BAB II

TINJAUAN PUSTAKA

2.1 Profil Tempat Kerja Praktek

2.1.1 Sejarah Instansi

Institut Teknologi Bandung (ITB), didirikan pada tanggal 2 Maret 1959. Kampus utama ITB saat ini merupakan lokasi dari sekolah tinggi teknik pertama di Indonesia. Walaupun masing-masing institusi pendidikan tinggi yang mengawali ITB memiliki karakteristik dan misi masing-masing, semuanya memberikan pengaruh dalam perkembangan yang menuju pada pendirian ITB. ITB mempunyai beberapa laboratorium penelitian salah satunya adalah Microsoft Innovation Center ITB (MIC ITB).

Secara organisatoris, MIC-ITB adalah lembaga yang bernaung di bawah Institut Teknologi Bandung melalui Kantor Lembaga Penelitian dan Pengabdian pada Masyarakat dan Sekolah Teknik Elektro dan Informatika ITB, dengan dukungan dari Microsoft. Seperti tersirat dari nama MIC-ITB yang berarti Microsoft Innovation Center ITB, tujuan dari berdirinya adalah untuk:

a. Membantu komunitas untuk dapat menguasai dan menggunakan teknologi Microsoft dengan berbagai skema Penguasaan Teknologi, baik untuk kebutuhan pendidikan, penelitian maupun dukungan terhadap industri.
b. Memantapkan dan mengembangkan hubungan antara Microsoft dan jejaringnya dengan komunitas pendidikan tinggi di Indonesia, khususnya di area Bandung Raya dan sekitarnya.
c. Menjadi hubungan dari berbagai stakeholder untuk dapat bekerja sama dengan mengkoordinasikan berbagai peluang dan fasilitas yang ada dalam berbagai kegiatan.

2.1.2 Logo Instansi

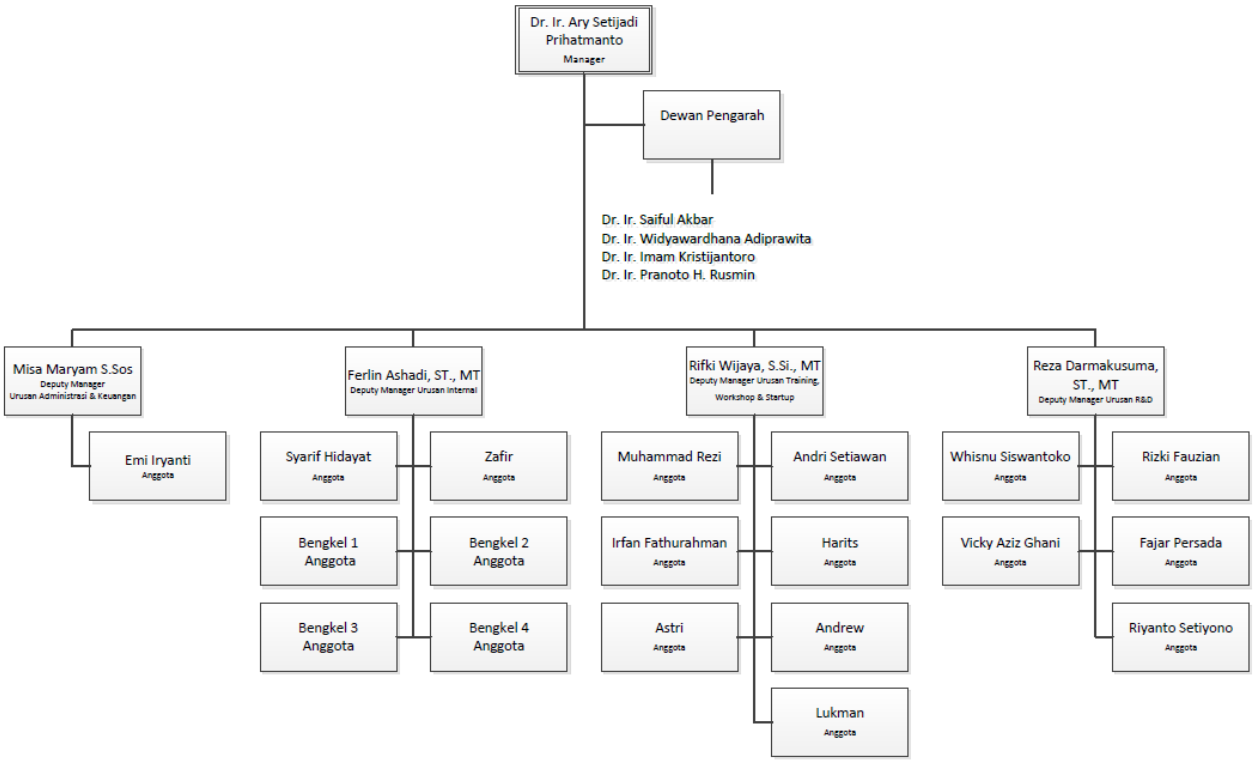
Berikut ini adalah logo dari MIC-ITB

Gambar 2. 1 Logo Instansi

2.1.3 Struktur Organisasi dan Job Description

Struktur organisasi merupakan susunan yang terdiri dari fungsi-fungsi dan hubungan-hubungan yang ada dalam suatu instansi yang menyatakan keseluruhan kegiatan untuk mencapai sasaran instansi.

2.1.3.1 Struktur Organisasi



Gambar 2. 2 Struktur Organisasi

2.1.3.2 Job Description

Adapun *job description* dari MIC – ITB itu sendiri yaitu :

Tabel 2. 1 Job Description

Jabatan	Job Description
1. Manager MIC-ITB	a. Memberikan arahan kepada seluruh personil baik dalam instruksional, administrasi serta kendali mutu. b. Membangun dan melakukan pengawasan penelitian di lingkungan MIC-ITB.
2. Dewan Pengarah	a. Memberikan arahan dalam perumusan, perencanaan, dan pelaksanaan proses penelitian di MIC-ITB.
3. Deputi Manager Urusan Administrasi dan Keuangan	a. Merencanakan dan menganalisa pembelanjaan perusahaan. b. Mengatur struktur aktiva.
4. Deputi Manager Urusan Internal	a. Mengatur semua urusan kebutuhan peralatan penelitian.
5. Deputi Manager Urusan Training	a. Mengatur perencanaan pelatihan dan *workshop* yang diselenggarakan MIC-ITB.
6. Deputi Manager Urusan R&D	a. Merencanakan pengembangan penelitian MIC-ITB.

2.2 Landasan Teori

2.2.1 Multitouch

Multitouch adalah pengembangan dari *touchpad* dan *touchscreen* yang memungkinkan pengguna menyentuh perangkat dengan beberapa jari secara bersamaan untuk melakukan interaksi dengan perangkat [2] (dalam hal ini adalah komputer). Sebenarnya *Multitouch* itu bisa dibilang memakai media input yang sama dengan *touchscreen*, yaitu layar Bedanya adalah dengan *Multi-Touch* kita bisa menggunakan beberapa jari sekaligus untuk melakukan sesuatu seperti melakukan *Zoom in/out* dan juga memutar (*rotate*) gambar. *Multitouch* telah diterapkan dengan berbagai cara berbeda, tergantung pada ukuran dan jenis antarmuka.



