BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian-penelitian sebelumnya yang relevan dengan model penelitian ini cukup banyak. Model penelitian seperti ini yaitu penelitian pengujian suatu metode baru untuk membuktikan apakah metode tersebut lebih bagus dari metode yang lama sudah sering dilakukan terutama oleh kalangan pendidik. Yang berbeda dari penelitian ini adalah penelitian ini menguji software yang kami kembangkan oleh tim kami sendiri. Dari berbagai literatur yang penulis baca ada beberapa literatur yang relevan dengan penelitian kami.

(Robins dkk, 2003) Melakukan sebuah study literatur dari persepektif seorang pengajar yang berkaitan dengan psikologi dan pendidikan pemrograman. Tujuan penelitian adalah untuk mengidentifikasi permasalahan-permasalahan dalam proses belajar programer pemula. Ada beberapa hal yang membuat kami tertarik seperti yang tercantum dalam studi literaturnya yaitu penelitian dari Guindon (1990) yang menyebutkan karakter dari seorang ahli adalah memiliki skema pengetahuan efisien terorganisir dan khusus, paham dalam algoritma yang mendasar (bukan rincian dangkal seperti sintaks bahasa). Kesimpulan dari penelitian Robins dan kawan-kawannya adalah sifat yang mendasari apa yang membuat seorang pemula yang efektif? Bagaimana kita dapat mengubah efektif pemula menjadi lebih efektif?. Berbagai faktor yang berpotensi relevan yaitu motivasi, kepercayaan diri atau tanggapan emosional, aspek umum atau pengetahuan khusus, dan strategi. Jadi disamping faktor dari dalam diri programer itu sendiri

juga ada faktor dari luar yaitu strategi (strategi pengajaran) yang bisa mempercepat proses menjadikan programer pemula menjadi programer ahli.

Selanjutnya adalah Penlitian dari Lahtinen dan kawan-kawannya. (Lahtinen dkk, 2005) Melakukan survei internasional dengan responden 500 siswa dan guru. Survei ini memberikan informasi tentang kesulitan yang dirasakan ketika belajar dan mengajar pemrograman. Hasil survei juga dapat dijadikan sebagai dasar untuk rekomendasi dan *developing* bahan pembelajaran. Dalam penelitian tersebut disimpulkan bahawa kesulitan yang biasa di alami baik siswa maupun guru dalam belajar dan mengajar pemrograman tidak hanya konsep pemrograman yang abstrak yang cukup susah untuk diinterpretasikan dalam dunia nyata akan tetapi konstruksi program juga termasuk didalamnya. Penelitian tersebut dilakukan menggunakan dua bahasa pemrograman sample yaitu C++ dan Java. Jadi selain konsep pemrograman yang abstrak, susunan bahasa / kostruksi dari bahasa yang di pelajari itu juga mempengaruhi apakah bahasa itu sulit atau mudah untuk dipelajari.

Yang ketiga yaitu Penelitian yang dilakukan oleh Kelleher dan Pausch. (Kelleher dan Pausch, 2003) Dalam penelitiannya Kelleher dan Pausch menyajikan taksonomi bahasa dan lingkungan yang dirancang untuk membuat pemrograman lebih mudah diakses untuk programer pemula dari segala usia. Penelitian ini menjelaskan semua kategori dalam taksonomi tersebut, memberikan gambaran singkat tentang sistem dalam setiap kategori, dan menyarankan beberapa jalan keluar untuk masa depan dalam lingkungan pemrograman pemula dan bahasa pemrograman .Penelitian tersebut membahas satu pertanyaan pokok yaitu dalam menciptakan lingkungan pemrograman untuk pemula, salah satu pertanyaan pertama yang harus dijawab adalah mengapa siswa/mahasiswa perlu memprogram?. Ada berbagai motivasi yang mendasari mengapa siswa/mahasiswa belajar memprogram yaitu : pemrograman sebagai karir,

memprogram untuk belajar bagaimana memecahkan masalah dengan cara yang terstruktur dan logis, untuk membangun perangkat lunak untuk keperluan pribadi, untuk mengeksplorasi ide-ide dalam bidang studi lainnya, dan lain sebagainya.

Pada Tulisan Kelleher dan Pausch ini taksonomi atau urutan tertinggi pada lingkungan pemrograman/*programming environment* adalah *teaching system* (sistem pengajaran) dan *Empowering Systems* (pemberdayaan sistem) kemudian disusul dengan berbagai hirarki taksonomi dibawahnya. *Teaching system* dan *Empowering Systems* sebagai hirarki yang paling tinggi, jelas bagaimana seorang dapat memahami pemrograman jika dia tidak belajar, dan bagaimana dia belajar jika tidak diajar. Kemudian disusul hirarki seperti *Mechanics of programming* (sisi teknis dari pemrograman) dan seterusnya. Dalam tulisannya juga disebutkan berbagai bahasa pemrograman yang bisa mempermudah siapapun yang ingin belajar pemrograman dan bisa digunakan dari segala usia. Disamping itu dia juga menyebutkan berbagai kendala dan hambatan bagi siapa saja yang belajar bahasa pemrograman. Kendala tersebut ada dua yaitu kendala teknis dan kendala sosial.

