Analisis dan Desain Pengembangan Modul Roughing 3-Axis pada Sistem CAM (*Computer Aided Manufacturing*) Berbasis Model Faset 3D

Gandjar Kiswanto, Abdurrasyid Mujahid
Departemen Teknik Mesin Universitas Indonesia,
Kampus Baru UI – Depok 16424
gandjar_kiswanto@eng.ui.ac.id

Abstrak

Setiap pengembangan perangkat lunak memiliki metodologi bagaimana perangkat tersebut dibuat. Metodologi tersebut seringkali merujuk kepada konsep pengembangan sistem perangkat lunak, atau bisa pula adalah modifikasi dari konsep yang sudah ada. Dalam studi kasus pada paper ini, akan dijelaskan metodologi perancangan sistem Object-Oriented Analysis & Design (OOAD) atau Analisis dan Desain Berbasis Objek yang digunakan oleh Laboratorium Teknologi Manufaktur dan Otomasi Departemen Teknik Mesin UI untuk mengembangkan modul roughing 3-Axis pada sistem CAM berbasis model faset 3D. Tahap awal OOAD sama halnya dengan dengan beberapa metode lainnya dimulai dengan mengindentifikasi kebutuhan atau fungsi-fungsi yang harus tersedia dalam sistem perangkat lunak. Selanjutnya OOAD melakukan realisasi dan visualisasi fungsi-fungsi tersebut dalam objek-objek. Interaksi dan relasi antar objek ini kemudian diwujudkan dalam impelementasi coding. Dengan demikian, di akhir proses pengembangan ini akan terkumpul dokumentasi konsep pengembangan sistem hingga implementasi teknis ke dalam code, serta manual untuk menjelaskan tata cara penggunaan dan pengembangan lanjutan.

Kata kunci: *sistem CAM, OOAD, Use case, Class diagram, Sequence diagram, objek, class, Java, UM, machining, roughing*

1. PENDAHULUAN

Analisis dan Desain Berbasis Objek atau OOAD adalah salah satu pendekatan dalam rekayasa atau pengembangan perangkat lunak yang memodelkan sebuah sistem sebagai kumpulan dari banyak objek yang saling berinteraksi. Setiap objek merupakan representasi dari entitas-entitas yang ada pada sistem tersebut dan dapat dibedakan berdasarkan attribut (elemen data) dan perilaku entitasnya [2].

Pada proses pengembangan perangkat lunak, OOAD memberikan pedoman berupa langkah-langkah yang bisa diikuti untuk memodelkan sistem ke dalam objek. Langkah-langkah tersebut tersusun secara sistematis mulai dari yang bersifat konsep hingga teknis implementasi.

Menurut [1] tahap-tahap yang perlu dilalui dalam OOAD adalah sebagai berikut :

1) Menentukan kebutuhan sistem yang akan dikembangkan. Yang dimaksud dengan kebutuhan adalah fungsi-fungsi (kemampuan) dan kondisi-kondisi yang harus ada pada sistem. Fungsi dan kondisi apa saja yang harus ada didokumentasikan dalam *Use Case Model*. *Use Case Model* memiliki dua output yaitu *Use Case Spesification* dan *Use Case Diagrams*.
2) Visualisasi konsep dalam *Domain Model*. *Domain Model* adalah representasi visual dari *conceptual classes* atau objek-objek dalam dunia nyata. *Domain Model* juga disebut dengan *conceptual model*, atau *domain object models*, atau *analysis object models*. Menggunakan notasi UML, *Domain Model* digambarkan dalam diagram kelas (*classes*) tanpa operasi (*method* atau *function*) di dalamnya. Yang disertakan dalam Domain Model adalah *conceptual classes*, asosiasi antar *class*, dan attribute masing-masing *class.*
3) Menjelaskan interaksi antar kelas (*class*) melalui diagram interaksi (*interaction diagram*). Ada dua bentuk diagram interaksi, yaitu diagram kolaborasi (*collaboration diagram*) dan diagram urutan (*sequence diagram*). Dalam studi kasus ini diagram interaksi yang digunakan adalah *sequence diagram.*
4) Realisasi kelas-kelas (mungkin juga *interface*) yang terlibat dalam sistem melalui diagram kelas (*class diagram*). Informasi yang disertakan dalam diagram ini adalah
   a. kelas, asosiasi antar kelas, dan atributnya;
   b. *interface*, beserta operasi dan konstantanya;
   c. *methods*;

d. jenis atribut;
e. *navigability*; dan
f. *dependencies*.