Gambar 2. 3 Multitouch

Layar yang sensitif terhadap sentuhan karena Sirkuit yang dibuat dapat merasakan perubahan dari setiap titik pada semua *grid* yang ada . Dengan kata lain, setiap titik pada *grid* diatur sedemikian rupa sehingga memiliki sinyal sendiri ketika disentuh dan menyambungkan sinyal tersebut ke *processor* oleh sebuah *controller*. *Controller* tersebut kemudian akan melakukan penerjemahan informasi dari sensor-sensor tersebut menjadi informasi yang dimengerti oleh prosessor komputer. Selanjutnya *processor* akan

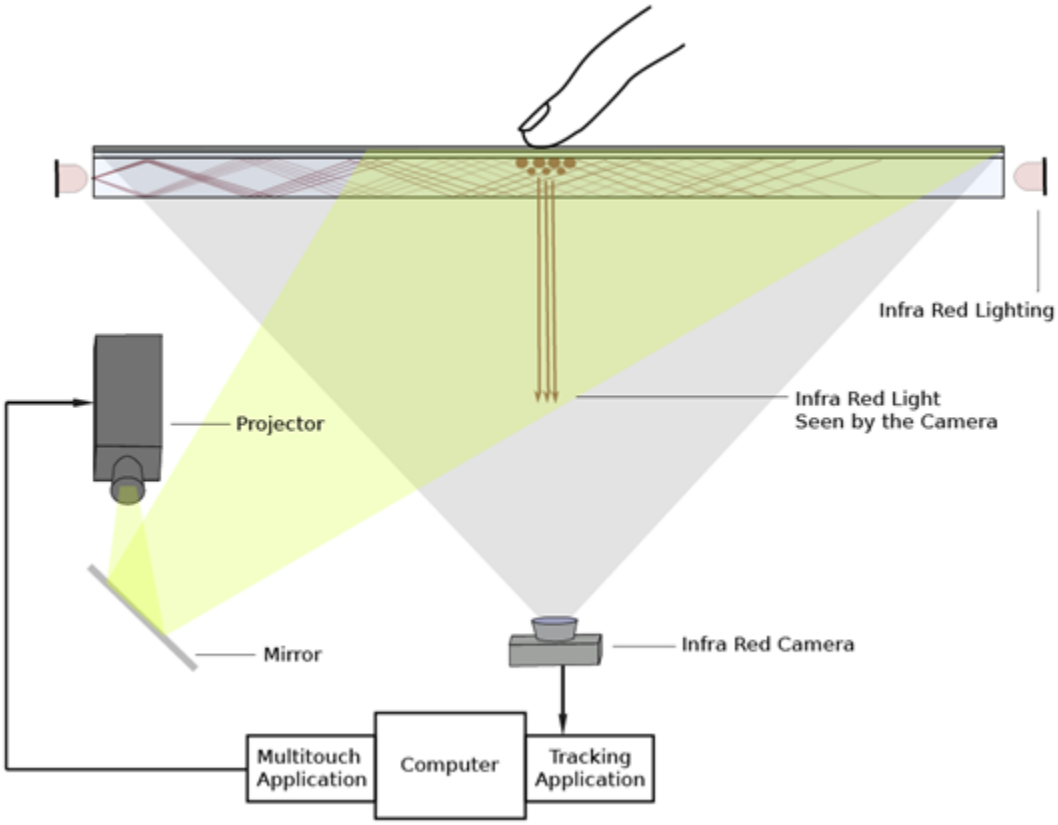
memproses inputan tersebut dengan menggunakan *software* dan menginterpretasikan data tersebut dan mengolahnya menjadi perintah.

2.2.1.1 Perangkat Keras Multitouch

Ada beberapa cara untuk membuat permukaan *multitouch*, ada kapasitif, resistif, dan teknik non tradisional lainnya. Teknik optik merupakan cara yang terbilang mudah dan paling efektif bagi kebanyakan orang untuk membuat perangkat *multitouch*. [3]

Setiap teknik menggunakan 3 komponen utama, yaitu :

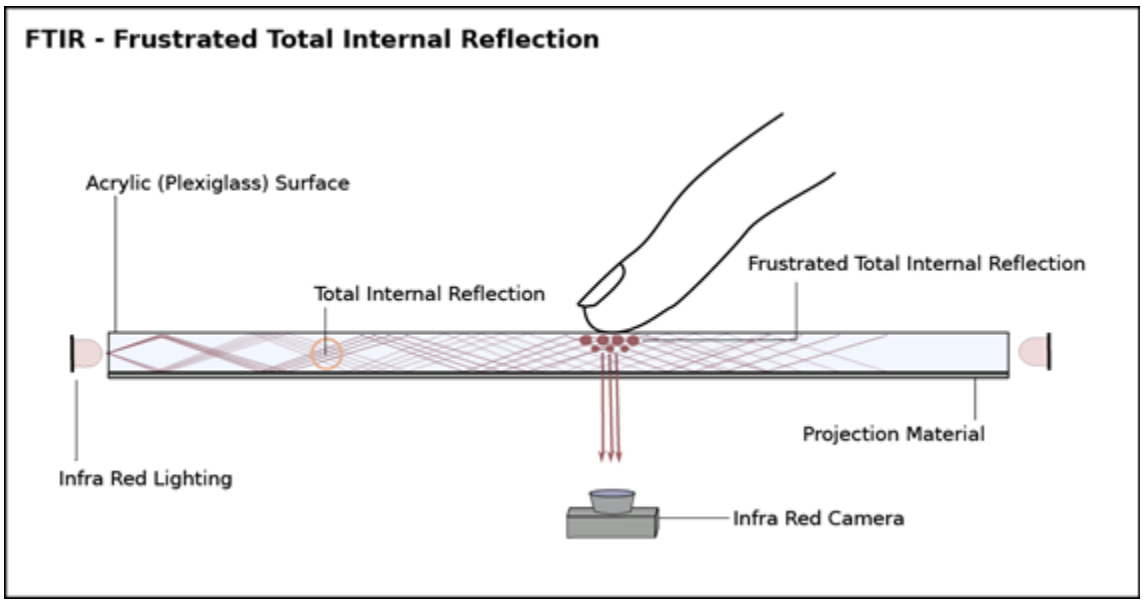
1. Kamera Inframerah (atau sensor optik lainnya)
2. Cahaya Inframerah
3. Tampilan Layar Komputer (Proyektor atau LCD)



Gambar 2. 4 Komponen Perangkat Multitouch [3]

Sebuah kamera inframerah mengarah pada permukaan sentuh dan mendeteksi ketika jari/benda menyentuh permukaannya. Cahaya inframerah digunakan untuk membedakan antara citra visual pada permukaan sentuh, kemudian jari pada permukaan akan dideteksi. Proyektor berfungsi untuk menampilkan *output* dari komputer yang diproyeksikan atau ditempatkan dibawah permukaan sentuh, merupakan hal yang penting untuk kamera agar tidak melihat gambar ini ketika pelacakan jari/benda pada layar. Untuk itu kamera dapat dimodifikasi hanya untuk melihat spektrum dari cahaya inframerah (dengan menghilangkan *filter* inframerah dan menggantinya dengan *bandpass filter*) ini akan memotong citra visual dari yang dilihat oleh kamera dan memungkinkan kamera hanya untuk melihat cahaya inframerah yang menerangi jari/benda pada permukaan sentuh. Kamera terhubung ke komputer, dan aplikasi pelacakan (CCV) menggunakan kamera untuk melacak dan menciptakan koordinat sentuhan. [3]
Teknik-teknik optik yang paling populer adalah :

a. FTIR (Frustrated Total Internal Reflection)