Berbagai kendala teknis seperti konsep yang abstrak, syntaks yang susah untuk dihafal dan sebagainya, untuk kendala teknis ini sudah banyak dibahas oleh banyak peneliti. Beberapa penliti mengajukan konsep dan design seperti pemrograman berbasis visual, game, dan sebagainya. Begitu juga dengan penelitian Kelleher dan Pausch ini mereka memberikan banyak contoh-contoh bahasa pemrograman visual yang dapat mempermudah pemula dalam memahami bahasa pemrograman. Kendala yang kedua yaitu kendala sosial. Analisis Kelleher dan Pausch menyebutkan bahawa ternyata tidak hanya kendala teknis yang dihadapi oleh programer pemula yang ingin belajar pemrograman tapi juga ada kendala sosial. Contoh kendala sosial adalah siswa/mahasiswa memutuskan dan memilih untuk tidak belajar pemrograman. Kelleher dan

Pausch mengatakan bahwa kendala sosial ini lebih sulit untuk diselesaikan daripada kendala teknis. Solusi dari kendala sosial ini adalah sosialisai dan dukungan sosial. Kelleher dan Pausch menyimpulkan beberapa solusi yang bisa digunakan untuk mengatasi kendala-kendala baik itu teknis maupun sosial dalam belajar pemrograman yaitu: menyederhanakan mekanisme pemrograman, memberikan dukungan bagi pelajar, dan memotivasi siswa untuk belajar memprogram. Dalam hal mengatasi hambatan sosial bisa dengan mendukung peserta didik atau memberikan alasan yang menarik untuk belajar pemrograman.

Dari penelitian Kelleher dan Pausch ini respon dari penulis sesuai dengan kondisi penelitian penulis adalah bahwa kendala-kendala itu memang benar-benar nyata. Hal tersebut bisa penulis utarakan berdasarkan pengalaman penulis sendiri baik ketika penulis belajar pemrograman maupun pengalaman penulis ketika mengajar pemrograman. Kendala teknis adalah kendala yang akan pertama dihadapi oleh siapapun yang ingin belajar pemrograman, oleh karena itu penulis mengusulkan sebuah konsep dan design yang sebenarnya sudah banyak diusulkan dan diimpelementasikan sejak dahulu akan tetapi belum di implementasikan dalam linkungan penulis yaitu VPL(Visual Programming Languages).

Penelitian yang relevan ke empat yaitu paper dari scratch . (Maloney dkk, 2008) Paper Maloney dan kawan-kawannya ini menjelaskan scratch yaitu bahasa pemrograman visual blok yang dirancang dan difasilitasi berbagai media untuk programer pemula. Pada penelitian ini Maloney dan kawan-kawannya membentuk sebuah Clubhouse yang di peruntukkan bagi siapa saja untuk datang ke Clubhouse. Penelitian tersebut dilakukan selama periode 18 bulan, setelah periode 18 bulan kemudian Maloney menganalisis 34 % atau sebanyak 536 proyek-proyek scratch yang ada di Clubhouse tadi. Dalam paper tersebut juga dibahas mengenai motivasi pemuda perkotaan yang memilih untuk memprogram menggunakan scratch daripada

menggunakan salah satu dari banyak paket perangkat lunak lain yang tersedia bagi mereka. Ada yang menurut penlulis cukup menarik disini. Pada tulisan Maloney dkk di jelaskan sebagai berikut: Pertanyaan yang lebih mendesak adalah mengapa pemuda memilih untuk terlibat dalam memprogram menggunakan scratch di Scratch Clubhouse mengingat bahwa mereka memiliki banyak pilihan perangkat lunak lain? Jawaban terbaik mungkin sudah dijawab oleh Kelleher dan Pausch yang mencatat bahawa sistem dapat membuat pemrograman menjadi lebih mudah diakses oleh pemula dan diikarenakan scratch dapat menyederhanakan mekanisme pemrograman, dengan memberikan dukungan bagi pelajar, dan dengan memberikan siswa dengan motivasi untuk belajar memprogram.

Maloney menyatakan kami berpikir bahwa scratch mempunyai tiga hal yang menarik. Yang pertama, desain blok scratch menyederhanakan mekanisme pemrograman dengan meminimalisir kesalahan sintaks, scratch langsung dapat memberikan umpan balik mengenai penempatan blok , dan scratch langsung dapat memberikan umpan balik untuk percobaan/ketika program dijalankan. Selain itu, kita berpikir bahwa infrastruktur sosial Komputer Clubhouse penting dalam memberikan dukungan bagi programer pemula. Jawaban tersebut jelas masuk akal karena di scratch Clubhouse kami juga menyediakan beberapa mentor yang bisa mengajari para programer pemula, dan juga mungkin dengan seringnya mereka berada di scratch Clubhouse mereka bisa bertukar program dan ide dengan teman-teman mereka.

