BAB II

LANDASAN TEORI

2.1 Permasalahan *Cutting Stock*

Permasalahan *Cutting Stock* merupakan suatu permasalahan yang muncul karena banyak dipakai aplikasinya dalam bidang perindustrian. Misalkan di dalam perindustrian kayu, bagaimana memanajemen pemotongan kayu supaya dapat meminimumkan sisa pemotongan yang dihasilkan dan dapat membentuk pemotongan yang optimal (Javanshir, 2007).

Dalam hal inilah permasalahan *cutting stock* dapat digunakan untuk menyelesaikan permasalahan di atas sebagai salah satu aplikasi dari permasalahan optimisasi, atau yang lebih spesifik adalah sebuah permasalahan program linier integer. Sebagai salah satu permasalahan program linier integer maka hasil yang diharapkan dalam suatu permasalahan adalah bilangan bulat.

Dikatakan sebagai permasalahan cutting stock jika terdapat permintaan ukuran dari pesanan dan adanya batasan yang ditetapkan, serta tujuannya untuk meminimumkan sisa (Fox, 1981). Adapun permasalahan cutting stock satu dimensi adalah terbatas hanya membahas satu kendala yang sesuai dengan kendala yang ditetapkan dan mengabaikan kendala lain.

Misalnya hanya untuk menentukan pola pemotongan optimal yang meminimumkan sisa pemotongan, dan diberikan satu kendala saja yaitu ukuran produk yang dihasilkan saja tanpa memperhitungkan kapasitas gudang penyimpanan, tingkat kualitas kayu, atau kendala lain yang berkaitan dengan

permasalahan. Sub bab berikut membahas tentang karakteristik, jenis, dan pola pemotongan bahan menurut Javanshir (2007).

2.1.1 Karakteristik *Cutting Stock*

Karakteristik *Cutting Stock* terdiri atas beberapa klasifikasi yang dijabarkan sebagai berikut:

1. Terdapat bahan baku yang berbentuk persegi empat (selanjutnya disebut *rectangle*) yang mempunyai ukuran tertentu.
2. Terdapat m jenis potongan yang dihasilkan (yang selanjutnya disebut dengan *pieces*) yang masing-masing berukuran $p_i \times l_i (i = 1...m)$ dengan jumlah permintaan n tertentu.
3. Setiap potong mempunyai nilai tertentu (v_i) yang bisa berupa keuntungan yang diperoleh atau berupa ukuran luas dalam upaya meminimasi sisa bahan baku.
4. Berusaha membentuk suatu *layout* potong yang meminimumkan fungsi tujuan yang melekat pada setiap potong yang ada.

2.1.2 Jenis *Cutting Stock*

Terdapat tiga pengelompokan jenis dari *cutting stock* yang dibahas menurut Javanshir (2007). Berikut beberapa jenis dari *cutting stock*:

1. Berdasarkan jumlah dimensi yang dipertimbangkan
   a. *One dimensional*
   b. *Two dimensional*
   c. *Three dimensional*
2. Berdasarkan jenis penugasan

a. *Big material to small pieces*

b. *Small pieces to big material*

3. Berdasarkan pada jumlah bahan yang dipertimbangkan

a. Satu macam ukuran bahan

b. Banyak ukuran bahan

2.1.3 Pola Pemotongan

Beberapa pola pemotongan yang dibahas dalam *cutting stock* adalah sebagai berikut:

1. *Guillotine Pattern*

Guillotine Pattern merupakan pola pemotongan yang dimulai dari satu sisi segi empat yang kemudian dilanjutkan pada sisi lainnya. Pemotongan pertama dengan tipe *Guillotine Pattern* adalah dengan memotong bahan baku dengan panjang atau lebar yang sama.

Pemotongan tersebut menghasilkan dua atau lebih *pieces* yang mempunyai panjang atau lebar yang sama, bukan kedua-duanya.

2. *Non-guillotine Pattern*

Pemotongan dengan tipe *non-guillotine* dilakukan apabila ukuran *pieces* yang diinginkan tidak memungkinkan untuk digabung dengan *pieces* yang lain.