5) Implementasi: *mapping designs to code*

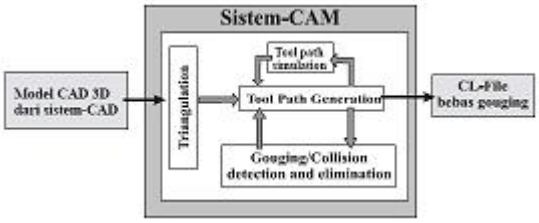
Tahap terakhir adalah pembuatan paket (*packaging delivery*) dan pembuatan dokumentasi penggunaan (*user manual*) dan dokumentasi pengembangan (*development manual*) perangkat lunak.

Sistematika penulisan dalam paper ini disusun berdasarkan tahapan OOAD di atas. Pertama akan dijelaskan fungsi atau fitur yang akan dibuat dalam modul roughing, kemudian permodelan analisis dan desain perancangan dalam beberapa diagram yang menggunakan notasi UML (*Unified Modeling Language*), yaitu *domain model*, *class diagram*, dan *sequence diagram*. Pada bagian akhir dilanjutkan dengan implementasi desain dalam code menggunakan bahasa pemrograman Java, serta ditutup dengan kesimpulan.

2. ANALISIS KEBUTUHAN

Untuk menggali kebutuhan atau fungsi-fungsi yang harus tersedia dalam perangkat lunak yang akan dikembangkan, maka terlebih dahulu harus diketahui arsitektur kerja sistem CAM ini.

Arsitektur kerja menjelaskan apa yang sebenarnya dikerjakan oleh sistem CAM ini. Sistem CAM bekerja sesuai dengan alur pemesinan pada umumnya dengan membaca input dari model geometri 3D (dari sistem CAD atau *Computer Aided Design*).



Gambar 1. Alur proses pembuatan lintasan pahat dari sistem CAD ke sistem CAM berbasis model faset 3D yang dikembangkan [3]

Karena sistem CAM ini berbasis faset 3D maka terlebih dahulu dilakukan triangulasi terhadap input 3D, kemudian baru bisa ditentukan lintasan pahatnya. Lintasan pahat ini berupa titik-titik koordinat yang akan dilalui oleh pahat pada mesin NC, biasanya dibuat dalam format CL (*cutter-location*) *file*. Untuk hasil yang tanpa cacat, maka lintasan pahat dibuat tanpa adanya interferensi atau *gouging* serta dibuatkan juga simulasi pemahatan model pahat di atas model produk 3D.

Sistem CAM ini memiliki modul utama sebanyak empat buah, sebuah kernel (*database*), dan *graphical user interface* (GUI). Empat modul utama tersebut adalah modul simulasi (*simulation*), modul perencana lintasan pahat (*tool path planning*), modul pembuat lintasan pahat dan pendeteksi dan penghilang interferensi atau gouging (*tool path generation*), dan modul pembaca triangulasi (*STL reader*).

Adapun modul yang dikembangkan dalam paper ini adalah modul perencana lintasan pahat (*tool path planning*) dan modul simulasi (*simulation*) untuk proses pemesinan mentah atau *roughing*.

Dalam skema alur pemesinan, *roughing* merupakan tahap awal yang bertujuan membentuk bahan mentah (misalnya dalam bentuk logam balok) sehingga mirip dengan model produk yang diinginkan. Proses *roughing* mengutamakan kecepatan pahat dalam memakan bahan mentah sehingga dengan cepat dapat diperoleh kemiripan dengan model produk. Hasilnya masih kasar, tugas untuk penghalusan diselesaikan pada tahap berikutnya, yaitu *finishing*. Paper ini tidak menjelaskan alur *finishing*.