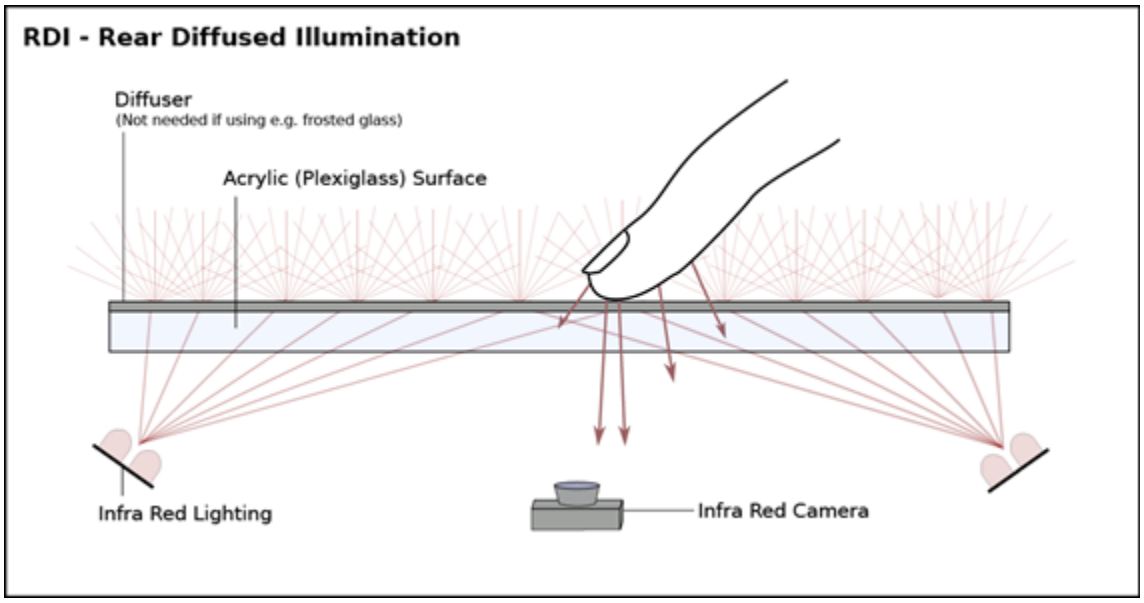


Gambar 2. 5 FTIR [3]

Dalam FTIR, cahaya inframerah ditempatkan dan diarahkan ke tepi (sisi) dari sebuah panel akrilik. Cahaya terperangkap di dalam akrilik oleh fenomena yang disebut "refleksi internal total". Ketika jari menyentuh permukaan akrilik, cahaya

inframerah "frustrasi" menyebabkan cahaya menyebar ke bawah di mana hal ini terlihat oleh kamera inframerah (*webcam* yang dimodifikasi). [3]

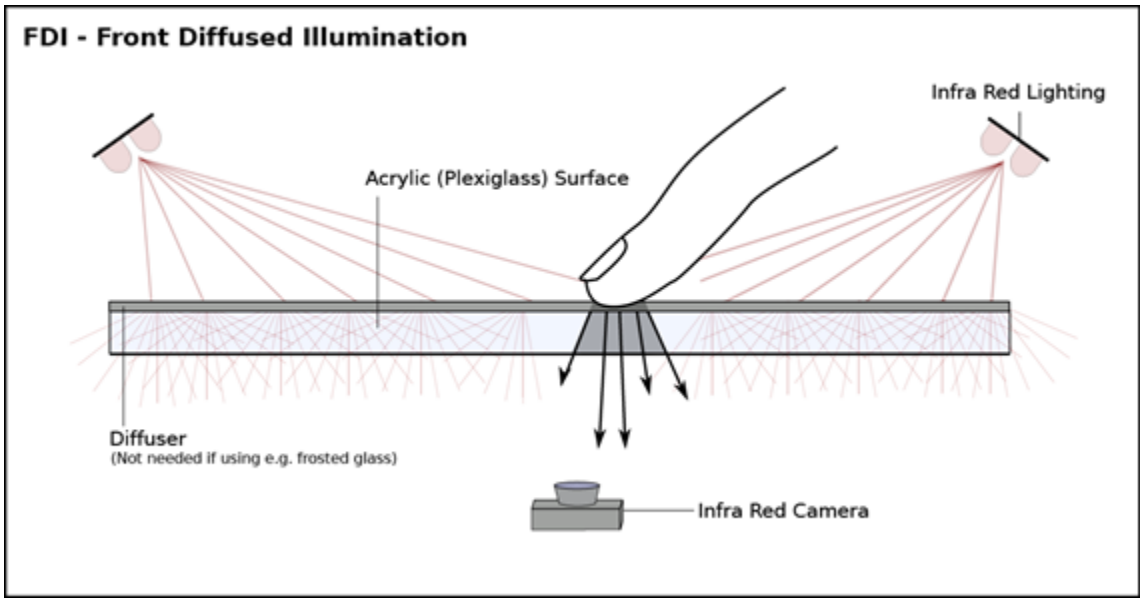
b. RDI (Rear Diffused Illumination)



Gambar 2. 6 RDI [3]

Dalam Rear DI, cahaya inframerah bersinar pada layar dari bawah permukaan sentuh. *Diffuser* A ditempatkan di atas atau di bawah permukaan sentuh. Ketika jari atau objek menyentuh permukaan, cahaya inframerah menyentuh objek dan direfleksikan ke bawah yang terlihat oleh kamera inframerah di bawah permukaan. Tergantung pada *diffuser*, metode ini juga dapat mendeteksi benda melayang, tangan, dan jari-jari di atas permukaan. [3]

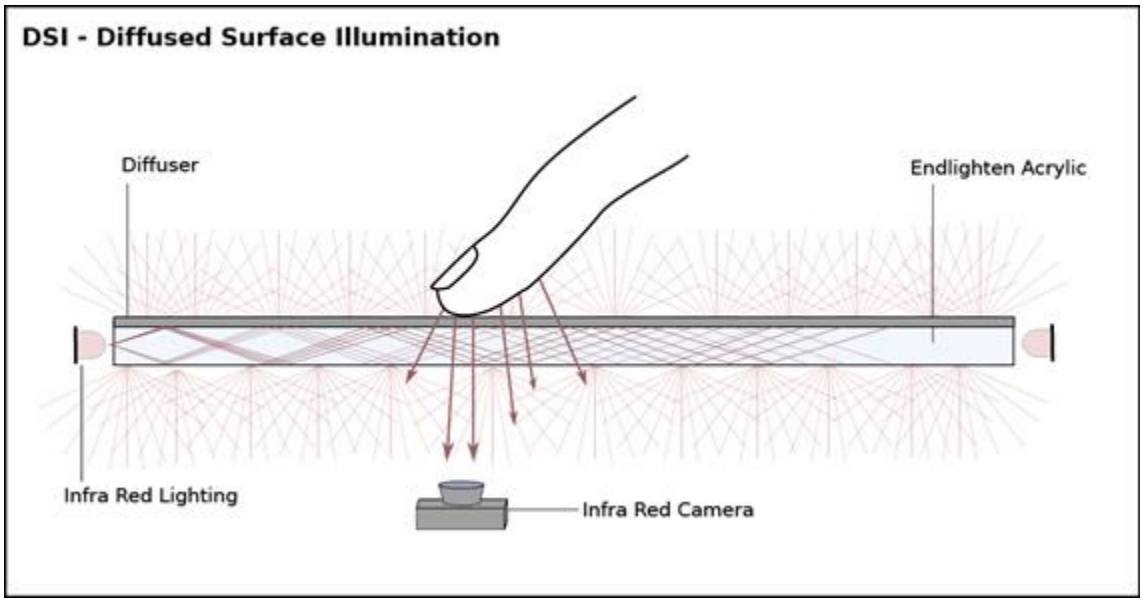
c. FDI (Front Diffused Illumination)



Gambar 2. 7 FDI [3]

Dalam Front DI, cahaya inframerah bersinar pada layar dari atas permukaan sentuh. *Diffuser* A ditempatkan di atas atau di bawah permukaan sentuh. Ketika jari atau objek menyentuh permukaan, bayangan yang dibuat di bawah objek akan dilihat oleh kamera inframerah di bawah permukaan. Metode ini adalah apa yang digunakan oleh MTmini, tapi tanpa memerlukan cahaya inframerah (karena tidak ada tampilan visual yang dihapus). [3]