3. Pola Dua Tahap Pemotongan (*Two Stage Pattern*)

Tahap pertama, pemotongan secara paralel atau pemotongan bahan secara horizontal, sehingga *rectangle* terbagi menjadi beberapa *rectangle* dengan panjang yang sama. Tahap kedua adalah pemotongan satu persatu bagian *rectangle*.

4. Pola Tiga Tahap Pemotongan (*Three Stage Pattern*)

   Tahap pertama, pemotongan *rectangle* menjadi bagian-bagian dengan panjang atau lebar yang sama. Arah pemotongan tersebut dapat secara vertikal maupun secara horizontal. Tahap kedua, hasil dari pemotongan tersebut dilanjutkan dengan pemotongan satu persatu yang terlebih dahulu mengubah arah pemotongan. Tahap ketiga, pemotongan dilakukan pada bagian yang menghasilkan *pieces*.

2.2 Teori Optimasi

Riset Operasi berusaha menetapkan arah tindakan terbaik (*optimum*) dari masalah keputusan dibawah pembatasan sumber daya yang terbatas. Istilah riset operasi seringkali diasosiasikan hampir secara khusus dengan penggunaan teknik-teknik matematika untuk membuat model dan menganalisa masalah keputusan. Walaupun matematika dan model matematis merupakan inti Riset Operasi, pemecahan masalah tidaklah hanya sekedar pengembangan dan pemecahan tiap model matematis (Taha, 1996). Beberapa tahapan utama yang harus dilalui oleh sebuah kelompok riset operasi untuk melakukan sebuah studi riset operasi mencakup:

1. Definisi Masalah
2. Pengembangan Model
3. Pengujian Keabsahan Model
4. Implementasi Hasil Akhir

2.2.1 Pemrograman Linier

Dalam tugas akhir ini digunakan model meminimalkan kerugian potongan kertas pada perusahaan percetakan dengan pemrograman Linier. Pemrograman Linier memakai model matematis untuk menggambarkan masalah yang dihadapi. Kata ”linier” berarti bahwa semua fungsi matematis dalam model ini harus merupakan fungsi-fungsi linier. Kata ”Pemrograman” merupakan sinonim dari kata perencanaan. Maka dari itu, membuat pemrograman linier adalah membuat rencana kegiatan-kegiatan untuk memperoleh hasil yang optimal, ialah suatu hasil untuk mencapai nilai yang paling menguntungkan baik nilai terbesar atau terkecil, tergantung pada apakah tujuannya memaksimalkan atau meminimumkan, dengan cara paling baik (sesuai model matematis) diantara semua alternatif yang mungkin (Yagiura, 2001).

Formulasi model matematik yang meliputi tiga tahap sebagai berikut:

1. Menentukan variabel keputusan dan menyatakan dalam simbol matematik
2. Fungsi obyektif/tujuan merupakan fungsi yang menggambarkan tujuan/sasaran dalam permasalahan program linier yang berkaitan dengan pengaturan secara optimal sumber daya, untuk memperoleh keuntungan secara maksimal atau biaya minimal.
3. Fungsi *Constraint*/batasan merupakan bentuk penyajian secara matematis batasan-batasan kapasitas yang tersedia yang akan dialokasikan secara optimal keberbagai kegiatan.
4. Sebagai contoh formulasi model matematik pemrograman linier:

Maksimumkan $Z = 3X_1 + 5X_2 + 2X_3$

Dengan kendala $5X_1 + 2X_2 + 4X_3 \leq 240$

$4X_1 + 6X_2 + 3X_3 \leq 400$

$X_1, X_2, X_3 \geq 0$

Dari contoh diatas dapat diketahui :

1. Variabel keputusan

   X_1, X_2 dan X_3

2. Fungsi tujuan

   Memaksimalkan $Z = 3X_1 + 5X_2 + 2X_3$

3. Batasan / kendala

   $5X_1 + 2X_2 + 4X_3 \leq 240$

   $4X_1 + 6X_2 + 3X_3 \leq 400$

   $X_1, X_2, X_3 \geq 0$

Bentuk umum dari pemrograman Linier adalah

$$Z = \sum_{j=1}^{n} CjXj$$

Maksimumkan (Minimumkan)