Metode yang akan digunakan dalam pengerjaan *roughing* ini adalah metode *parallel* [4] dan metode *modified copying* [5]. Metode parallel memerlukan *roughing points* dasar sebelum prosesnya. Untuk itu perlu dibuat terlebih dahulu *roughing points* dasarnya. Sebenarnya *roughing points* dasar juga merupakan metode tersendiri dalam proses *roughing*. Jadi, ada tiga metode yang akan digunakan dalam modul *roughing* ini.

Masing-masing metode dilengkapi dengan fitur visualisasi titik-titik potong hasil *roughing* dan simulasinya di atas model produk.

Dengan demikian, secara ringkas kebutuhan fungsional yang harus dibuat dalam implementasi *roughing* ini adalah sebagai berikut:

1) Membuat *roughing points* dasar.
2) Melakukan *roughing* dengan teknik *parallel.*
3) Melakukan *roughing* dengan teknik *modified copying.*
4) Visualisasi *cutter-contact points* hasil *roughing* dasar.
5) Visualisasi *cutter-contact* points hasil *parallel roughing.*
6) Visualisasi *cutter-contact points* hasil *modified copying roughing.*
7) Visualisasi lintasan pahat hasil *roughing* dasar.
8) Visualisasi lintasan pahat hasil *parallel roughing.*
9) Visualisasi lintasan pahat hasil *modified copying roughing.*
10) Simulasi proses pemahatan menggunakan *roughing* dasar.
11) Simulasi proses pemahatan dengan teknik *parallel*
12) Simulasi proses pemahatan dengan teknik *modified copying*

Selain itu, ada pula kebutuhan non-fungsional yang harus terpenuhi, antara lain membuat sistem interaksi (*user interface*) yang menarik dan cukup *friendly*, membuat sistem navigasi sistem CAM yang mudah dipahami, serta sistem dapat bekerja dalam batas waktu yang bisa ditoleransi.

3. DESAIN PERANCANGAN SISTEM

Use Case Model

Langkah selanjutnya setelah *requirements review* adalah memodelkan kebutuhan-kebutuhan tersebut sehingga lebih jelas terlihat, beserta interaksi dengan pelaku-pelaku dalam sistem, atau yang sering disebut dengan aktor (*actor*). Kebutuhan sistem tersebut dimodelkan dalam *use case*.

Dalam sistem CAM ini, tidak ada klasifikasi khusus mengenai aktor-aktor yang terlibat. Jadi, hanya ada satu jenis aktor sistem, sebut saja dengan *User*. Semua *use case* ini bisa dibagi menjadi empat kelompok *use case* berdasarkan kelengkapan yang harus ada pada masing-masing metode *roughing.* Empat kelompok ini adalah

1) menghasilkan (*generating*) points,
2) visualisasi titik potong (*cutter-contact points*),
3) visualisasi lintasan pahat, dan
4) simulasi pergerakan pahat di atas model produk.

Empat kelompok *use case* di atas diimplementasikan dalam masing-masing metode *roughing*. Dengan demikian ada 12 *use case* yang harus diselesaikan dalam implementasi roughing ini (sama dengan 12 kebutuhan fungsional pada bagian sebelumnya).

Masing-masing *use case* ini dijelaskan secara detil dalam Use Case Spesification [1]. Contoh Use Case Spesification untuk *use case* Visualisasi Cutter-Contact Points Hasil *Roughing* Dasar sebagai berikut.

Deskripsi singkat	Dalam *use case* ini, User bisa meminta sistem untuk menampilkan *cutter-contact points* hasil *roughing* dasar.
Alur skenario	Skenario utama: 1) User memilih *view* untuk *roughing* (karena ada juga *view* untuk *finishing*). 2) User menekan *toggle button* untuk menampilkan *cutter-contact points* pada panel utama. 3) User bisa menekan kembali tombol tersebut untuk tidak menampilkannya. Skenario alternatif: Jika sistem belum selesai melakukan *roughing*, maka sistem akan memberikan notifikasi kepada User untuk menunggu hingga proses *roughing* selesai.
Precondition	Sistem sudah selesai melakukan *roughing* dasar.
Postcondition	*Cutter-contact points* ditampilkan di panel utama.
Kebutuhan khusus	Visualisasi hasil *roughing* ini bisa dilakukan bersama dengan visualisasi lainnya, yaitu visualisasi model (STL), lintasan pahat, dan simulasi.

