BAB II. LANDASAN TEORI

II.1 *RATIONAL UNIFIED PROCESS* (RUP)

Metodologi *Rational Unified Process* (RUP) merupakan suatu proses rekayasa perangkat lunak yang dikembangkan oleh *Rational Software Corporation*, dengan mengumpulkan beberapa prosedur terbaik di industri pengembangan perangkat lunak, seperti siklus yang bersifat iteratif dan pengendalian resiko [Larman, 2002].

Daur proses pengembangan perangkat lunak dalam RUP terdiri dari empat tahapan yang saling berkesinambungan dalam waktu, yang disebut sebagai fase. Setiap fase memiliki target yang harus dicapai. Fase-fase dalam RUP adalah sebagai berikut [RUPBest, 1998]:

- Insepsi

  Tujuan utama dari fase insepsi adalah untuk mencapai keselarasan antara semua *stakeholder* dalam sebuah daur proses suatu proyek. Dalam fase ini dirumuskan kesamaan visi, batasan masalah, tujuan, dan estimasi.

- Elaborasi

  Dalam fase elaborasi dilakukan penyelarasan dasar-dasar arsitektur sistem yang nantinya akan memberikan panduan yang stabil untuk membuat desain secara keseluruhan dan pengimplementasian kerja dalam fase selanjutnya.

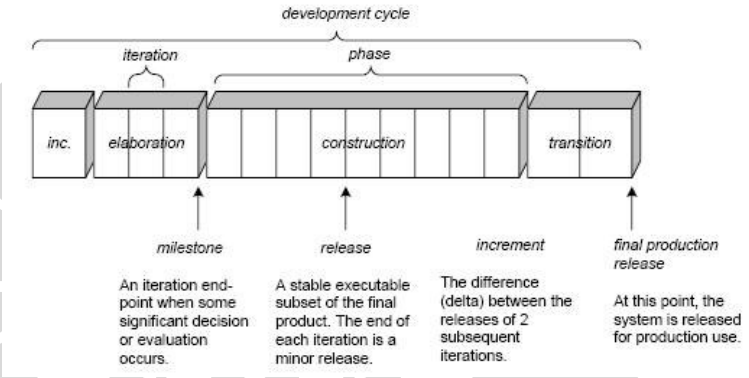
- Konstruksi

  Pada fase ini dilakukan klarifikasi terhadap kebutuhan yang telah dirumuskan

dan juga melengkapi pembangunan sistem berdasarkan arsitektur.

- Transisi

  Fokus dari fase ini adalah untuk meyakinkan bahwa perangkat lunak yang dikembangkan dapat digunakan oleh pengguna akhir. Fase ini dikembangkan dengan beberapa iterasi, mencakup beberapa pengujian terhadap produk berdasarkan umpan balik dari pengguna.



Gambar II-1. Fase dalam *Rational Unified Process* [Larman, 2002]

Arsitektur RUP dibagi menjadi dua dimensi, yaitu waktu dan proses kerja. Kedua bagian tersebut digambarkan sebagai dua sumbu yang saling tegak lurus dalam suatu matriks. Sumbu horisontal menggambarkan waktu dan menunjukkan daur hidup dari proses. Dimensi waktu dibagi menjadi 2 (dua) bagian, yaitu fase dan iterasi. Sumbu vertikal menggambarkan proses kerja, yang merupakan suatu kumpulan aktivitas yang saling berhubungan satu dengan lainnya.

Di dalam metodologi RUP terdapat 9 (sembilan) proses kerja, dimana masing-masing mempunyai tahapan yang berbeda sesuai alur kerja detailnya masing-masing. Dengan mengacu pada sub bab ruang lingkup pembahasan, maka tidak semuanya akan dibahas di dalam penulisan proyek akhir ini. Proses kerja dalam

RUP yang menjadi fokus utama diantaranya:

1) Pemodelan bisnis

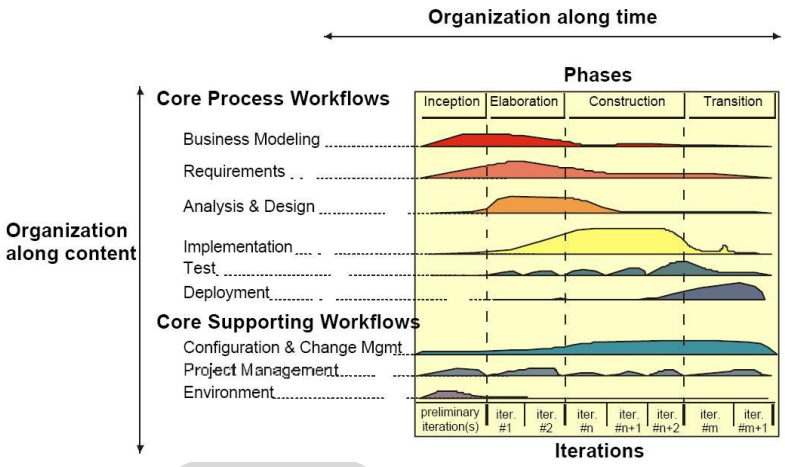
   Tujuan utama dari proses ini adalah untuk memahami struktur dan dinamika dari organisasi dimana sistem akan diterapkan, memahami permasalahan yang dihadapi oleh organisasi dan perbaikan yang perlu dilakukan, memastikan pengguna dan pengembang memahami organisasi dan mengumpulkan daftar kebutuhan yang diperlukan bagi pengembangan sistem.