Dengan syarat C_jX_j $(\leq, =, \geq) b_i$, untuk semua i (i=1,2,3,…m) semua $x_j \geq 0$

Keterangan :

x_j = banyaknya kegiatan di j, dimana j=1,2,3,….n

Berarti disini terdapat n variable keputusan

Z : nilai fungsi tujuan

c_j : sumbangan per-unit kegiatan j

Untuk masalah maksimasi c_j, menunjukkan keuntungan atau penerimaan per unit, sementara kasus minimasi ia menunjukkan biaya per unit.

b_i : jumlah sumberdaya i (i=1,2,…,m)

Berarti terdapat m jenis sumber daya.

a_{ij} : banyaknya sumber daya i yang dikonsumsi sumber daya j

Optimasi potongan kertas yang diselesaikan dengan pemrograman linier dapat dipecahkan dengan pengembangan model matematis. Pengembangan model ini dapat dimulai dengan menentukan variablel, batasan dan menentukan tujuan yang harus dicapai untuk menentukan pemecahan optimum dari semua nilai yang layak dari variable tersebut. Cara efektif untuk menyelesaikan penentuan nilai-nilai di atas adalah memberi ringkasan verbal untuk masalah optimasi potongan kertas.

2.2.2 Metode Simpleks

Karena kesulitan menggambarkan grafik berdimensi banyak maka penyelesaian masalah pemrograman linier yang melibatkan lebih dari dua variabel menjadi tidak praktis atau tidak mungkin. Dalam keadaan ini kebutuhan mtode solusi yang lebih umum menjadi nyata. Metode umum ini dikenal dengan nama *simplex algorithm* yang dirancang untuk menyelesaikan seluruh masalah program linier, baik yang melibatkan dua variabel maupun lebih dua variabel (Fox, 1981).

Penyelesaian malasah pemrograman linier menggunakan metode simpleks ini melalui perhitungan ulang (*iteration*) dimana langkah langkah perhitungan yang sama diulang berkali-kali sebelum hasil optimum dicapai.

Dalam penyelesaian masalah program linier dengan grafik, telah dinyatakan bahwa solusi optimum selalu terletak pada titik pojok ruang solusi

(Yagiura, 2001). Metode simpleks didasar pada gagasan ini, dengan langkah-langkah sebagai berikut:

1. Dimulai pada suatu titik pojok yang layak, biasanya titik asal (yang disebut sebagai solusi awal).
2. Bergerak dari suatu titik ke pojok yang lain yang berdekatan, pergerakan ini akan menghasilkan nilai fungsi tujuan yang lebih baik (meningkat untuk masalah maksimasi dan menurunkan untuk masalah minimasi). Jika solusi yang lebih baik telah diperoleh, prosedur simpleks dengan sendirinya akan menghilangkan semua solusi-solusi lain yang kurang baik.
3. Proses ini dilakukan berulang-ulang sampai suatu solusi yang lebih baik tak dapat ditemukan. Proses simpleks kemudian berhenti dan solusi optimum diperoleh.

Mengubah bentuk baku model pemrograman linier ke dalam bentuk tabel akan memudahkan proses perhitungan simpleks (Yagiura, 2001). langkah-langkah perhitungan dalam algroritma simpleks adalah :

1. Berdasar bentuk baku, tentukan solusi awal, dengan menetapkan (n - m) variabel nonbasis sama dengan nol. Dimana n jumlah variabel dan m banyaknya kendala.
2. Pilih sebuah entering variabel diantara yang sedang menjadi variabel nonbasis, yang jika dinaikkan diatas nol dapat memperbaiki nilai fungsi tujuan. Jika tak ada, berhenti berarti solusi sudah optimal. jika tidak dilanjutkan ke langkah 1.

