi dan nilai batas ambang pengenalan dari image objek tersebut. Bila nilai tingkat kemiripan berada di bawah nilai batas ambang maka *image* objek tersebut dikategorikan sebagai objek tidak dikenal. Selanjutnya untuk dapat mengimplementasikan metode *templete matching* maka perlu dilakukan sejumlah operasi pengolahan citra digital, antara lain:

   • Penapisan Citra (*Filtering*) : dilakukan bila citra yang akan dianalisis memiliki derau sehingga perlu dihaluskan dengan tapis citra. Perancangan tapis dengan memanipulasi piksel-piksel tetangga membuat citra lebih halus, bentuk sudut, dan tepi citra tetap terjaga. Pada proses perekaman citra digital dapat terjadi gangguan yang bersifat frekuensi rendah, dimana

terjadi proses pemerataan intensitas cahaya pada suatu titik sampel dengan titik-titik tetangganya. Gangguan lain yang sering terjadi pada proses perekaman citra digital adalah terjadinya gangguan berbentuk garis-garis akibat adanya kerusakan pada sebagian detektor sensor. Juga sering dijumpai gangguan lain dalam bentuk bercak hitam yang acak.

• Pengambangan (*Tresholding*) : digunakan untuk mengubah citra dengan format keabuan yang mempunyai nilai lebih dari dua ke format citra biner yang hanya memiliki dua nilai (0 atau 1). Dalam hal ini titik dengan rentang nilai keabuan tertentu diubah menjadi warna hitam dan sisanya menjadi warna putih atau sebaliknya.Aplikasi yang akan dibangun adalah sebuah simulasi sederhana untuk proses AOI (*Automated Optical Inspection*), dalam hal ini diasumsikan PCB master dan PCB input deteksi telah tersedia dalam bentuk file bitmap. Secara umum implementasi dari identifikasi kecacatan PCB dengan menggunakan metode *Template Matching*

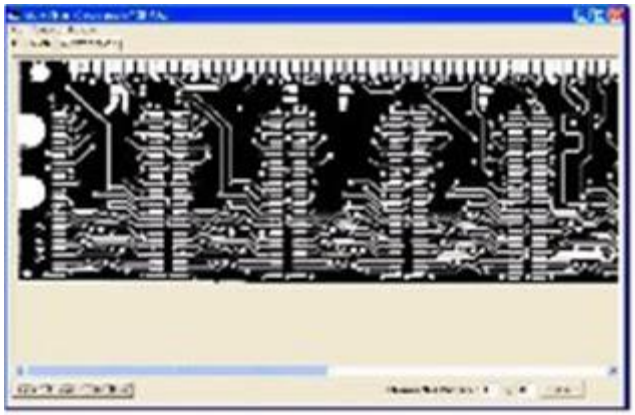
• *Grayscale* digunakan untuk merubah citra warna menjadi citra keabuan.
• *Median Filter* : Digunakan untuk melakukan proses penapisan jika citra dianggap masih mengandung derau.
• Batas Ambang : Digunakan untuk mengatur tingkat proses pengambangan pada citra dapat dilihat pada gambar 2.2.



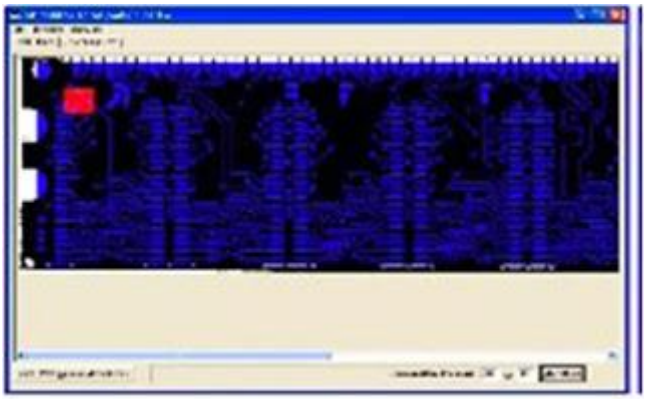
Gambar 2.2 Proses Pengambangan Pada Citra