Gambar 2. Use Case Spesification: visualisasi cutter-contact points hasil roughing dasar

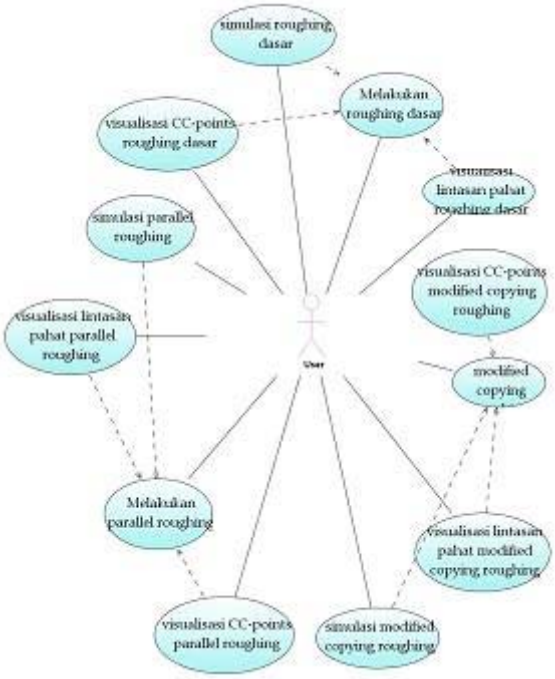
Secara keseluruhan, kedua belas *use case* di atas beserta interaksinya dengan user dapat digambarkan dalam Use Case Diagram. Notasi yang digunakan dalam Use Case Diagram ini adalah

notasi *Unified Modeling Language* (UML), dimana use case digambarkan dengan bentuk elips. Hubungan antar entitas (*use case* dan *actor*) digambarkan dengan notasi garis.

Dari diagram pada gambar 3 di bawah dapat terlihat ada tiga paket *roughing*, yaitu *roughing* dasar, *parallel roughing*, dan *modified copying roughing*, masing-masing dengan use case untuk visualisasi dan simulasi.

Domain Model

Tahap berikutnya adalah realisasi *use case* menjadi objek-objek dan interaksinya dalam Domain Model. Domain Model ini menjelaskan entitas-entitas dalam dunia nyata atau sebenarnya yang berkaitan atau terlibat dalam sistem. Entitas ini dinyatakan dalam *conceptual class*. *Conceptual class* bukanlah class dalam komponen perangkat lunak, ia hanyalah konsep objek.



Gambar 3. Use Case diagram

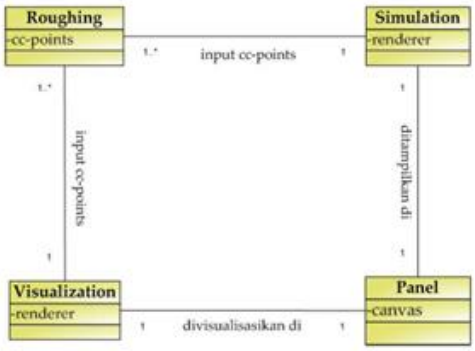
Domain Model digambarkan dengan sekumpulan diagram kelas tanpa operasi. Diagram kelas ini berisi *conceptual class*, asosiasi antar class, dan attribut masing-masing class.

Ada dua cara untuk mendapatkan conceptual classes, yaitu melalui *category list* dan *noun phrase indentification* [2]. Dalam implementasi modul roughing ini, *conceptual classes* dibuat menggunakan *category list*. Metode *noun phrase indentification* agak sulit dilakukan mengingat aktivitas dan entitas (seperti simulasi, roughing, dan visualisasi) merupakan benda yang bersifat abstrak. Daftar kategori yang digunakan untuk menemukan *conceptual classes* bisa dilihat pada tabel di bawah ini.

Tabel 1. Daftar kategori conceptual class

Conceptual Classes Category	Conceptual Classes Indentification
Tempat	Panel (tempat visualisasi & simulasi)
Benda abstrak	Simulasi, visualisasi
Event	Roughing

Dari *conceptual classes* yang sudah bisa diindentifikasi pada tabel di atas, maka bisa dibuat diagram keterhubungan yang mirip dengan *class diagram*. Gambar di bawah ini menggambarkan diagram keterhubungan dalam Domain Model.