3. Pilih sebuah leaving variabel diantara yang sedang menjadi variabel basis yang harus menjadi nonbasis (nilainya menjadi nol) ketika entering variabel menjadi variabel basis.
4. Tentukan solusi yang baru dengan membuat entering variabel dan leaving variabel menjadi nonbasis. Kembali ke langkah 2.

2.2.3 Penambahan Batasan Baru

Penambahan batasan baru dapat menghasilkan satu di antara dua kondisi antara lain (Taha, 1996).

1. Batasan itu dipenuhi oleh pemecahan saat ini, dalam kasus mana batasan tersebut berlebihan dan penambahannya tidak mengubah pemecahan.
2. Batasan tersebut tidak dipenuhi oleh pemecahan saat ini,. Dalam kasus ini pemecahan baru diperoleh dengan metode simpleks dual.

Yang dilakukan di sini adalah mendapatkan kembali kelayakan. Tahap pertama, tempatkan batasan baru tersebut dalam bentuk standar dengan menambahkan variabel slack atau surplus sebagaimana diperlukan. Lalu substitusi keluar setiap variabel dasar saat ini dalam batasan tersebut dalam bentuk variabel non dasar (saat ini). Langkah terakhir adalah menambahkan batasan yag dimodifikasi ke tabel optimum saat ini dan menerapkan *simpleks dual* untuk memperoleh kembali kelayakan.

Batasan yang dimodifikasi ini sekarang ditambahkan ke tabel optimal saat ini seperti diberikan berikut ini:

Tabel 2.1 Kombinasi Potongan

Dasar	X1	X2	X3	X4	X5	X6	X7	Pemecahan
Z	0	0	1/3	4/3	0	0	0	38/3
X2	0	1	2/3	-1/3	0	0	0	4/3
X1	1	0	-1/3	2/3	0	0	0	10/3
X5	0	0	-1	1	1	0	0	3
X6	0	0	-2/3	1/3	0	1	0	2/3
X7	0	0	1/3	-2/3	0	0	1	-1/3

Dimana variabel :

X1 = Jumlah pola potong yang pertama.

X2 = Jumlah pola potong yang kedua.

X3 = Jumlah pola potong yang ketiga.

X4 = Jumlah pola potong yang keempat.

X5 = Jumlah pola potong yang kelima.

X6 = Jumlah pola potong yang keenam.

X7 = Jumlah pola potong yang ketujuh.

Jadi batasan baru tersebut yang diekspresikan dalam bentuk variabel nondasar menjadi:

$(10/3) + (1/3)x3 – (2/3)x4 + x7 = 3$

$(1/3)x3 – (2/3)x4 + x7 = -1/3$

2.3 SWEBOK (*Software Engineering Body of Knowledge*)

Sebagaimana yang dikemukakan oleh IEEE Computer Society professional Practices Committee (2004), "SWEBOK menggambarkan pengetahuan secara umum tentang rekayasa perangkat lunak yang dibagi ke dalam sepuluh area pengetahuan (*Knowledge Area's*) atau disebut KAs". Software Engineering Body of Knowledge (SWEBOK) adalah produk dari Komite Koordinasi Rekayasa Perangkat Lunak disponsori oleh IEEE Computer Society. SWEBOK sendiri mempunyai panduan yang disebut *Guide to the SWEBOK*, panduan ini dibuat untuk 5 tujuan, yaitu:

1. Untuk memperlihatkan kesamaan pandangan tentang rekayasa perangkat lunak di seluruh dunia.
2. Untuk memperjelas tempat dan menetapkan batas dari rekayasa perangkat lunak dan hubungannya dengan disiplin ilmu lain seperti ilmu komputer, manajemen proyek, teknik komputer dan matematika.
3. Untuk membuat karakter isi dari disiplin ilmu rekayasa perangkat lunak
4. Untuk memberikan akses topik ke SWEBOK
5. Untuk memberikan pengetahuan dasar bagi pengembangan kurikulum dan sertifikasi serta perizinan.