Gambar 1 Penerapan *Template Matching* pada Identifikasi Cacat PCB
Pada penelitian sejenis, umumnya output dari penggunaan *template matching* adalah berupa prosentase kemiripan antara *image master* dengan *image input*. Untuk permasalahan identifikasi kecacatan pada PCB ini, maka outputnya adalah posisi blok pada PCB input yang tidak sesuaidengan PCB master. Posisi blok itulah yang diidentifikasi terdapat kecacatan.Pada penelitian ini dilakukan upaya untuk mendeteksi kecacatan pada PCB RAM. Dalam hal ini digunakan dua buah model PCB, yaitu PCB acuan (master) dan PCB RAM yang cacat. Aplikasi akan berusaha untuk mendeteksi kecacatan yang terjadi dalam bentuk *output* yang menunjukkan letak titik kecacatan pada PCB RAM tersebut. Masing-masing PCB yang digunakan adalah sebuah citra berkarakteristik bitmap. Sebelum digunakan sebagai acuan pada pada proses *template matching* ini,maka PCB master terlebih dahulu dibuatkan model grayscalenya. Pembentukan model *grayscale* ini dilakukan setelah sebelumnya menerapkan pemrosesan penapisan dan pengambangan citra. Gambar 2 menunjukkan pola *grayscale* dari PCB master. Hal yang serupa juga dilakukan pada model PCB yang akan diidentifikasi. Gambar 3 menunjukkan pola PCB masukan yang siap diidentifikasi. Terlihat secara sekilas antara dua pola gambar tersebut tidak nampak perbedaan. Dengan demikian apabila inspeksi kecacatan dilakukan secara manual maka tidak akan mudah terdeteksi.Setelah dua buah citra tersebut diproses dengan menggunakan aplikasi templete *matching*, maka terlihat hasil identifikasinya berupa lokasi dimana terdapat ketidak cocokan pola dan diasumsikan bahwa pada lokasi tersebut terdapat kecacatan PCB. Dalam hal ini titik-titik yang dianggap cacat karena tidak sesuai dengan citra pada master akan ditandai dengan blok korelasi berwarna merah dimana ukuran blok korelasi tersebut telah ditentukan sebelumnya oleh user. Gambar 4 menunjukkan output hasil *template matching* pada PCB input. Pada Gambar 4 tersebut terlihat adanya blok korelasi pada titik yang dianggap cacat karena memiliki kesalahan berupa putusnya jalur sirkuit PCB.

Gambar 2.3 Pola Grayscale PCB Master



Gambar 2.4 Pola PCB Yang siap diidentifikasi



Gambar 2.5 Hasil Identifikasi Kecacatan Pada PCB

Dalam penelitian ini PCB master dan PCB input diasumsikan telah tersedia dalam bentuk file bitmap. Proses penting untuk mendapatkan kedua jenis file tersebut adalah tahap pemindaian / *scanning*. Bila proses ini tidak dilakukan dengan teliti akan berakibat pada proses deteksi kecacatan yang tidak akurat.

2.6.1 Pendekatan Berbasis Fitur

Sebuah pendekatan berbasis fitur dapat dianggap; pendekatan dapat membuktikan lebih berguna, jika *template* gambar memiliki fitur yang kuat jika pencocokan di pencarian gambar bisa diubah dengan cara tertentu. Karena pendekatan ini tidak mempertimbangkan keseluruhan dari *template* gambar, komputasi dapat lebih efisien ketika bekerja dengan sumber gambar beresolusi lebih besar, sebagai pendekatan alternatif, berbasis *template*, mungkin memerlukan pencarian titik – titik yang berpotensi untuk menentukan lokasi pencocokan yang terbaik.