Dari beberapa penelitian yang relevan dengan penelitian ini yang sudah penulis sampaikan diatas yang paling mendekati dengan penelitian yang penluis lakukan ini adalah penelitian ke empat yang dilakukan oleh Maloney dan kawan-kawannya. Tentu sudah banyak yang mengetahui tentang scratch dan begitu mudahnya memprogram dengan menggunakan scratch. Penelitian yang penulis lakukan jelas masih jauh dari penelitian ini, kalau penelitian

yang dilakukan Maloney dan kawan-kawannya memang akan memberikan hasil yang general karena sampel yang digunakan berupa proyek-proyek general dari berbagai anak/siswa yang sering *nongkrong/*bermain di scratch ClubHouse. Proyek scatch memang tidak main-main scratch berani membuat sebuah ClubHouse demi untuk kelangsungan dan analisis penelitian proyek scratch itu sendiri dan disini inilah yang belum bisa penulis lakukan. Semoga pada penelitian IndoBlockly berikutnya akan lebih baik.

2.2 Landasan Teori

Landasan teori menjelaskan berbagai teori yang berhubungan dengan penelitian yang dilakukan oleh penulis baik yang berbubungan dengan pendidikan dan pembelajaran, kemudian pemrograman dan juga software yang digunakan sebagai objek penelitian.

2.2.1 Pengertian Belajar

Belajar adalah suatu aktivitas mental/psikis yang berlangsung dalam interaksi aktif dengan lingkungan yang menghasilkan perubahan dalam pengetahuan, pemahaman, keterampilan dan nilai sikap (Winkel, 1983). Sedangkan menurut Gagne dalam Dahar (1989) belajar adalah suatu proses di mana suatu organisme berubah perilakunya sebagai akibat dari pengalaman. Dengan belajar tindakan perilaku siswa akan berubah ke arah yang lebih baik. Berhasil baik atau tidaknya belajar tergantung dari faktor-faktor yang mempengaruhinya. Faktor-faktor tersebut terdiri dari faktor internal, eksternal dan pendekatan belajar.

1. Faktor internal adalah faktor dari dalam diri siswa, yaitu keadaan/kondisi jasmani dan rohani siswa meliputi aspek fisiologis (kondisi tubuh dan panca indera), dan aspek

psikologis antara lain: intelegensi dalam, sikap misalnya dalam beradaptasi dengan teman, bakat dalam mengerjakan soal, minat dalam mengikuti pelajaran serta punya kemauan besar untuk belajar dan mempunyai motivasi untuk belajar baik individu maupun dalam kelompok.

2. Faktor eksternal adalah faktor dari luar diri siswa, yaitu kondisi lingkungan di sekitar siswa meliputi faktor lingkungan sosial (guru, teman, masyarakat, dan keluarga) dan faktor lingkungan non-sosial (gedung, sekolah, tempat tinggal, alat belajar, cuaca dan waktu belajar) .

2.2.2 Media Pembelajaran

Kata media berasal dari bahasa latin Medius yang secara harfiah berarti *'perantara'* atau *'pengantar'*. Menurut Boove (dalam Ena, 2007), media adalah sebuah alat yang berfungsi untuk menyampaikan pesan pembelajaran. Pembelajaran adalah sebuah komunikasi antara pembelajar, pengajar dan bahan ajar. Komunikasi tidak akan berjalan tanpa bantuan sarana penyampai pesan atau media. Bentuk-bentuk stimulus yang bisa dipergunakan sebagai media diantaranya adalah hubungan atau interaksi manusia, realita, gambar bergerak atau tidak, tulisan dan suara yang direkam (Ena, 2007).

Menurut Gerlach dan Ely (dalam Arsyad, 2002:12-14), ciri-ciri media ada tiga, yaitu.

1. Ciri Fiksatif (Fixative Property)

   Ciri ini menggambarkan kemampuan media merekam, menyimpan, melestarikan, dan merekonstruksi suatu peristiwa atau objek.

2. Ciri Manipulatif (Manipulatif Property)

Transformasi suatu kejadian atau objek dimungkinkan karena media memiliki ciri manipulatif. Kejadian yang memakan waktu berhari-hari dapat disajikan kepada siswa dalam waktu dua atau tiga menit dengan teknik pengambilan gambar time-lapse recording.

3. Ciri Distributif (Distributive Property)

   Ciri distributif dari media memungkinkan suatu objek atau kejadian ditransportasikan melalui ruang, dan secara bersamaan kejadian tersebut disajikan kepada sejumlah besar siswa dengan stimulus pengalaman yang relatif sama mengenai kejadian itu.