2) Kebutuhan Sistem

   Kebutuhan sistem merupakan sebuah keadaan atau kapabilitas dimana sebuah sistem harus dapat melakukannya. Tujuan dari disiplin ini adalah untuk mencapai suatu persetujuan dengan *stakeholder* tentang apa yang harus dilakukan oleh sistem, menyediakan pemahaman tentang kebutuhan sistem bagi pengembang sistem, menentukan batasan sistem, menyediakan dasar untuk memperkirakan biaya dan waktu untuk membangun sistem.

3) Analisis dan Desain

   Pada tahapan ini dideskripsikan beberapa hal mengenai bagaimana sistem ini akan direalisasikan di fase implementasi. Tujuan utama tahapan ini adalah mengubah kebutuhan sistem menjadi suatu desain dari sistem. Dalam tahapan ini juga dilakukan adaptasi desain yang ada, kemudian dicocokkan dengan sumber daya dari implementasi.



Gambar II-2. Arsitektur *Rational Unified Process* [RUPBest, 1998]

II.2 *UNIFIED MODELING LANGUAGE* (UML)

Unified Modeling Language (UML) didefinisikan sebagai sebuah "bahasa grafis" yang digunakan untuk proses visualisasi, perancangan, dan pendokumentasian sebuah sistem atau perangkat lunak [Booch, 1998]. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Untuk mendapatkan spesifikasi perangkat lunak yang sesuai dengan kebutuhan pengguna, para pengembang melakukan pemodelan-pemodelan secara visual. Pemodelan visual adalah proses penggambaran informasi-informasi secara grafis dengan notasi-notasi baku yang telah disepakati sebelumnya. Notasi-notasi baku ini sangat penting demi suatu keseragaman pembacaan suatu gambar. Dengan adanya "bahasa" yang bersifat standar, komunikasi perancang dengan pemrogram serta calon pengguna diharapkan menjadi lancar.

Dalam kerangka spesifikasi, UML menyediakan model-model tepat, tidak ambigu, serta lengkap. Secara khusus, UML menspesifikasi langkah-langkah

penting dalam pengambilan keputusan analisis, perancangan, serta implementasi dalam sistem yang bernuansa perangkat lunak (*software intensive system*). Dalam hal ini, UML bukanlah merupakan bahasa pemrograman, tetapi model-model yang tercipta berhubungan langsung dengan berbagai macam bahasa pemrograman, sehingga dimungkinkan melakukan pemetaan langsung dari model-model yang dibuat dengan UML ke bahasa-bahasa pemrograman berorientasi obyek, seperti: *Java*, *Borland Delphi*, *Visual Basic*, C++, dan lain-lain. Pemetaan ini dapat bersifat dua arah, yakni generasi kode bahasa pemrograman tertentu dari UML (*forward engineering*), atau sebaliknya melakukan langkah balik (bersifat iteratif) dari implementasi ke UML (*reverse engineering*).

Untuk melakukan pemodelan sistem informasi / perangkat lunak dalam proyek akhir ini, penulis menggunakan notasi-notasi UML yang akan digambarkan secara elektronik menggunakan aplikasi bantu *Rational Rose*.

Dengan pemodelan menggunakan UML, pengembang sistem / perangkat lunak dapat melakukan:

- Tinjauan umum bagaimana arsitektur sistem secara keseluruhan;
- Penelaahan bagaimana obyek-obyek dalam sistem saling mengirimkan pesan dan saling bekerjasama satu sama lain;
- Menguji apakah perangkat lunak sudah berfungsi dengan seharusnya;
- Dokumentasi sistem / perangkat lunak untuk keperluan-keperluan tertentu di masa yang akan datang.