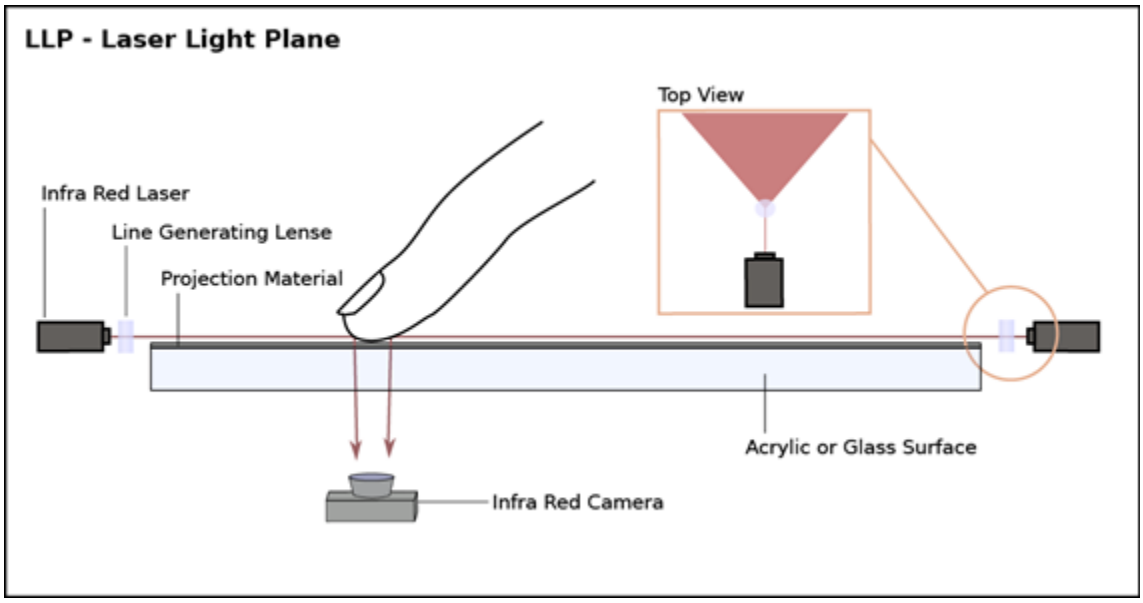
d. DSI (Diffused Surface Illumination)



Gambar 2. 8 DSI [3]

DSI mirip dengan FTIR dimana cahaya inframerah ditempatkan di tepi panel akrilik yang diarahkan kedalam. Sebuah akrilik khusus dengan partikel kecil di dalamnya, bertindak seperti ribuan cermin kecil, digunakan untuk mendistribusikan cahaya inframerah secara merata. Ketika jari atau benda menyentuh *diffuser*, cahaya inframerah memantul dari dalam dan akan terlihat oleh kamera dibawah permukaan. Efeknya mirip dengan iluminasi melebur. [3]

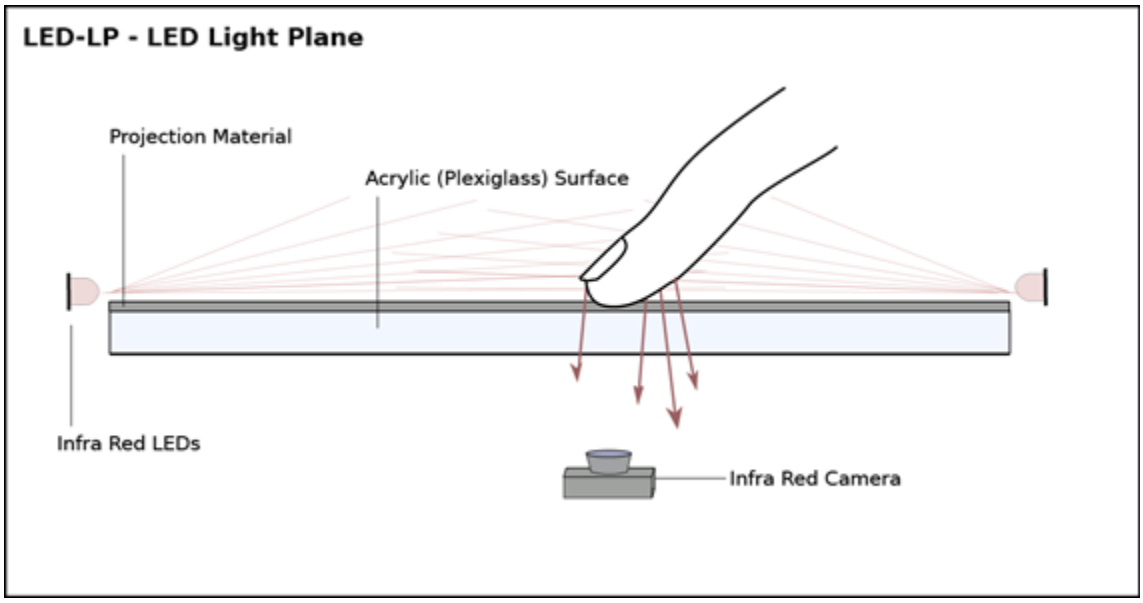
e. LLP (Laser Light Plane)



Gambar 2. 9 LLP [3]

Dalam LLP, cahaya inframerah dari laser tunggal atau ganda bersinar tepat di atas permukaan. Cahaya laser memiliki ketebalan sekitar 1mm dan diposisikan dekat dengan permukaan sentuh. Ketika jari atau benda menyentuh permukaan, sinar objek akan terlihat oleh kamera inframerah di bawah permukaan. [3]

f. LED-LP (LED Light Plane)



Gambar 2. 10 LED-LP [3]

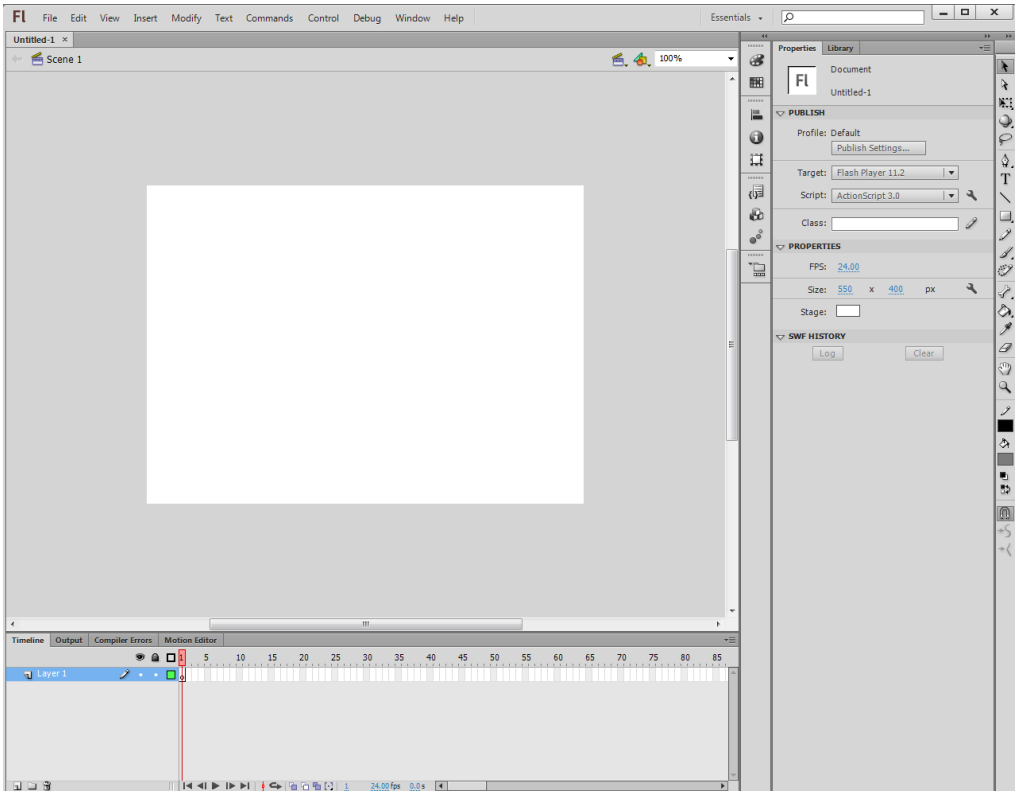
LED-LP mirip dengan LLP dalam aliran cahaya diciptakan di atas permukaan. Bukan menggunakan laser seperti LLP, melainkan menggunakan LED. LED diposisikan tepat di atas permukaan sentuhan untuk menciptakan bidang cahaya. Ketika jari atau benda menyentuh permukaan, cahaya terang akan dilihat oleh kamera inframerah di bawah permukaan. [3]