Fungsi utama media pembelajaran adalah sebagai alat bantu mengajar yang ikut mempengaruhi iklim, kondisi, dan lingkungan belajar yang ditata dan diciptakan oleh guru. Sedangkan manfaat penggunaan media (Arsyad, 2002:25-27), antara lain:

1. Media pembelajaran dapat memperjelas penyajian pesan dan informasi sehingga dapat memperlancar dan meningkatkan proses dan hasil belajar.
2. Media pembelajaran dapat meningkatkan dan mengarahkan perhatian anak sehingga dapat menimbulkan motivasi belajar, interaksi yang lebih langsung antara siswa dan lingkungannya, dan kemungkinan siswa untuk belajar sendiri-sendiri sesuai dengan kemampuan dan minatnya.
3. Media pembelajaran dapat mengatasi keterbatasan indera, ruang, dan waktu.
4. Media pembelajaran dapat memberikan kesamaan pengalaman kepada siswa tentang peristiwa-peristiwa di lingkungan mereka, serta memungkinkan terjadinya interaksi langsung.

2.2.3 Proses Pembelajaran

Penilaian adalah upaya atau tindakan untuk mengetahui sejauh mana tujuan yang telah ditetapkan itu tercapai atau tidak. Penilaian berfungsi sebagai alat untuk mengtahui keberhasilan proses dan hasil belajar siswa/mahasiswa. Dalam sistem pendidikan nasional rumusan tujuan pendidikan, baik tujuan kurikuler maupun tujuan instruksional, menggunakan klasifikasi hasil belajar dari Benyamin Bloom (Bloom, 1956) yang secara garis besar membaginya menjadi tiga ranah, yakni ranah kognitif, ranah afektif, dan ranah psikomotorik.

Salah satu prinsip dasar yang harus senantiasa diperhatikan dalam pendidikan adalah melaksanakan evaluasi hasil belajar secara menyeluruh terhadap peserta didik, baik dari segi pemahamannya terhadap materi atau bahan ajar yang telah diberikan (aspek kognitif), maupun dari segi penghayatan (aspek afektif), dan pengamalannya (aspek psikomotor). Benjamin S. Bloom dan kawan-kawannya itu berpendapat bahwa pengelompokkan tujuan pendidikan itu harus senantiasa mengacu kepada tiga jenis domain (ranah) yang melekat pada diri peserta didik, yaitu:

a) Ranah proses berfikir (cognitive domain)

b) Ranah nilai atau sikap (affective domain)

c) Ranah keterampilan (psychomotor domain)

Dalam konteks evaluasi hasil belajar, maka ketiga domain atau ranah itulah yang harus dijadikan sasaran dalam setiap kegiatan evaluasi hasil belajar. Sasaran kegiatan evaluasi hasil belajar adalah:

1. Apakah peserta didik sudah dapat memahami semua bahan atau materi pelajaran yang telah diberikan pada mereka?

2. Apakah peserta didik sudah dapat menghayatinya?
3. Apakah materi pelajaran yang telah diberikan itu sudah dapat diamalkan secara kongkret dalam praktek atau dalam kehidupannya sehari-hari?

Diantara ketiga ranah tersebut, ranah kognitiflah yang paling banyak dinilai oleh para dosen /guru karena berkaitan dengan kemampuan para siswa dalam menguasai isi bahan pengajaran.

2.2.3.1 Pengertian Ranah Penilaian Kognitif

Ranah kognitif adalah ranah yang mencakup kegiatan mental (otak). Menurut Bloom, segala upaya yang menyangkut aktivitas otak adalah termasuk dalam ranah kognitif. Ranah kognitif berhubungan dengan kemampuan berfikir, termasuk didalamnya kemampuan menghafal, memahami, mengaplikasi, menganalisis, mensintesis, dan kemampuan mengevaluasi. Dalam ranah kognitif itu terdapat enam aspek atau jenjang proses berfikir, mulai dari jenjang terendah sampai dengan jenjang yang paling tinggi. Keenam jenjang atau aspek yang dimaksud adalah:

a. Pengetahuan/hafalan/ingatan *(knowledge)*

Adalah kemampuan seseorang untuk mengingat-ingat kembali (recall) atau mengenali kembali tentang nama, istilah, ide, rumus-rumus, dan sebagainya, tanpa mengharapkan kemampuan untuk menggunkannya. Pengetahuan atau ingatan adalah merupakan proses berfikir yang paling rendah. Salah satu contoh hasil belajar kognitif pada jenjang pengetahuan adalah dapat menghafal jenis-jenis variable yang biasa digunakan dalam memprogram dsb.