2.6.2 Pendekatan Berbasis *Template*

Untuk *template* tanpa fitur yang kuat, atau ketika sebagian besar *template* gambar merupakan gambar yang cocok, sebuah pendekatan berbasis *template* mungkin efektif. Seperti disebutkan di atas, karena berbasis *template*, *template matching* berpotensi memerlukan sampling dari sejumlah besar poin, untuk mengurangi jumlah sampling poin dengan mengurangi resolusi pencarian dan *template* gambar oleh faktor yang sama dan melakukan operasi pada perampingan gambar yang dihasilkan (multiresolusi, atau piramida, pengolahan citra), menyediakan pencarian titik data dalam pencarian gambar sehingga *template* tidak harus mempunyai pencarian titik data, atau kombinasi keduanya.

2.7 Teori *Euclidean Distance*

Euclidean distance adalah perhitungan jarak dari 2 buah titik dalam *Euclidean space.* Euclidean space diperkenalkan oleh *Euclid,* seorang matematikawan dari Yunani sekitar tahun 300 B.C.E. untuk mempelajari hubungan antara sudut dan jarak. Euclidean ini berkaitan dengan *Teorema Phytagoras* dan biasanya diterapkan pada 1, 2 dan 3 dimensi. Tapi juga sederhana jika diterapkan pada dimensi yang lebih tinggi.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Rumus Euclid

Jarak antara dua titik adalah panjang jalan yang menghubungkan mereka. Di pesawat, jarak antara titik (x_1, y_1)dan (x_2, y_2) diberikan oleh *teorema Pythagoras* ,

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} .$$

Dalam *Euclidean* tiga ruang, jarak antara titik (x_1, y_1, z_1)dan (x_2, y_2, z_2) adalah

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} .$$

Secara umum, jarak antara titik $\mathbf{x}$dan $\mathbf{y}$dalam ruang Euclidean $\mathbb{R}^n$ diberikan oleh

$$d = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} .$$

Untuk permukaan melengkung atau lebih rumit, yang disebut metrik dapat digunakan untuk menghitung jarak antara dua titik dengan integrasi. Ketika wajar tanpa pengecualian, "" jarak umumnya berarti terpendek jarak antara dua titik. Sebagai contoh, ada jumlah tak terbatas jalur antara dua titik padabola tapi, secara umum, hanya jalur terpendek tunggal. Berarti terpendek jarak antara dua titik adalah panjang yang disebut geodesik antara

titik-titik. Dalam kasus bola, geodesik adalah segmen dari lingkaran besar yang berisi dua poin.

Mari $\gamma(t)$ menjadi kurva halus dalam berjenis M dari $x$ke $y$dengan $\gamma(0) = x$ dan $\gamma(1) = y$. Lalu $\gamma'(t) \in T_{\gamma(t)}$, di mana $T_x$adalah ruang tangen dari Mpada x . Panjang kurva dari γsehubungan dengan struktur Riemannian diberikan oleh

$$\int_0^1 |\gamma'(t)|_{\gamma(t)} \, dt,$$

dan jarak $d(x, y)$antara $x$dan $y$adalah jarak terpendek antara $x$dan $y$diberikan oleh

$$d(x, y) = \inf_{\gamma: x \text{ to } y} \int |\gamma'(t)|_{\gamma(t)} \, dt.$$

Dalam rangka untuk menentukan jarak relatif $n > 1$poin dalam pesawat, $1 + 2(n-2) = 2n - 3$koordinat diperlukan, sejak pertama selalu dapat diambil sebagai (0, 0) dan yang kedua sebagai $(x, 0)$, yang mendefinisikan *x* sumbu . Sisa $n - 2$poin memerlukan dua koordinat masing-masing.Namun, jumlah total jarak adalah

$$\binom{n}{2} = \tfrac{1}{2} n(n-1),$$

di mana $\binom{n}{k}$adalah koefisien binomial . Jarak antara $n > 1$titik karenanya tunduk pada $m$hubungan, di mana

$$\begin{aligned} m &\equiv \tfrac{1}{2} n(n-1) - (2n-3) \\ &= \tfrac{1}{2}(n-2)(n-3). \end{aligned}$$

Untuk $n = 1$, 2, ..., ini memberikan 0, 0, 0, 1, 3, 6, 10, 15, 21, 28, ... (OEIS A000217) hubungan, dan jumlah hubungan antara $n$titik adalah jumlah segitiga T_{n-3} .