Berikut adalah penjabaran tentang ruang lingkup pengetahuan atau yang disebut juga sebagai *Knowledge Area's* (KAs) yang digunakan sebagai panduan dalam mengembangkan aplikasi oleh penulis sesuai dengan literatur:

2.3.1 *Software Requirements*

Tahapan awal dalam membangun aplikasi, *Software Requirements* merupakan sebuah properti yang disajikan untuk memenuhi kebutuhan dalam menyelesaikan permasalahan yang ada. Merujuk kepada aplikasi yang dikembangkan karena permasalahan yang ada akan diselesaikan oleh aplikasi tersebut. Menjabarkan bagaimana mengotomatiskan sebuah permasalahan sebuah tugas yang dihadapi oleh pengguna, membantu menganalisa proses bisnis perusahaan yang telah menggunakan aplikasi, menganalisa kekurangan yang ada pada aplikasi yang ada, dan lainnya. Berikut penjabaran tentang beberapa tahapan yang ada pada *software requirement*:

A *Requirement elicitation*

Tahapan awal dalam pemenuhan *software requirements* makna dari kebutuhan mendatang ini berhubungan dengan darimana kebutuhan perangkat lunak itu sendiri dan bagaimana para pengembang perangkat lunak dapat mengumpulkannya. Pada dasarnya, kegiatan yang dijabarkan adalah dari tiap individu dan tiap pemegang kendali sistem tersebut untuk membangun ketersinambungan antara pihak pengembang dan pengguna perangkat lunak itu sendiri.

B *Requirement analysis*

Tahapan ini membahas tentang kegiatan analisa kebutuhan untuk Mendeteksi dan menyelesaikan ketidakcocokan yang pada setiap kebutuhan yang ada.

1. Menggali batasan yang ada pada perangkat lunak yang dikembangkan dan bagaimana perangkat lunak tersebut akan berinteraksi dengan sistem.
2. Menguraikan kebutuhan sistem yang akan digunakan sebagai kebutuhan perangkat lunak.

C *Requirement specification*

Secara teknis pada kata "*specification*" mengacu pada banyaknya jumlah pekerjaan atau kemampuan perangkat lunak dalam mencapai tujuannya. Dalam sebuah istilah pengembangan perangkat lunak, "*software requirements specification*" secara khusus mengarah kepada hasil ketepatan, atau penyamaan elektronik, yang dapat di tinjau, di nilai, dan di benarkan.

D *Requirement Verification and Validation*

Beberapa dokumen *requirement*s diatas dapat menjadi bahan dari tahapan validasi dan verifikasi. Kebutuhan yang ada di validasi untuk menjamin bahwa pengembang dari perangkat lunak tersebut dapat memahami kebutuhan yang akan dicapai. Penyesuaian kebutuhan untuk standar perusahaan sangat penting untuk diperhatikan bahwa kebutuhan tersebut dapat dimengerti, konsisten, dan lengkap.

2.3.2 *Software design*

Design atau rancangan didefinisikan baik sebagai proses yang menjelaskan tentang arsitektur, komponen, antarmuka, dan karakteristik lain dari sistem atau komponen dan hasil yang memproses. Hasil rancangan perangkat lunak harus menggambarkan arsitektur perangkat lunak yang ada yaitu, bagaimana perangkat lunak diuraikan dan disusun dalam komponen dan

antarmuka dari tiap komponen. *Software Design* juga harus menggambarkan tiap komponen pada tingkat detail yang memungkinkan dalam konstruksinya.

Software Design memainkan peran penting dalam pengembangan perangkat lunak yang memungkinkan para pengembang perangkat lunak untuk menghasilkan berbagai model yang membentuk semacam cetak biru dari solusi yang akan diimplementasikan. Kita dapat menganalisa dan mengevaluasi beberapa macam model untuk menjabarkan berbagai macam kebutuhan apa saja yang saat kita sertakan kedalam perangkat lunak yang dikembangkan. Begitu pula dalam pertimbangan untuk beberapa solusi alternatif dan saran pengembangan. Pada akhirnya, pemodelan yang dihasilkan dapat dilanjutkan pada kegiatan pengembangan, dan juga pemodelan tersebut dapat digunakan sebagai masukan dan titik awal pada *software construction* dan *software testing*.