2.2.2 Software Pendukung Pembangun Perangkat Lunak (Tools)

2.2.2.1 Adobe Flash

Adobe Flash (dahulu bernama Macromedia Flash) adalah salah satu perangkat lunak komputer yang merupakan produk unggulan Adobe Systems. Adobe Flash digunakan untuk membuat gambar vektor maupun animasi gambar tersebut. Berkas yang dihasilkan dari perangkat lunak ini mempunyai file *extension* .swf dan dapat diputar di penjelajah *web* yang telah dipasangi Adobe Flash Player. Flash menggunakan bahasa pemrograman bernama ActionScript yang muncul pertama kalinya pada Flash 5.

Sebelum tahun 2005, Flash dirilis oleh Macromedia. Flash 1.0 diluncurkan pada tahun 1996 setelah Macromedia membeli program animasi vektor bernama FutureSplash. Versi terakhir yang diluncurkan di pasaran dengan menggunakan nama 'Macromedia' adalah Macromedia Flash 8. Pada tanggal 3 Desember 2005 Adobe Systems mengakuisisi Macromedia dan seluruh produknya, sehingga nama Macromedia Flash berubah menjadi Adobe Flash.



Gambar 2. 11 Interface Adobe Flash CS6

Adobe Flash merupakan sebuah program yang didesain khusus oleh Adobe dan program aplikasi standar *authoring tool professional* yang digunakan untuk membuat animasi dan *bitmap* yang sangat menarik untuk keperluan pembangunan situs *web* yang interaktif dan dinamis. Flash didesain dengan kemampuan untuk membuat animasi 2 dimensi yang handal dan ringan sehingga flash banyak digunakan untuk membangun dan memberikan efek animasi pada *website*, CD Interaktif dan yang lainnya. Selain itu

aplikasi ini juga dapat digunakan untuk membuat animasi logo, *movie*, *game*, pembuatan navigasi pada situs *web*, tombol animasi, *banner*, menu interaktif, interaktif form isian, e-card, *screen saver* dan pembuatan aplikasi-aplikasi web lainnya. Dalam Flash, terdapat teknik-teknik membuat animasi, fasilitas *action script*, *filter*, custom *easing* dan dapat memasukkan video lengkap dengan fasilitas *playback* FLV. Keunggulan yang dimiliki oleh Flash ini adalah ia mampu diberikan sedikit *code* pemograman baik yang berjalan sendiri untuk mengatur animasi yang ada didalamnya atau digunakan untuk berkomunikasi dengan program lain seperti HTML, PHP, dan Database dengan pendekatan XML, dapat dikolaborasikan dengan *web*, karena mempunyai keunggulan antara lain kecil dalam ukuran file outputnya.

Movie-movie Flash memiliki ukuran file yang kecil dan dapat ditampilkan dengan ukuran layar yang dapat disesuaikan dengan keingginan. Aplikasi Flash merupakan sebuah standar aplikasi industri perancangan animasi *web* dengan peningkatan pengaturan dan perluasan kemampuan integrasi yang lebih baik. Banyak fiture-fiture baru dalam Flash yang dapat meningkatkan kreativitas dalam pembuatan isi media yang kaya dengan memanfaatkan kemampuan aplikasi tersebut secara maksimal. Fiture-fiture baru ini membantu kita lebih memusatkan perhatian pada desain yang dibuat secara cepat, bukannya memusatkan pada cara kerja dan penggunaan aplikasi tersebut. Flash juga dapat digunakan untuk mengembangkan secara cepat aplikasi-aplikasi web yang kaya dengan pembuatan *script* tingkat lanjut. Di dalam aplikasinya juga tersedia sebuah alat untuk men-debug *script*. Dengan menggunakan *Code hint* untuk mempermudah dan mempercepat pembuatan dan pengembangan isi ActionScript secara otomatis. Untuk memahami keamanan Adobe Flash dapat dilihat dari beberapa sudut pandang, berdasarkan beberapa sumber referensi bahwa tidak ada perbedaan menyolok antara HTML dan JavaScript dimana didalamnya terdapat banyak *tools* yang dapat diambil dari SWF termasuk ActionScript. Sehingga kode data dapat terjamin keamanannya. Oleh sebab itu, semua kebutuhan data yang terdapat dalam SWF dapat diambil kembali melalui *server*. Keuntungan menggunakan metode yang sama dengan menggunakan aplikasi *web* yang standar adalah akan menjamin dan mengamankanpenyimpanan dan perpindahan data.

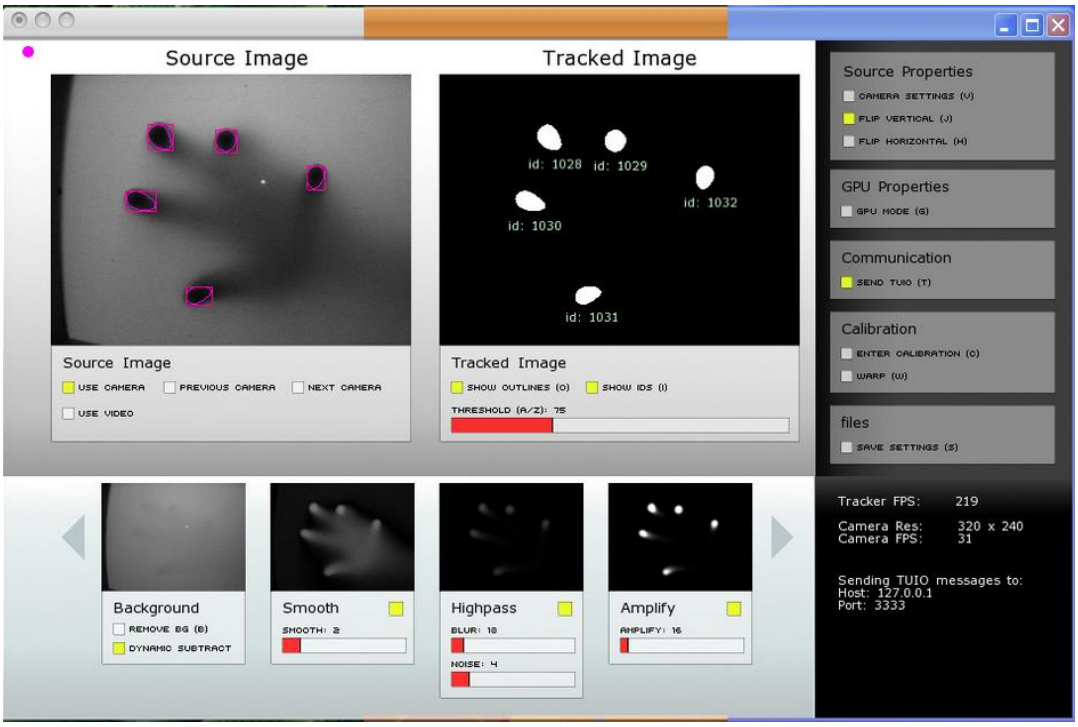
2.2.2.2 Action Script

ActionScript adalah bahasa pemrograman yang dibuat berdasarkan ECMAScript, yang digunakan dalam pengembangan situs *web* dan perangkat lunak menggunakan platform Adobe Flash Player. ActionScript juga dipakai pada beberapa aplikasi basis data, seperti Alpha Five. Bahasa ini awalnya dikembangkan oleh Macromedia, tapi kini sudah dimiliki dan dilanjutkan perkembangannya oleh Adobe, yang membeli Macromedia pada tahun 2005.