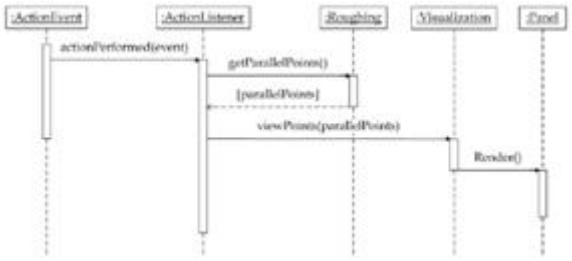
Gambar 4. Diagram Domain Model

Class-class dalam Domain Model ini akan menjadi input bagi proses berikutnya untuk menentukan class-class yang sebenarnya, beserta operasi-operasi yang dibutuhkan.

Sequence Diagram

Diagram ini dibuat untuk menjelaskan hubungan atau interaksi antar *class* yang sudah didentifikasi pada tahap sebelumnya. Interaksi antar *class* dibuat bedasarkan attribut apa saja bisa dimiliki oleh sebuah *class* dan apa yang bisa dilakukan oleh atau menggunakan attribut itu. Misalnya, class `Roughing` berinteraksi dengan class `Visualization` berkaitan dengan attribut `renderer` yang dimilikinya. Dengan menggunakan attribut `renderer` ini, *roughing points* yang sudah dibuat oleh class `Roughing` bisa divisualisasikan.

Contoh Sequence Diagram untuk visualisasi *cutter-contact* points hasil *parallel roughing* dijelaskan pada bagian di bawah ini.



Gambar 5. Sequence Diagram visualisasi cutter-contact points hasil parallel roughing

Use case ini dimulai dengan sebuah *event* dari tombol *view points*. *Listener*dari *event* ini kemudian meminta *parallel points* kepada *instance* `Roughing`. *Parallel points* yang sudah didapat dijadikan parameter untuk melakukan visualisasi dengan memanggil `viewPoints()`. *Instance* Visualization kemudian menggunakan operasi `Render()` untuk menampilkannya di atas *canvas* `Panel`.

Masing-masing use case dibuatkan Sequence Diagram, sehingga dengan demikian secara keseluruhan ada dua belas diagram.

Class Diagram

Sampai pada tahap ini, sudah dihasilkan *class-class* beserta operasi-operasinya. Dengan demikian, sudah bisa dibuat struktur *class-class* lengkap beserta attribut dan operasinya. Berdasarkan *conceptual classes* dalam Domain Model, ada empat kelas yang bisa dibuat, yaitu class `Roughing`, `Simulation`, `Visualization`, dan `Panel`. Berdasarkan *sequence diagram*, ada tambahan *class* yang berfungsi sebagai *listener*, nama *class*nya `MyActionListener`. Dengan demikian, sudah siap dibuat enam *class*.

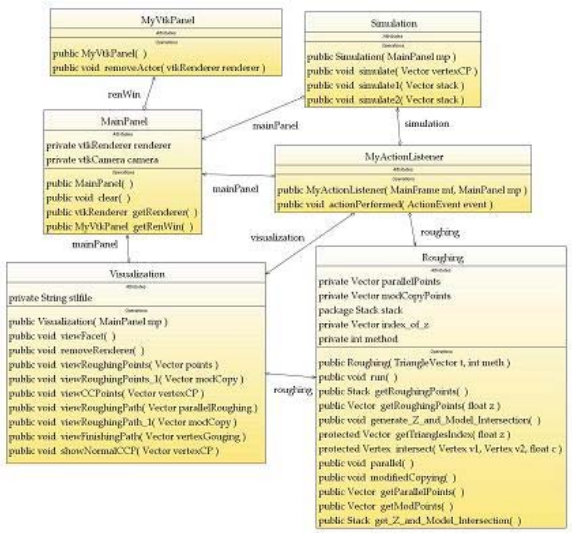
Implementasi class `Panel` (dinamai `MainPanel`) menggunakan class `javax.swing.Jpanel` dan dipadukan dengan class `vtkPanel` (dinamai `MyVtkPanel`) dari

library VTK. `MyVtkPanel` berfungsi sebagai tempat 'menggambar' objek-objek geometri dari *library* VTK, dan diletakkan di dalam class `MainPanel`.