b. Pemahaman *(comprehension)*

Adalah kemampuan seseorang untuk mengerti atau memahami sesuatu setelah sesuatu itu diketahui dan diingat. Dengan kata lain, memahami adalah mengetahui tentang sesuatu dan dapat melihatnya dari berbagai segi. Seseorang peserta didik dikatakan memahami sesuatu apabila ia dapat memberikan penjelasan atau memberi uraian yang lebih rinci tentang hal itu dengan menggunakan kata-katanya sendiri. Pemahaman merupakan jenjang kemampuan berfikir yang setingkat lebih tinggi dari ingatan atau hafalan. Salah satu contoh hasil belajar ranah kognitif pada jenjang pemahaman ini adalah mahasiswa bisa paham terhadap algoritma searching kemudian dapat menguraikanya kembali dengan bahasa mereka sendiri.

c. Penerapan *(application)*

Adalah kesanggupan seseorang untuk menerapkan atau menggunakan ide-ide umum, tata cara ataupun metode-metode, prinsip-prinsip, rumus-rumus, teori-teori dan sebagainya, dalam situasi yang baru dan kongkret. Penerapan ini adalah merupakan proses berfikir setingkat lebih tinggi ketimbang pemahaman. Salah satu contoh hasil belajar kognitif jenjang penerapan adalah mahasiswa mampu menerapakan algoritma yang sudah ia pahami ke dalam bentuk code dengan bahasa pemrograman yang ia kuasai.

d. Analisis *(analysis)*

Adalah kemampuan seseorang untuk merinci atau menguraikan suatu bahan atau keadaan menurut bagian-bagian yang lebih kecil dan mampu memahami hubungan di antara bagian-bagian atau faktor-faktor yang satu dengan faktor-faktor lainnya. Jenjang analisis adalah setingkat lebih tinggi ketimbang jenjang aplikasi. Contoh dari hasil belajar kognitif jenjang analisis adalah mahasiswa mampu menganalisis algoritma dan *source code,* misalnya terjadi

error ketika di jalankan maka mahasiswa dalam tahap analisis ini harus mampu menyelesaikan permasalahan tersebut *(debugging)* sampai program bisa benar-benar berjalan dengan baik.

e. Sintesis *(syntesis)*

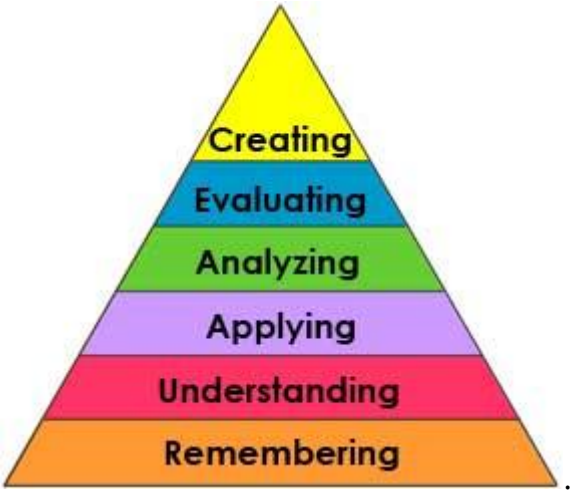
Adalah kemampuan berfikir yang merupakan kebalikan dari proses berfikir analisis. Sisntesis merupakan suatu proses yang memadukan bagian-bagian atau unsur-unsur secara logis, sehingga menjelma menjadi suatu pola yang yang berstruktur atau bebrbentuk pola baru. Jenjang sintesis kedudukannya setingkat lebih tinggi daripada jenjang analisis. Contoh: mahasiswa tidak hanya mampu memahami algoritma kemudian menuliskan dalam *source code* dan juga melakukan *debugging* tetapi mahasiswa juga mampu untuk memodifikasi atau membuat algoritma baru.

f. Penilaian *(evaluation)*

Adalah merupakan jenjang berpikir paling tinggi dalam ranah kognitif dalam taksonomi Bloom. Penilian/evaluasi disini merupakan kemampuan seseorang untuk membuat pertimbangan terhadap suatu kondisi, nilai atau ide, misalkan jika seseorang dihadapkan pada beberapa pilihan maka ia akan mampu memilih satu pilihan yang terbaik sesuai dengan patokan-patokan atau kriteria yang ada.

Salah satu contoh hasil belajar kognitif jenjang evaluasi adalah: mahasiswa mampu untuk menilai dan memilih algoritma yang paling efektif yang akan dia gunakan.

Keenam jenjang berpikir yang terdapat pada ranah kognitif menurut Taksonomi Bloom itu, jika diurutkan secara hirarki piramidal adalah sebagai tertulis pada Gambar 2.1



Gambar 2.1 Hirarki Piramida Taksonomi Bloom

2.2.3.2 Ranah Afektif dan Psikomotorik

Pemrograman komputer memang lebih banyak berkaitan dengan ranah kognitif akan tetapi bukan berarti kedua ranah atau domain yang lain itu dihilangkan. Ranah afektif adalah ranah penghayatan kemudian disusul dengna ranah psikomotorik atau ranah skill. Proses belajar pemrograman itu semua berada pada ranah kognitif dimulai dari menghafal sintaks, memahami, menganalisi kesalahan, mensintesis algoritma baru, menilai dan memilih algoritma yang paling efektif dan sebagainya.