Action Script terbaru saat ini adalah Action Script 3.0. Action Script 3.0 adalah bahasa terbaru dari edisi yang sebelumnya dikenal dengan Action Script 2.0. Action Script 3.0 memiliki beberapa kelebihan dibanding pendahulunya, antara lain fitur yang ditawarkan adalah file pada Action Script 3.0 dapat dibuat terpisah saat *runtime*.

2.2.2.3 CCV

Community Core Vision adalah *library open source* untuk pengolahan citra dan pengenalan pola *input video* atau *output tracking data* (contohnya koordinat dan ukuran sentuhan) dan kondisi (contoh nya jari menekan, bergerak, dan mengangkat) yang dibutuhkan untuk membangun sebuah aplikasi *multitouch*. [4]



Gambar 2. 12 Interface CCV [4]

2.2.2.4 TUIO

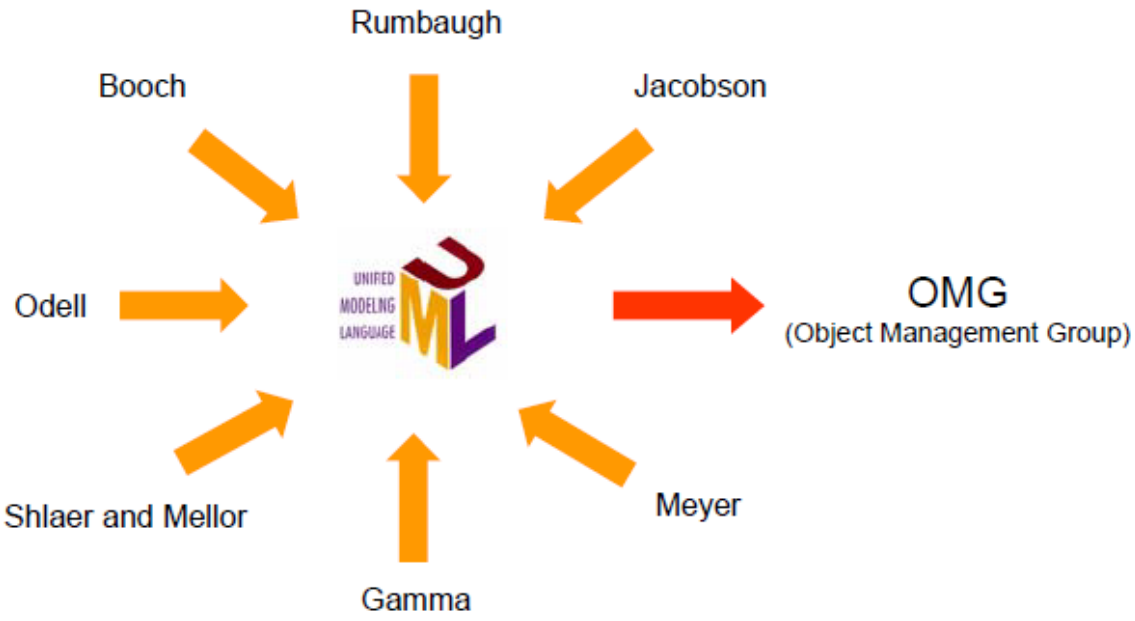
TUIO merupakan program *opensource* yang mendefinisikan protokol umum dan API untuk permukaan *multitouch*. Protokol TUIO memungkinkan transmisi deskripsi abstrak dari permukaan interaktif, termasuk peristiwa sentuhan dan objek nyata. Protokol ini mengkodekan data kontrol dari aplikasi *tracker* (misalnya didasarkan pada visi komputer) dan mengirimkannya ke aplikasi *client*. [5]

2.2.3 UML (Unified Modeling Language)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. [6] UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru

apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan.



Gambar 2. 13 UML

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML, sebenarnya konsepsi dasar UML dapat dirangkum seperti gambar 2.14.

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Gambar 2. 14 Konsep dasar UML

Abstraksi konsep dasar UML yang terdiri dari *structural classification, dynamic behavior,* dan model *management,* dapat dipahami dengan mudah apabila melihat gambar dari diagram diatas. *Main concepts* dapat dipandang sebagai term yang akan muncul pada saat membuat diagram, sedangkan *view* adalah kategori dari diagram tersebut.

Berikut merupakan beberapa diagram yang ada pada UML :

a. *Use Case* Diagram
b. *Class* Diagram
c. *Statechart* Diagram
d. *Activity* Diagram
e. *Sequence* Diagram

2.2.3.1 Use Case Diagram

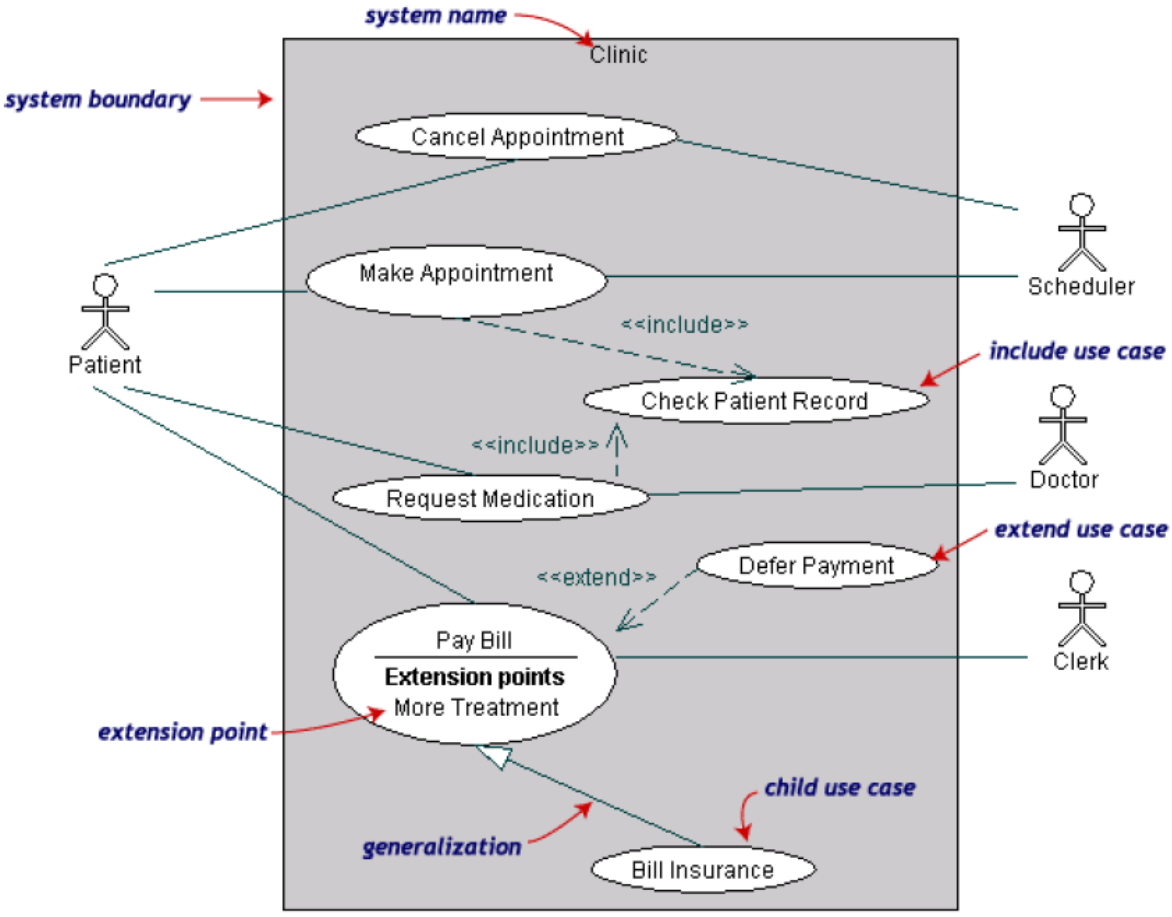
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-include dieksekusi secara normal.