Meskipun tidak ada hubungan untuk $n = 2$dan $n = 3$poin, untuk $n = 4$ (a quadrilateral), ada satu (Weinberg 1972):

$$\begin{vmatrix} d_{12}^4 d_{34}^2 + d_{13}^4 d_{24}^2 + d_{14}^4 d_{23}^2 + d_{23}^4 d_{14}^2 + d_{24}^4 d_{13}^2 + d_{34}^4 d_{12}^2 + d_{12}^2 d_{23}^2 \\ d_{31}^2 + d_{12}^2 d_{24}^2 d_{41}^2 + d_{13}^2 d_{34}^2 d_{41}^2 + d_{23}^2 d_{34}^2 d_{42}^2 - d_{12}^2 d_{23}^2 d_{34}^2 - d_{13}^2 \\ d_{24}^2 - d_{12}^2 d_{24}^2 d_{43}^2 - d_{14}^2 d_{42}^2 d_{23}^2 - d_{13}^2 d_{34}^2 d_{42}^2 - d_{14}^2 d_{43}^2 d_{32}^2 - d_{23}^2 \\ d_{14}^2 - d_{21}^2 d_{13}^2 d_{34}^2 - d_{24}^2 d_{41}^2 d_{13}^2 - d_{21}^2 d_{14}^2 d_{43}^2 - d_{31}^2 d_{12}^2 d_{24}^2 - d_{32}^2 \end{vmatrix} :$$

Persamaan ini dapat diturunkan dengan menulis

$$d_{ij} \equiv \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

dan menghilangkan $x_i$dan $y_j$dari persamaan untuk d_{12}, d_{13}, d_{14}, d_{23}, d_{24}, dan d_{34}. Hal ini menghasilkan penentu Cayley-Menger

$$0 = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & d_{12}^2 & d_{13}^2 & d_{14}^2 \\ 1 & d_{21}^2 & 0 & d_{23}^2 & d_{24}^2 \\ 1 & d_{31}^2 & d_{32}^2 & 0 & d_{34}^2 \\ 1 & d_{41}^2 & d_{42}^2 & d_{43}^2 & 0 \end{vmatrix},$$

seperti yang diamati oleh Uspensky (1948, p. 256).

2.8 Pencocokan Berbasis *Template* dan Konvolusi

Sebuah metode dasar *template matching* menggunakan konvolusi bayangan (*template*), disesuaikan dengan fitur tertentu dari *template matching*, yang ingin kita deteksi. Teknik ini dapat dengan mudah dilakukan pada gambar abu-abu atau tepi gambar. Hasil konvolusi akan di tempat tertinggi di mana struktur gambar sesuai dengan struktur bayangan, di mana nilai-nilai gambar besar dapat dikalikan dengan nilai-nilai bayangan besar.

Metode ini biasanya diimplementasi dengan terlebih dahulu memilih sebuah bagian dari pencarian gambar untuk digunakan sebagai *template*: Kita akan memanggil pencarian gambar S (x, y), dimana (x, y) mewakili koordinat setiap

pixel dalam pencarian gambar. Kita akan memanggil *template* T (x t, y t,), dimana (x t, t y) merupakan koordinat dari setiap pixel dalam *template*. Kemudian kita hanya memindahkan pusat (atau asal) dari *template* T (x t, x t,) atas setiap titik (x, y) dalam pencarian gambar dan menghitung jumlah produk antara koefisien dalam S (x, y) dan T (x t, y t,) atas seluruh wilayah dari *template*. Karena semua kemungkinan posisi dari *template* yang berkenaan dengan pencarian gambar dianggap posisi terbaik. Metode ini kadang-kadang disebut sebagai *'Linear Spasial Filtering'* dan *template* disebut masker penyaring.