Ranah afektif akan ada ketika mahasiswa yang tadinya dia tidak mengetahui apa-apa mengenai pemrograman, kemudian dia sudah melalui ranah kognitif dan akhirnya jiwa programer secara tidak sadar akan terbentuk di dalam diri mahasiswa tersebut. Ketika penghayatan sebuah algoritma dan juga penghayatan dalam menulis kode dalam proses selanjutnya ini akan menuji ranah psikomotorik yaitu ranah skill. Ranah skill adalah ranah dimana yang tadinya hanya seorang mahasiswa atau programer pemula akan mempunyai skill yang cukup hebat yang menjadikan dirinya menjadi programer ahli.

2.2.4 Belajar Pemrograman *(Learning Programming)*

(Robins dkk, 2003) Pemrograman adalah keterampilan yang sangat berguna dan dapat menjadi karir. Baru-baru ini kebutuhan akan permintaan terhadap programer dan minat siswa dalam pemrograman berkembang pesat, dan kursus pemrograman telah menjadi semakin populer. Belajar pemrograman membutuhkan usaha yang keras bagi seorang programer pemula biasanya mengalami berbagai kesulitan. Kursus/Sekolah Pemrograman umumnya dianggap sulit, dan sering memiliki risiko tingkat putus sekolah paling tinggi.(Winslow, 1996).Dalam penelitian disebutkan bahawa dibutuhkan sekitar 10 tahun untuk mengubah programer pemula menjadi programer ahli.

Pemahaman dalam memprogram merupakan bagian penting dari keterampilan pemrograman komputer, baik dari yang praktis,perspektif maupun teoritis. Ini adalah sebuah keterampilan kognitif yang kompleks, yang melibatkan akuisisi representasi mental dari struktur program dan fungsi. Dari sudut pandang Theoretical , pemahaman melibatkan penugasan makna tertentu yaitu sesuatu yang membutuhkan pengetahuan khusus. Dari sudut pandang praktis, kemampuan untuk memahami program yang ditulis oleh orang lain atau ketika membuat program sendiri adalah komponen penting keahlian seorang programer. Skill/keahlian pemahaman ini akan digunakan programer untuk melakukan tugas-tugas pemrograman seperti *debugging*, *editing* dan, *code reuse*. (Navarro-Prieto, 2000).

Pennington (1987a). Memperkirakan bahwa lebih dari 50% dari semua waktu programer profesional dihabiskan untuk tugas-tugas pemeliharaan program yang melibatkan modifikasi/editing dan update dari program yang sebelumnya ditulis. Dari berbagai hal diatas dapat kita simpulkan bahwa pemahaman memainkan peran sentral dalam pemrograman. Oleh

karena itu, penelitian tentang strategi pemahaman akan sangat berguna karena hasilnya dapat memberikan informasi yang berguna untuk meningkatkan kinerja programer, kemajuan pendidikan, kemajian teknologi desain dan lingkungan permrograman (programming environments).

Dari berbagai referensi diatas kita dapat menyimpulakan bahawa belajar pemrograman itu memang susah terutama bagi programer pemula/ mahasiswa yang sedang belajar pemrograman. Sperti penelitian yang dilakukan oleh Wislow bahwa bagai programer pemula butuh 10 tahun untuk menjadi programer yang ahli. Untuk menjadi programer yang baik, mahasiswa membutuhkan skill dan usaha yang keras untuk menghafalkan sintak dan memahami aturan belajar pemrograman itu sendiri (Esteves dan Mendes, 2004). Mahasiswa harus tahu dan paham bagaimana menyelesaikan suatu masalah dan belajar membuat dan merinci algoritma yang efisien itulah yang menyebabkan belajar pemrograman terkesan sulit. Pada Kenyataanya banyak mahasiswa yang gagal dan menemui masalah ini pada awal pelajaran/mata kuliah pemrograman untuk pertama kalinya. Sudah banyak peneliti yang meneliti masalah ini (Lahtinen dkk, 2005). Salah satu penyebab yang disebutkan adalah mahasiswa tidak siap untuk berfikir dan menyelesaikan masalah. Pada proses belajar mengajar sebelumnya mahasiswa tidak terbiasa untuk menyelesaikan suatu masalah/kasus, sehingga mahasiswa tidak mempunyai kemampuan problem solving yang baik (Gomes, 2006). Beberapa mahasiswa yang cukup mampu meyelesaikan masalah dan mebuat algoritma biasanya juga terkendala dalam masalah waktu penyelesaian.