Dengan demikian, sudah tersusun enam buah *class* yang bisa diimplementasikan, yaitu:

1) *class* `Roughing`,
2) *class* `Visualization`,
3) *class* `Simulation`,
4) *class* `MainPanel`,
5) *class* `MyVtkPanel`, dan
6) *class* `MyActionListener`.

Struktur *class-class* ini digambarkan dalam Class Diagram berikut ini :



Gambar 6. Class Diagram

Keenam *class* ini sudah siap untuk diimplementasikan dan dibuatkan paket-paketnya menggunakan bahasa pemrograman Java dan dibantu dengan alat pengembangan IDE NetBeans.

4. IMPLEMENTASI CODE

Tahap ini adalah merealisasikan class-class dalam diagram class di atas ke dalam *code* bahasa Java. Pemilihan bahasa pemrograman Java adalah sangat tepat mengingat aktivitas analisis dan desain perancangan yang dilakukan berbasis objek. Java (JDK atau Java Development Kit) adalah bahasa pemrograman berbasis obek (*object-oriented programming*).

Contoh implementasi code untuk class `Simulation` adalah sebagai berikut :

```
17  import vtk.vtkPolyDataMapper;
18  import vtk.vtkSphereSource;
19  import vtk.vtkTubeFilter;
20
21  /**...*/
25  public class Simulation{
26
27      private MainPanel mainPanel;
28      private Machining machining;
29
30      /** Creates a new instance of Simulation */
31      public Simulation(MainPanel mp) {
32          mainPanel= mp;
33      }
34
35      /**...*/
36      public void simulate(Vector vertexCP){...}
126
127     /** tested for roughing: circle, parallel */
128     public void simulate1(Vector stack){
129         int size = stack.size();
130         float radius= Configuration.TOOL_RADIUS;
131         float height= Configuration.TOOL_LENGTH;
132
133         int n=0;
134         while (n<stack.size()){
135             VertexVector roughPoints= (VertexVector) stack.get(n++);
136             int vertexSize= roughPoints.size();
137
138             for (int i = 0; i<vertexSize;i++){
139                 Vertex v = (Vertex)roughPoints.get(i);
```

```
140
141
142                 }
143             }
144         }
145
146     /** tested for roughing: modified copying */
147     public void simulate2(Vector stack){
148         int size = stack.size();
149         float radius= Configuration.TOOL_RADIUS;
150         float height= Configuration.TOOL_LENGTH;
151
152         for (int i = 0; i<size; i++){
153             Vertex v = (Vertex)stack.get(i);
154
155         }
156     }
157 }
```

5. DESAIN SISTEM INTERAKSI

Dengan mengacu kepada *8 Golden Rules of Interface Design*, sistem interaksi dan *user interface* yang dibuat sebisa mungkin memenuhi kriteria berikut :

1. Konsisten

Konsistensi ini dilihat dari bahasa yang digunakan, yaitu bahasa Inggris. Semua menu dan *icon shortcut* menggunakan bahasa Inggris. Semua satuan pengukuran menggunakan milimeter (mm). Istilah yang digunakan juga sesuai dengan istilah pemesinan, seperti *finishing*, *roughing*, dan sebagainya.

2. Sesering mungkin menggunakan *shortcut*

Icon-icon untuk operasi utama pemesinan disediakan pada toolbar utama. Ini bisa dilihat pada Gambar 7 di bawah.