2.8.1 Mempercepat Proses

Di masa lalu, tipe spasial filtering biasanya hanya digunakan dalam solusi hardware khusus karena kompleksitas komputasi operasi, namun kita dapat mengurangi kompleksitas ini dengan penyaringan dalam domain frekuensi dari gambar itu, disebut sebagai ' *frekuensi domain filtering'*, hal ini dilakukan melalui penggunaan teorema konvolusi.

Cara lain untuk mempercepat proses pencocokan adalah melalui penggunaan dari suatu gambar piramida. Ini adalah serangkaian gambar, pada skala yang berbeda, yang terbentuk dengan berulang kali menyaring dan subsampling gambar asli agar menghasilkan gambar resolusi berkurang berurutan. Gambar resolusi lebih rendah dapat dicari untuk *template* (dengan mengurangi resolusi yang sama), untuk menghasilkan posisi semula yang memungkinkan untuk mencari pada skala yang lebih besar. Foto yang lebih besar kemudian dapat dicari dalam jendela kecil di sekitar posisi mulai menemukan lokasi *template* terbaik. Metode lain yang dapat menangani masalah seperti ini antara lain terjemahan, skala dan rotasi gambar.

2.8.2 Penjelasan Tentang *Cylinder Head*

Kepala Silinder (*Cylinder Head*) Sepeda Motor Bagian paling atas dari kontruksi mesin sepeda motor adalah kepala silinder. Kepala silinder berfungsi sebagai penutup lubang silinder pada blok silinder dan tempat dudukan busi. Kepala silinder bertumpu pada bagian atas blok silinder. Titik

tumpunya disekat dengan gasket (paking) untuk menjaga agar tidak terjadi kebocoran kompresi, disamping Blok Silinder Mesin Sepeda Motor Silinder *liner* dan blok silinder merupakan dua bagian yang melekat satu sama lain. Daya sebuah motor biasanya dinyatakan oleh besarnya isi silinder suatu motor. Silinder *liner* terpasang erat pada blok,dan bahannya tidak sama. Silinder *liner* dibuat dari bahan yang tahan terhadap gesekan dan panas, sedangkan blok dibuat dari besi Perbedaan kontruksi kepala silinder dan blok silinder dari mesin dua langkah dan empat langkah Kontruksi luar blok silinder dibuat seperti sirip, ini untuk melepaskan panas akibat kerja mesin. Dengan adanya sirip-sirip tersebut, akan terjadi pendinginan terhadap mesin karena udara bisa mengalir diantara sirip-sirip. Sirip juga memperluas bidang pendinginan, sehingga penyerapan panas lebih besar dan suhu motor tidak terlampau tinggi dan sesuai dengan temperatur kerja. Blok Silinder merupakan inti dari pada mesin yang terbuat dari besi tuang. Belakangan ini ada beberapa blok silinder yang dibuat dari paduan aluminium. Seperti kita ketahui, bahwa aluminium ringan dan dapat meradiasikan panas yang lebih efisien dibandingkan dengan besi tuang. Blok silinder dilengkapi dengan rangka pada bagian luar untuk memberikan kekuatan Konstruksi Tabung Silinder Konstruksi Tabung Silinder Secara umum terdapat tiga tipe utama konstruksi silinder. Silinder integral adalah dimana silindernya dicetak menjadi satu unit dengan blok *engine*, hal ini secara umum adalah blok engine yang terbuat dari bahan besituang kelabu dan khusus untuk silindernya ditambah dengan bahan lain agar kuat dan dapat dibentuk sesuai dengan *Gasket* Kepala Silinder (*Cylinder Head Gasket*) letaknya antara blok silinder dan kepala silinder, fungsinya untuk mencegah kebocoran gas pembakaran, air pendingin dan oli. Gasket Kepala silinder harus tahan panas dan tekanan dalam setiap perubahan temperatur. Umumnya *gasket* dibuat dari *carbon clad steel* (gabungan carbon dengan lempengan baja)