Penyebab lain dari kegagalan belajar pemrograman adalah mahasiswa sudah mengatakan bahawa dia tidak menyukai pemrogramman dikarenakan konsepnya susah untuk dipahami, seperti variabel, tipe data, alamat memori dikarenakan itu adalah konsep yang abstrak yang tidak

bisa direpresentasikan di dunia nyata (Miliszewska, 2007). Penyebab lain adalah proses pembelajaran yang cenderung masih tradisional yaitu dosen hanya mengajarkan sintak bahasa pemrograman tertentu dan tidak matang dalam hal pemahaman konsep (Lethbridge, 2007). Dengan kemajuanya teknologi dan perkembangan software yang begitu pesat beberapa peneliti mengusulkan design dan sofware pemrograman berbasis visual dan memang kenyataanya sekarang banyak software-software yang diciptakan dengan tujuan untuk mengatasi masalah-masalah diatas. Memang untuk saat ini masih dominan sofware pemrograman yang berupa text based, tetapi ada juga yang visual based bahkan ada yang berbentuk game (Hundhausen, 2007).

2.2.5 Bahasa Pemrograman Visual (*Visual Programming Language* /VPLs)

(Boshernitsan dan Downes, 2004)Dari lukisan di gua hieroglif sejak dulu manusia sudah lama berkomunikasi satu dengan lainya menggunakan gambar. Banyak para peneliti di bidang VPL bertanya: Mengepa kemudian kita saat ini berkomunikasi dengan komputer menggunakan pemrograman tekstual? Bukankah akan lebih produktif jika dapat menginstruksikan komputer hanya menggunakan gambar.? Jelas para pendukung VPLs menjawab ya. Pertanyaan-pertanyaan diatas menjadi motivasi utama para peneliti di bidang VPLs. Pertama. banyak orang berpikir dan mengingat hal hal dari segi gambar. Selain itu pemrograman tekstual telah terbukti menjadi masalah bagi orang-orang kreatif dan cerdas , mereka harus banyak belajar terlebih dahulu mengenai sintak-sintak tekstualnya dari bahasa pemrograman itu sendiri. Permasalahan yang lain adalah ide lebih mudah diterjemahkan dalam bentuk gambar visual atau grafik akan tetapi kita harus menerjemahkan ide tersebut kedalam bentuk tekstual.Selain itu banyak bidang-bidang ilmu yang memerlukan simulasi dalam bentuk visual dan tidak bisa di interpretasikan dalam bentuk

tekstual.Dari berbagai alasan diatas itu membuktikan bahawa penelitian mengenai VPLs itu menarik dan penting.

2.2.5.1 Kelebihan dan Kekurangan VPLs

Pemrograman grafis/visual memiliki kelebihan dan kekurangan dibandingkan dengan pemrograman tekstual. Landasan teroi mengenai bahasa pemrograman visual ini kami mengambil dari tulisan Andrew Begel.

2.2.5.1.1 Kelebihan VPLs

(Begel, 1996) Menggunakan representasi grafis dari benda-benda, Anda bisa lebih konkret merepresentasikan suatu objek (klik dua kali pada objek koper dan melihat apa yang ada di dalam), menghilangkan sintak-sintak yang susah untuk dipahami (seperti {} dan () dalam C, BEGIN dan END dan ()dalam Pascal, dan () dalam Lisp).

Pemrograman grafis/visual juga dapat dikatakan sebagai metafora dari kehidupan nyata untuk membuat pemrograman lebih mudah.Sebagai contoh, pemrograman tombol lampu untuk menghidupkan dan mematikan lampu. Pemrograman grafis juga memungkinkan lebih mudah untuk berbagi program. Anda dapat mendefinisikan program Anda menjadi blok tertentu dan hanya "memberikan" blok ke teman untuk mencoba sendiri. Hal ini mirip dengan manfaat yang diusulkan desain OOP dalam bahasa tekstual. Keuntungan lain adalah mudah dipahami. Melihat gambar dari sebuah program, pengguna dapat lebih mudah membedakan maknanya, daripada melihat program tekstual yang terdiri dari file dan kode yang sangat banyak. Mungkin salah satu keuntungan terbaik adalah penggunaan isyarat visual dalam bahasa grafis/visual.Sambungan/koneksi antar objek dapat dibuat lebih eksplisit melalui desain dan grafis daripada menggunakan tekstual.

2.2.5.1.2 Kekurangan VPLs

(Begel, 1996) Di sisi lain, ada juga kerugian. Beberapa bahasa grafis/visual yang dapat menyebabkan frustrasi bagi programer expert yang ingin mengungkapkan pernyataan yang mungkin lebih baik dan efektif diwakili menggunakan teks. Selain itu Masalah pemrograman visual adalah bahwa Anda tidak bisa memiliki lebih dari 50 primitif visual pada layar pada saat yang sama. "Pertanyaan selanjutnya adalah Bagaimana Anda akan menulis sebuah sistem operasi dengan pemrograman visual? "Masalah lain adalah extendibility bahasa. C dan Lisp dibangun untuk dikembangkan oleh para programer. Pada saat ini, bahasa grafis cenderung terbatas pada pembuat desain tanpa berpikir untuk menambahkan fitur tambahan.