Mengenai beberapa 'pendekatan umum' yang dapat dijumpai sebagai panduan dalam proses pengembangan perangkat lunak. Berbeda tentang 'pendekatan umum' tersebut, 'metode' lebih banyak dijadikan sebagai acuan secara detail dan menyediakan kumpulan cara penggambaran yang akan digunakan, penjelasan pada tiap fungsi proses yang akan digunakan, dan kumpulan panduan dalam penggunaan metode tersebut.

A Perancangan terstruktur (*Function-Oriented Design*)

Perancangan terstruktur merupakan aktivitas mentransformasikan suatu hasil analisis ke dalam suatu perencanaan untuk dapat diimplementasikan (diotomasikan). Pendekatan perancangan terstruktur dimulai dari awal 1970. Pendekatan terstruktur dilengkapi dengan alat-alat (tools) dan teknik-teknik (*techniques*) yang dibutuhkan dalam pengembangan sistem, sehingga hasil akhir

dari sistem yang dikembangkan akan diperoleh sistem yang strukturnya didefinisikan dengan baik dan jelas (Jogiyanto, 2008).

Melalui pendekatan terstruktur, permasalahan yang komplek di organisasi dapat dipecahkan dan hasil dari sistem akam mudah untuk dipelihara, fleksibel, lebih memuaskan pemakainya, mempunyai dokumentasi yang baik, tepat waktu, sesuai dengan anggaran biaya pengembangan, dapat meningkatkan produktivitas dan kualitasnya akan lebih baik (bebas kesalahan) (Jogiyanto, 2008). Adapun aspek, elemen, *tools*, serta metodologi dari perancangan terstruktur, menurut Jogiyanto (2008) adalah sebagai berikut.

1. Aspek Perancangan Terstruktur
   a. Membantu pemecahan masalah
   b. Melakukan penyederhanaan system
   c. Menggunakan *graphic tool* agar sistem dapat dengan mudah dibaca dan dimengerti
   d. Memberikan rangkaian strategi untuk pengembangan solusi
   e. Memberikan kriteria dalam mengevaluasi solusi dengan melihat pada permasalahan aslinya
2. Elemen Perancangan Terstruktur
   a. Modul

      Modul merupakan sebuah instruksi atau sekumpulan instruksi program yang terdiri dari : input(masukan), output(keluaran), fungsi, mekanisme dan data internal. Contoh : Foxpro/Pascal (Procedure, function), COBOL (Program, section, paragraph), FORTRAN (subroutine).

b. Bagan terstruktur (Structured Chart)

Menggambarkan partisi sistem ke dalam : modul-modul, organisasi, dan komunikasi. Adapun keuntungan yang didapat dalam pemetaan secara visual kedalam bagan terstruktur yaitu ; Menggunakan gambar, Dapat dipartisi, Fleksibel, Input sangat berguna pada implementasi, Membantu pemeliharaan (maintenance) dan modifikasi.

c. Strategi Perancangan

Merubah hasil analisis (DFD) menjadi Bagan Terstruktur, untuk diimplementasi. DFD memperlihatkan aliran data dan informasi dari sistem. Jika dalam suatu DFD aliran datanya ditentukan oleh suatu data item, misalnya 'T' yang mempunyai nilai/ karakteristik tertentu, kemudian nilai ini akan mempengaruhi / menentukan arah aliran data (memicu arah), maka titik proses dimana terjadi percabangan arah aliran data tsb disebut titik pusat transaksi

3. *Tools* Perancangan Terstruktur

a. DFD (Data Flow Diagram)

b. Kamus Data

c. Entity Relationship Diagram (ERD)

d. State Transition Diagram (STD)

4. Metodologi Perancangan Terstruktur

a. Metodologi pemecahan fungsional

Metodologi ini menekankan pada pemecahan sistem ke dalam subsistem-subsistem yang lebih kecil, sehingga akan lebih mudah untuk dipahami, dirancang, dan diterapkan.

b. Metodologi berorientasi data

Metodologi ini menekankan pada karakteristik data yang akan diproses.

c. *Prescriptive methodologies*

Metodologi ini merupakan metodologi yang dikembangkan oleh sistem house dan pabrik-pabrik perangkat lunak dan tersedia secara komersial dalam paket-paket program.