Sebuah *use case* dapat di-include oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Contoh *use case* diagram :



Gambar 2. 15 Contoh Use Case Diagram

2.2.3.2 Class Diagram

Class 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).

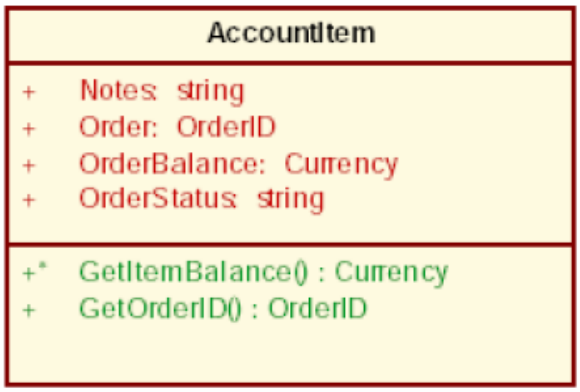
Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

1. Nama (dan *stereotype*)

2. Atribut
3. Metoda

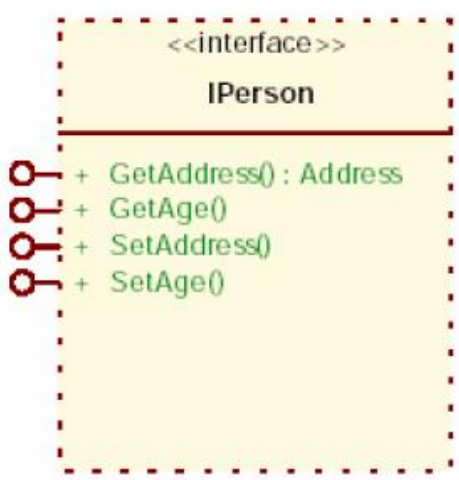
Atribut dan metoda dapat memiliki salah satu sifat berikut :

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.



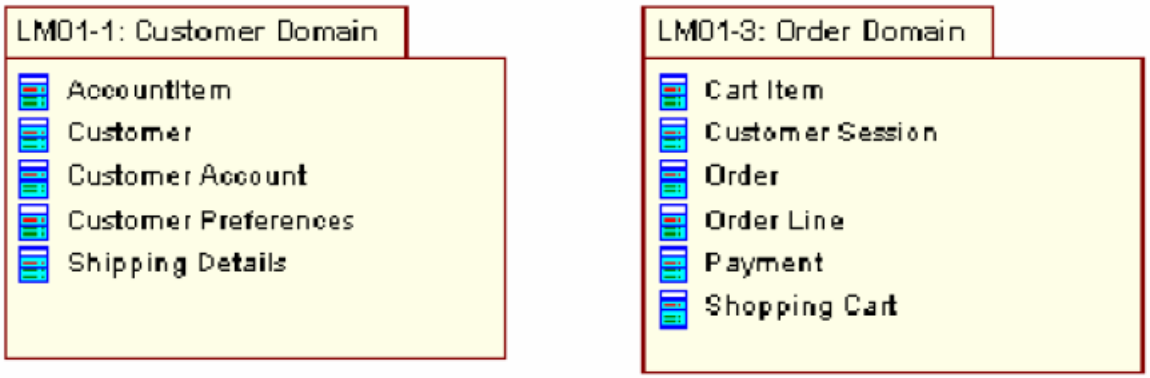
Gambar 2. 16 Contoh Class Diagram

Class dapat merupakan implementasi dari sebuah *interface*, yaitu class abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.



Gambar 2. 17 Contoh Interface

Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.



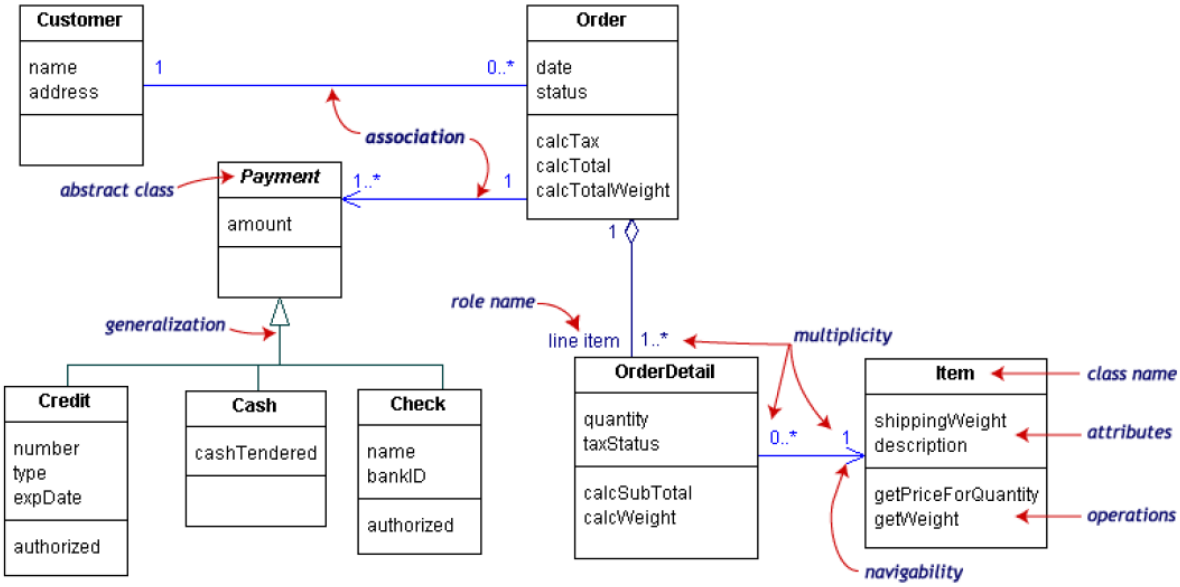
Gambar 2. 18 Contoh Package

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 metoda *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. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence* diagram yang akan dijelaskan kemudian.

Contoh Class Diagram :



Gambar 2. 19 Contoh Class Diagram Lengkap

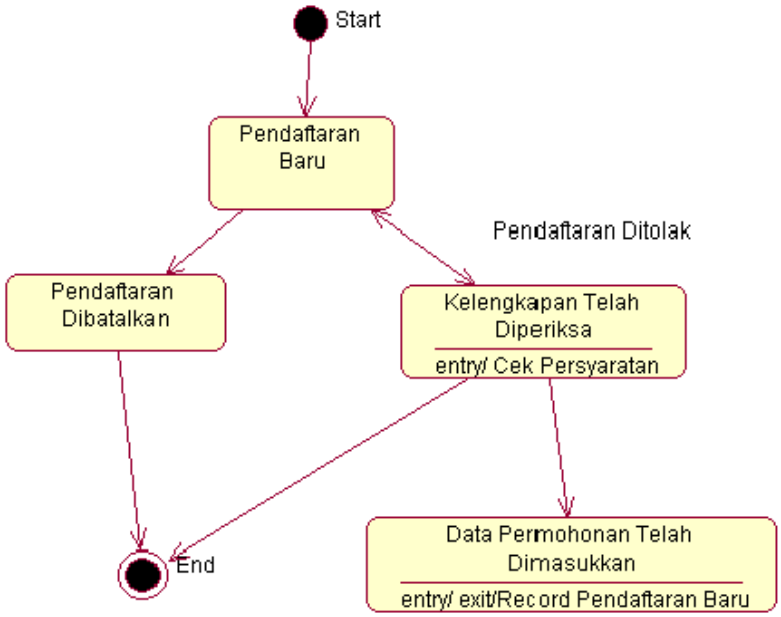
2.2.3.3 Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart* diagram).