2.9 *OpenCV*

Menurut Bradski dan Kaehler (2008, p1) *OpenCV* adalah *open source Computer Vision Library.* Pustaka tersebut dikembangkan menggunakan bahasa

pemrograman C dan C++, yang berjalan pada 3 sistem operasi, yaitu : *Linux, Windows, Mac OS X*. *OpenCV* didesain untuk memberikan efisiensi komputasi dengan fokus pada *real time application. OpenCV* dikembangkan dengan optimisasi bahasa C yang memberikan kelebihan pada *multicore processors.* Satu tujuan dari *OpenCV* adalah menghasilkan infrastruktur dari *Computer Vision* yang mudah digunakan dan dapat membantu membuat aplikasi yang mutakhir secara cepat. Pustaka *OpenCV* memiliki lebih dari 500 fungsi yang dapat digunakan dalam membantu *Computer Vision*.

2.10 *Visual Studio*

Mengenal *Visual Basic.Net* Pada zaman dahulu ada sebuah bahasa pemrograman yang diberi nama *Basic* (*Beginner's All-purpose Symbolic Instruction Code*). Sesuai dengan namanya, *Basic* ditujukan sebagai bahasa yang paling sederhana bagi mereka yang tidak terlalu familiar dengan dunia pemrograman. Pada tahun 1991 Microsoft mengeluarkan *Visual Basic*, pengembangan dari *Basic* yang berubah dari sisi pembuatan antarmukanya. *Visual Basic* sampai sekarang masih menjadi salah satu bahasa pemrograman terpopuler di dunia. Pada akhir tahun 1999, teknologi.Net diumumkan *Microsoft* memposisikan teknologi tersebut sebagai platform untuk membangun XML *Web services*. XML *Web services* memungkinkan aplikasi tipe apa pun dapat berjalan pada sistem *computer* dengan tipe manapun dan dapat mengambil data yang tersimpan pada *server* dengan tipe apa pun melalui internet. *Microsoft Visual Basic*.Net adalah *Visual Basic* yang direkayasa kembali untuk digunakan pada platform.Net sehingga aplikasi yang dibuat menggunakan *Visual Basic*.Net dapat berjalan pada sistem computer apa pun, dan dapat mengambil data dari *server* dengan tipe apa pun asalkan terinstal *Net framework*.

Berikut ini perkembangan *Visual Basic*. Net :

a. *Visual Basic*. Net 2002 (VB 7.0)
b. *Visual Basic*. Net 2003 (VB 7.1)
c. *Visual Basic*. 2005 (VB 8.0)
d. *Visual Basic*. 2008 (VB 9.0)
e. *Visual Basic*. 2010 (VB 10.0)

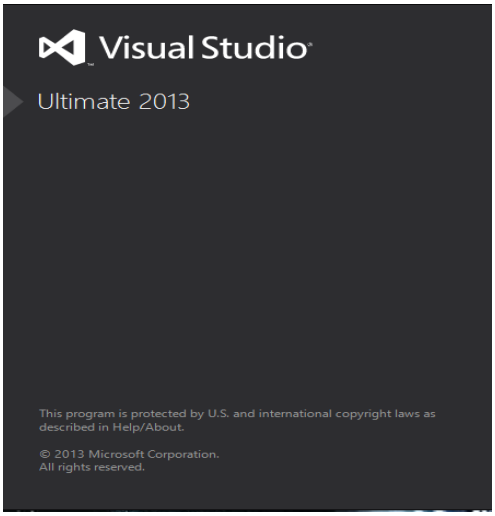
f. *Visual Basic*. 2012 (VB 11.0)

g. *Visual Basic*. 2013

Kelebihan *Visual Basic* .Net antara lain:

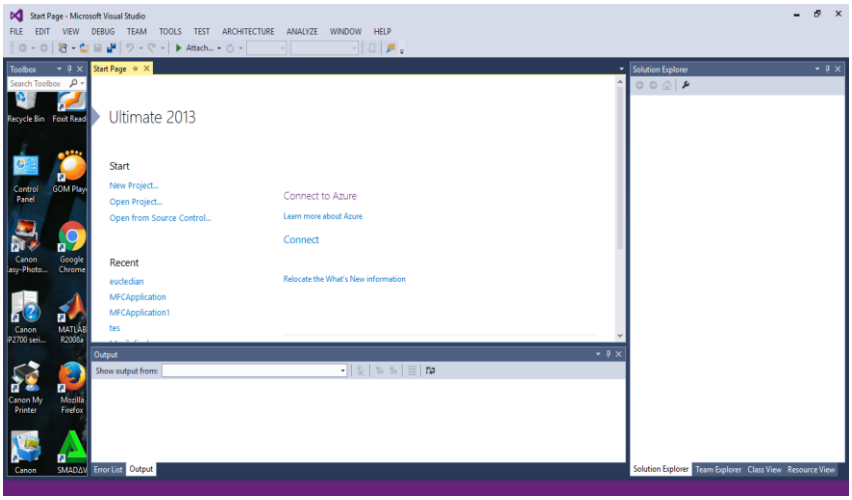
1. Sederhana dan mudah dipahami.
2. Mendukung GUI.
3. Menyederhanakan deployment.
4. Menyederhanakan pengembanga perangkat lunak.
5. Mendukung penuh OOP.
6. Mempermudah pengembangan aplikasi berbasis web.
7. Migrasi ke VB .Net dapat dilakukan dengan mudah.
8. Banyak digunakan oleh programmer-programmer di seluruh dunia.

Penjelasan tentang tampilan awal apabila program di jalankan pertama kali. Dapat dilihat di gambar 2.6 di bawah ini :



Gambar 2.6 *Visual Studio Software*

Penjelasan tentang tampilan awal untuk membuat program baru atau bisa disebut juga proses awal pembuatan program. Dapat dilihat pada gambar 2.7 berikut :



Gambar 2.7 Jendela *Dekstop Visual Studio*

2.11 *Unified Modeling Language* (*UML*)

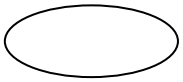
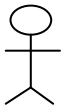
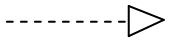
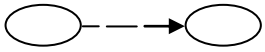
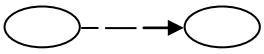
Rosa dan Salahudin (2011,p.131), menjelaskan bahwa *unified modeling language* (UML) menguraikan salah satu *standar* bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat aplikasi dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

Kesimpulan metode *UML* (*Unified Modeling Language*) adalah merupakan sebuah metode atau sebuah bahasa yang digunakan dalam menterjemahkan, menjelaskan, memodelkan, mendefinisikan suatu sistem dengan bentuk simbol-simbol tertentu yang bertujuan untuk memberikan penjelasan-penjelasan detail dari sebuah sistem.

2.11.1 *Use Case Diagram*

Rosa dan Salahudin (2011,p.131) menguraikan bahwa *use case* diagram merupakan pemodelan untuk kelakuan sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Simbol-simbol yang digunakan untuk pembuatan *use case* diagram dapat dilihat pada tabel 2.2 berikut.

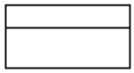
Tabel 2.2 Simbol *Use Case Diagram*

Simbol	Keterangan
Use Case 	Menggambarkan bagaimana seseorang akan menggunakan atau memanfaatkan sistem.
Aktor 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri.
Asosiasi	Komunikasi antara *use case* dan aktor yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Generalisasi 	Sebagai penghubung antara aktor-*use case* atau *use case*-*use case*.
<<Include>> 	*Include Relationship* (relasi cakupan) : Memungkinkan suatu *use case* untuk menggunakan fungsionalitas yang disediakan oleh *use case* yang lainnya.
<<Extend>> 	*Extend Relationship* : Memungkinkan relasi *use case* memiliki kemungkinan untuk memperluas fungsionalitas yang disediakan oleh *use case* yang lainnya.