2.3.3 *Software construction*

Istilah *Software Construction* mengacu pada penjabaran proses pembuatan pengerjaan, yang berarti sebuah perangkat lunak terbuat melalui serangkaian *coding*, pemeriksaan, pengetesan unit, pengetesan integrasi, dan *debugging*.

Ruang lingkup pengetahuan (KAs) dari *software construction* saling bersangkutan dengan seluruh KAs yang lain, lebih tepatnya sangat berhubungan erat pada kegiatan *Software Design* dan *Software Testing*. Hal ini dikarenakan pada proses yang ada pada *software construction* sendiri menyertakan kegiatan penting dari *software design* dan *software testing*. *Software construction* juga menggunakan keluaran (*output*) dari perancangan dan memberikan masukan (*input*) kepada pengetesan, dalam kasus ini perancangan dan pengetesan tersebut merupakan kegiatan, bukan KAs-nya. Batasan yang dijabarkan pada perancangan, pengembangan, dan pengetesan (jika ada) akan beragam berdasarkan pada proses siklus dari pengembangan pada perangkat lunak itu sendiri.

A Sekilas tentang Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, Integrated Development Environment (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual Source Safe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam bentuk Microsoft Intermediate Language di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).

Visual Studio kini telah menginjak versi Visual Studio 12.0.21005.1, atau dikenal dengan sebutan Microsoft Visual Studio 2012 yang diluncurkan pada 1 Agustus 2012, yang ditujukan untuk platform Microsoft .NET Framework 4.5.1 Versi tersebut kini dikenal dengan sebutan Visual Studio .NET, karena memang membutuhkan Microsoft .NET Framework. Sementara itu, sebelum muncul Visual Studio .NET, terdapat Microsoft Visual Studio 6.0 (VS1998).

B Sekilas tentang Microsoft SQL Server

Microsoft SQL Server diperkenalkan pada tahun 1990 untuk platform Microsoft OS/2 dalam kerjasamanya dengan Sybase. Produk ini berasal dari Sybase SQL Server 4.x untuk *platform* Unix. Dengan adanya Windows NT, muncul inisiatif untuk membangun SQL Server versi Windows NT sehingga dihasilkan Microsoft SQL Server versi 4.2 untuk platform Windows NT. Kerjasama dengan Sybase masih berlanjut dan diluncurkan SQL Server 6.0 pada tahun 1995 dan setahun kemudian SQL Server versi 6.5 diluncurkan.

SQL Server 6.5 memperbarui kemampuan transaksi dan menjadi produk database client/server yang banyak dipakai pada platform Windows NT. Untuk memenuhi kebutuhan pengguna yang makin meningkat, maka SQL Server perlu didisain ulang dan kerjasama dengan Sybase dihentikan. Kemudian Microsoft mengembangkan SQL Server 7.0 yang difokuskan pada tiga area yaitu : *easy to use*, *scalability* dan *data warehousing*. Pada tahun 2000, kemudian Microsoft meluncurkan SQL Server 2000. Di tahun 2005 ini, Microsoft mengeluarkan produk SQL Server versi terbarunya yaitu Microsoft SQL Server 2005 seiring dengan diluncurkannya Microsoft Visual Studio 2005 beta 2.

2.3.4 *Software Testing*

Tahapan yang dilakukan dalam mengevaluasi kualitas dari perangkat lunak itu sendiri, dan meningkatkannya, dengan mengidentifikasi kekurangan dan permasalahan yang dihadapi pada saat *software construction*.

Software testing terdiri dari beragam (*dynamic*) verifikasi yang terdapat pada tindakan program dengan batasan (*finite*) kumpulan uji kasus, yang paling tepat untuk dipilih (*selected*) dari bidang eksekusi yang tak terbatas, terhadap tindakan yang akan dicapai (*expected*).