Dalam UML, state digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi

guard yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

Contoh *statechart* diagram :



Gambar 2. 20 Contoh Statechart Diagram

2.2.3.4 Activity Diagram

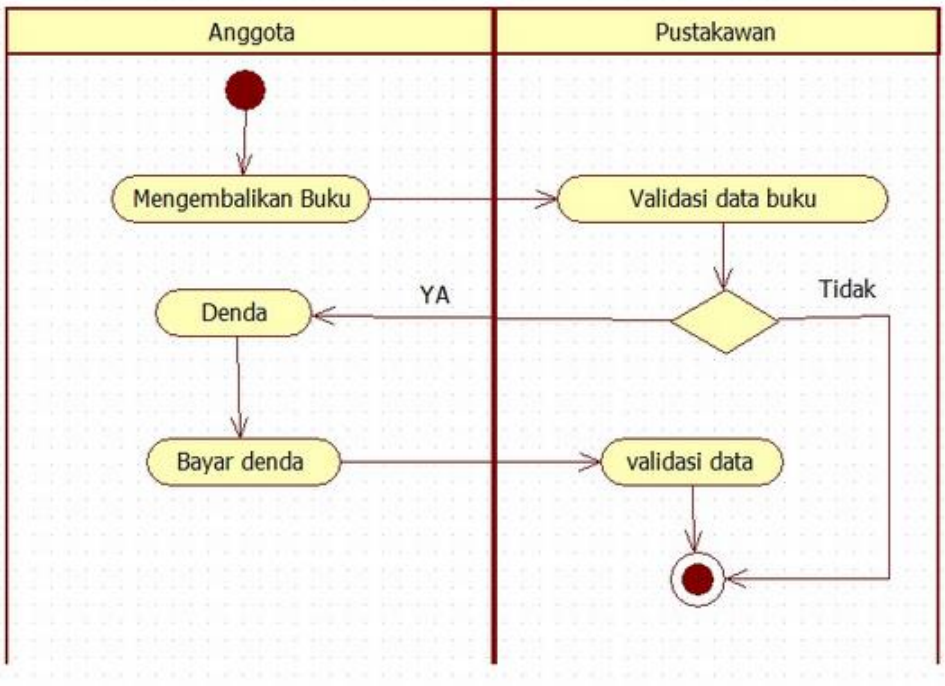
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari *level* atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity* diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

Contoh *activity* diagram :



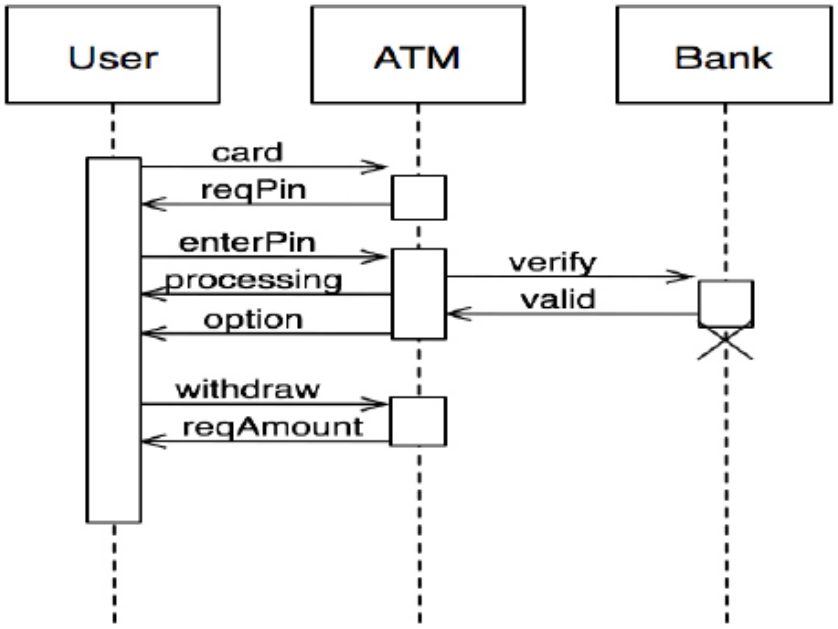
Gambar 2. 21 Contoh Activity Diagram

2.2.3.5 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal.

Message digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary, controller* dan *persistent entity*. Contoh *sequence* diagram :



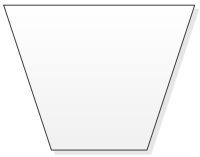
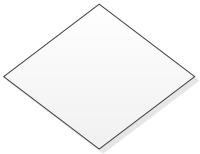
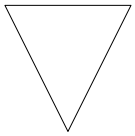
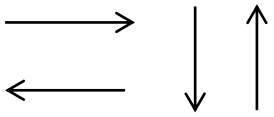
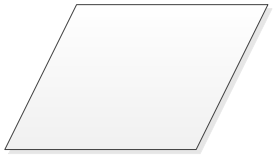
Gambar 2. 22 Contoh Sequence Diagram

2.2.4 Flow Map

Digunakan untuk mendefinisikan hubungan antara bagian (pelaku proses), proses (manual atau berbasis komputer) dan aliran data dalam bentuk dokumen keluaran dan masukan. *Flow Map* disebut juga diagram aliran dokumen atau diagram prosedur kerja

merupakan bagan alir yang menunjukkan arus dari laporan dan termasuk tembusan-tembusannya. *Flow map* menggambarkan pergerakan proses diantara unit kerja yang berbeda-beda, sekaligus menggambarkan arus dari dokumen, aliran data fisik, entitas-ntitas sistem informasi dan kegiatan operasi yang berhubungan dengan sistem informasi. Simbol-simbol pada *flowmap* diantaranya terdapat pada tabel dibawah ini :

Tabel 2. 2 Simbol Flow Map

Simbol	Nama Simbol	Fungsi
	Dokumen Manual	Menunjukan dokumen sebagai masukan dan keluaran dalam proses manual
	Proses Manual	Menunjukan proses yang dilakukan tanpa bantuan komputer
	Kondisi	Menunjukan ada suatu kondisi yang harus diperiksa untuk melihat hasil keluaran
	Arsip	Menggambarkan kumpulan dokumen sejenis yang disimpan
	Aliran Dokumen	Menunjukan aliran dokumen
	Data	Menunjukan data untuk membentuk dokumen komputerisasi

	Proses Terkomputerisasi	Menggambarkan proses yang dilakukan dengan bantuan komputer
	File/*Database*	Menggambarkan penyimpanan jika menggunakan proses terkomputerisasi

Hal yang harus diperhatikan dala *flowmap*:

a. Memodelkan aliran dokumen pada sistem yang sedang berjalan.
b. Bentuk dokumen bisa manual atau berupa file komputer.
c. Satu alur aliran dokumen terdiri dari *input* – proses – *output*.
d. Tidak boleh ada dokumen yang hilang dalam runtunan prosesnya.

Kegunaan dari *Flow Map* ini adalah:

a. Menggambarkan aktivitas apa saja yang sedang berjalan.
b. Menjabarkan aliran dokumen yang terlihat.
c. Menjelaskan hubungan-hubungan data dan informasi dengan bagian-bagian dalam aktivitas tersebut.