2.11.2 *Class Diagram*

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Simbol-simbol yang digunakan untuk pembuatan *class diagram* dapat dilihat pada tabel 2.3 berikut:

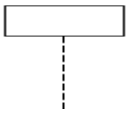
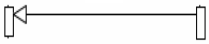
Tabel 2.3 *Simbol Class Diagram*

SIMBOL	NAMA	KETERANGAN
	Generalization	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
	*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
	*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
	Realization	Operasi yang benar-benar dilakukan oleh suatu objek.
	*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya

2.11.3 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa pesan yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Simbol-simbol yang digunakan untuk pembuatan *class diagram* dapat dilihat pada tabel 2.4 berikut :

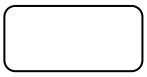
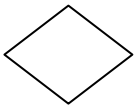
Tabel 2.4 *Simbol Sequence Diagram*

GAMBAR	NAMA	KETERANGAN
	*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
	*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi
	*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas.

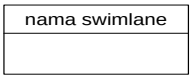
2.11.4 *Activity Diagram*

Activity Diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. (Rosa A.S dan Salahudin (2011,p.134). Simbol-simbol yang digunakan untuk pembuatan *activity diagram* dapat dilihat pada tabel 2.5 di bawah ini :

Tabel 2.5 *Simbol Activity Diagram*

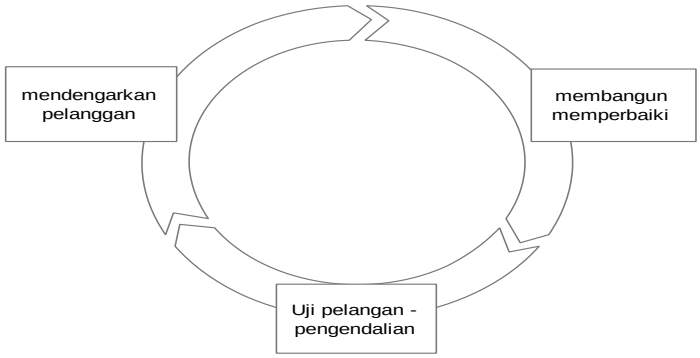
Simbol	Keterangan
Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan 	Asosiasi percabangan dimana ada pilihan aktivitas lebih dari satu.
Penggabungan	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.

Tabel Lanjutan 2.5

Simbol	Keterangan
Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas.

2.12 Metode Pengembangan Perangkat Lunak

Untuk metode pengembangan perangkat lunak, penulis menggunakan metode *prototyping*. "*Prototype*" adalah implementasi bagian dari produk *software* yang secara typical fungsinya dibatasi, realibitas rendah, tampilannya miskin, dan kurang ketegasan. (Al Bahra Bin Ladjamudin, 2006:22) *Prototype* sering dikembangkan secara cepat dalam bahasa tingkat tinggi atau bahasa *prototype* tertentu, tanpa memperhatikan kebenaran dan ketegapan dan sebagainya.



Gambar 2.8. *Prototype Diagram*

Tahapan - tahapan dalam *Prototyping* adalah sebagai berikut:

1) Pengumpulan kebutuhan : D*eveloper* dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya.

2) Perancangan : Perancangan dilakukan cepat dan rancangan mewakili semua aspek perangkat lunak yang diketahui, dan rancangan ini menjadi dasar pembuatan *prototype*.
3) Evaluasi *prototype* : Klien mengevaluasi *prototype* yang dibuat dan digunakan untuk memperjelas kebutuhan perangat lunak.

Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. *Prototype-prototype* dibuat untuk memuaskan kebutuhan klien dan untuk membangun perangkat lunak lebih cepat, namun tidak semua *prototype* bisa dimanfaatkan. Demi kebutuhan klien lebih baik p*rototype* yang dibuat diusahakan dapat dimanfaatkan.