3. Memberikan *feedback* yang informatif

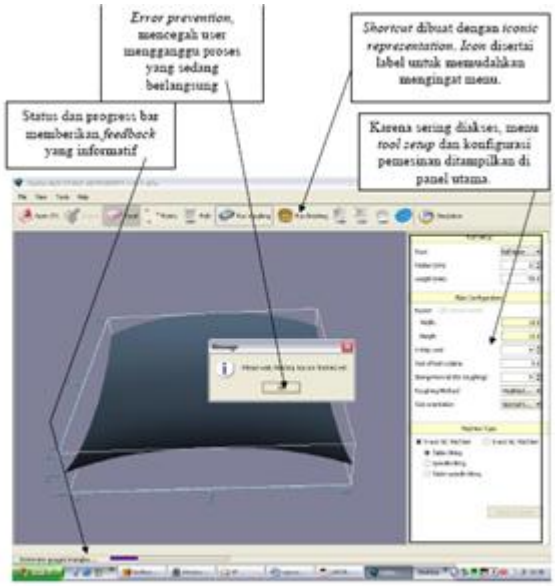
User diberikan informasi tentang proses yang sedang berlangsung dan yang telah selesai. Informasi ini diberikan melalui *progress bar* dan *status bar* pada bagian bawah *user interface*.

4. Pencegahan kesalahan (error)

Salah satu contohnya adalah jika proses pemesinan sedang berlangsung, maka user belum bisa menampilkan titik-titik potong atau lintasan pahat. Untuk mencegah user melakukan yang demikian, maka sistem memberikan notifikasi kepada user untuk menunggu hingga proses tersebut selesai.

5. Mengurangi beban memori jangka pendek

Ini dilakukan dengan menggunakan *menu-menu item* yang sederhana, tidak dalam, dan tersebar. Pada beberapa *icon* juga diberikan label untuk membantu mengingat fungsi *icon* tersebut.



Gambar 7. User interface sistem

6. KESIMPULAN

Pengembangan perangkat lunak menggunakan metodologi *Object-Oriented Analysis and Design* atau OOAD diawali dengan penentuan kebutuhan atau fungsi-fungsi yang harus tersedia dalam sistem. Kebutuhan tersebut dimodelkan dalam *usecase-usecase* menggunakan Use Case Diagram. Masing-masing *use case* dijelaskan dalam Use Case Spesification. Tahap berikutnya adalah realisasi kebutuhan (*requirements*) dalam objek-objek. Relasi dan interaksi antar objek harus ditentukan agar diperoleh realisasi fungsi-fungsi yang dibutuhkan dalam sistem. Relasi dan interaksi antar objek ini digambarkan melalui Sequence Diagram dan Class Diagram. Class Diagram inilah yang memberikan arah taktis mengimplementasikan objek-objek beserta attribut dan perilakunya dalam class-class sesuai bahasa pemrograman yang digunakan. Dalam bahasa pemrograman berbasis objek, yaitu Java, setiap class dalam Class Diagram menjadi satu class dalam *coding*nya, berisi *method* dan *variable* yang tidak lain adalah attribut dan perilaku dari class itu sendiri.

DAFTAR ACUAN

[1] Larman, Craig. (2002). **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process 2nd Ed**, Prentice Hall, USA.

[2] http://en.wikipedia.org/wiki/Object-oriented_analysis_and_design. Diakses pada tanggal 26 September 2007.

[3] Kiswanto, Gandjar. (2005). **Pengembangan dan Pembuatan Sistem CAM (Computer-Aided Manufacturing) yang Handal berbasis Model Faset 3D untuk Pemesinan Multi-axis dengan Optimasi Orientasi Pahat dan Segmentasi Area dan Arah Pemesinan**, Laporan Kemajuan RUT XII Tahap II Tahun 2005, Kementrian Riset dan Teknologi dan LIPI.

[4] Kiswanto, Gandjar**.** (2007). **Implementasi Roughing 3-Axis dengan Teknik Parallel pada Sistem Computer-Aided Manufacturing (CAM) berbasis Model Faset 3D dengan Simulasi Pergerakan Pahat.**

[5] Kiswanto, Gandjar**.** (2007). **Implementasi Roughing 3-Axis dengan Teknik *Modified Copying* pada Sistem Computer-Aided Manufacturing (CAM) berbasis Model Faset 3D dengan Simulasi Pergerakan Pahat.**

[6] Mandel, Theo. (1997). **The Golden Rules of User Interface Design**. http://www.theomandel.com/docs/Mandel-GoldenRules.pdf. Diakses pada tanggal 27 September 2007.