Ada daya tarik yang sangat tinggi dari Pemrograman Visual yaitu untuk anak-anak. Anak-anak suka memanipulasi blok dan mengumpulkan koleksi benda. seringkali masalah utama untuk mengajarkan pemrograman kepada anak-anak adalah bahwa sintak-sintak pemrograman yang sulit dipahami.VPLs menghilangkan masalah ini, anak-anak akan lebih mudah untuk membuat program. Akan tetapi Sebuah bahasa grafis/visual yang terlalu sederhana dapat mengurangi kreatifitas seorang anak untuk membuat program/sesuatu yang menyenangkan. Ketika anak bertambah usia dan lebih cerdas, ia dapat bermigrasi ke bahasa tekstual dan menggunakan kompleksitas yang lebih besar.

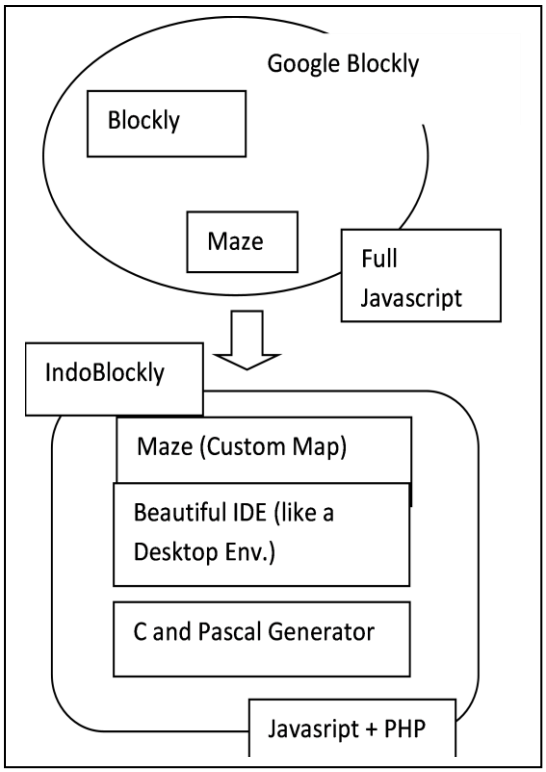
2.2.6 IndoBlockly

(Mafrur, 2012) IndoBlockly mirip dengan bahasa visual block sebelumnya seperti *scratch, Greenfoot, AppInventor*,dan *Google Blockly*. Dengan menggunakan IndoBlockly pengguna tidak akan merasa coding tetapi yang dirasakan adalah menyusun puzzle. *"Semua bisa jadi programer dengan IndoBlockly",* itu adalah slogan IndoBlockly, IndoBlockly

dilengkapi dengan generator bahasa C dan Pascal, blok-blok yang sudah disusun oleh pengguna langsung dapat diubah menjadi source code C atau Pascal.

2.2.6.1 Konsep IndoBlockly

(Mafrur, 2012) Konsep awalnya ialah kami ingin membangun sebuah software yang bisa membantu anak-anak ataupun siapa saja yang ingin belajar pemrograman menjadi lebih mudah, nyaman, dan senang dalam belajar pemrograman, secara grafis bisa dilihat pada Gambar 2.2., dengan penjelasan sebagai berikut:



Gambar 2.2 . Konsep IndoBlockly

Kriteria software sesuai dengan konsep IndoBlockly adalah :

1. Berbahasa Indonesia.
2. Menarik, tidak membosankan, user tidak terasa seperti coding tetapi seperti bermain.

3. Meminimalisir penggunakan sintak-sintak yang susah untuk dimengerti oleh pengguna baru.
4. IDE yang portabel (cloud/web based).
5. IDE yang mempunyai tampilan seperti IDE Desktop Environment.
6. Hasilnya bisa dikonvert ke dalam source code C atau Pascal dan langsung bisa di eksekusi dengan C atau Pascal compailer
7. Format file penyimpanan yang portabel.
8. Ada beberapa game logika untuk mengasah kemampuan otak anak.
9. Bisa meningkatkan pemahaman user terhadap konsep pemrograman dan algoritma.

Pada awal tahun 2012 Google meluncurkan *Google Blockly* dengan lisensi opensource. Kemudian kami bersepakat untuk mengambil source dari *Google Blockly* kemudian kami modifikasi sesuai dengan konsep awal tadi. Nama IndoBlockly juga diambil dari *Google Blockly*, Indo adalah Indonesia dan Blokly adalah blok-blok puzzle dari *Google Blockly*.