Setiap sitem yang kompleks seharusnya bisa dipandang dari sudut yang berbeda-beda, sehingga kita bisa mendapatkan pemahaman secara menyeluruh. Untuk

upaya tersebut, UML (versi 2.0) menyediakan 13 (tiga belas) jenis diagram resmi, dimana masing-masing diagram tersebut adalah:

- *Use case diagram*
- *Activity diagram*
- *Sequence diagram*
- *Collaboration diagram* (UML versi 1.x) atau *Communication Diagram* (UML versi 2.0)
- *Class diagram*
- *Statechart diagram* (UML versi 1.x) atau *State Machine* (UML versi 2.0)
- *Component diagram*
- *Deployment diagram*
- *Object diagram*
- *Package diagram*
- *Composite structure diagram*
- *Interaction overview diagram*
- *Timing diagram*

Ketigabelas diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Dalam pemodelan dengan UML, dimungkinkan pula penggunaan diagram-diagram lain sejauh itu memang diperlukan untuk mendapatkan pemahaman mendalam tentang suatu sistem / perangkat lunak. Diagram-diagram lain yang dapat digunakan seperti *data flow diagram* (DFD), *entity relationship diagram* (ERD), diagram alir (*flowchart*), dan sebagainya.

Umumnya sebuah sistem mempunyai sejumlah *stakeholder*, yakni orang yang

mempunyai ketertarikan pada suatu sistem, namun dari sudut pandang yang berbeda. Sebagai contoh, saat kita merancang sebuah sistem untuk seorang pelanggan, maka akan sangat berbeda ketika sistem tersebut kita terapkan untuk banyak pelanggan. Dari sini jelas bahwa kita butuh banyak diagram dari berbagai sudut pandang yang berbeda-beda. Dengan demikian tujuan utama dari banyaknya diagram ini adalah untuk memuaskan semua keinginan *stakeholder*.

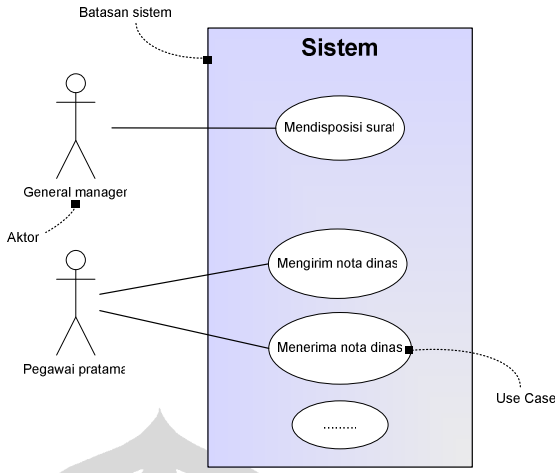
II.3 DIAGRAM-DIAGRAM UML

II.3.1 Diagram *Use Case*

Diagram ini memperlihatkan himpunan *use case* dan aktor-aktor. Diagram ini sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.

Use case adalah alat bantu guna menstimulasikan pengguna potensial untuk mengatakan tentang suatu sistem dari sudut pandangnya. Ide dasarnya dalah bagaimana melibatkan penggunaan sistem di fase-fase awal analisis dan perancangan sistem. Dengan demkian diharapkan akan dibangun suatu sistem yang bisa membantu pengguna.

Diagram *use case* menunjukkan tiga aspek dari sistem, yakni aktor, *use case*, dan sistem. Aktor dapat mewakili peran orang, alat atau sistem yang lain ketika berkomunikasi dengan *use case*. Untuk lebih jelasnya contoh digaram *use case* dapat dilihat pada gambar Gambar II-3.



Gambar II-3. Contoh Diagram *Use Case*

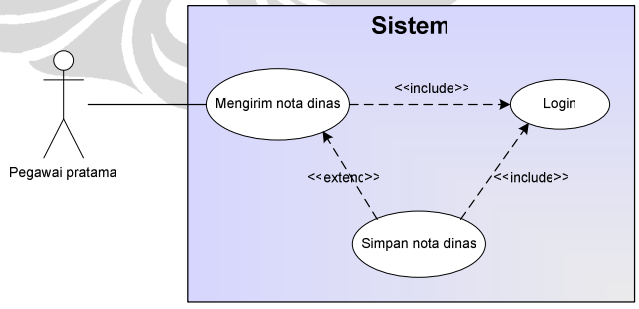
Adapun penggunaan diagram *use case* memiliki fungsi dan kegunaan sebagai berikut [Prasetyo, 2006]:

- Para pengembang dapat menggunakan diagram *use case* untuk mendapatkan pemahaman yang komprehensif tentang sistem dan lingkungan luar yang melingkupi sistem;
- Para calon pengguna dan manajer proyek dapat menggunakan diagram *use case* untuk mendapatkan pandangan paling atas tentang sistem dan untuk menentukan persetujuan tentang lingkup proyek;
- Manajer proyek dapat menggunakan diagram *use case* untuk membagi sistem secara keseluruhan menjadi bagian-bagian yang dapat dikelola dengan seksama;
- Calon pengguna dan analis sistem dapat melihat diagram *use case* untuk dapat memahami fungsionalitas sistem yang diharapkan;
- Para penulis teknik dapat melihat diagram *use case* untuk menulis panduan-panduan sistem dan merancang pelatihan-pelatihan.

Wawancara dengan *stakeholder* adalah teknik yang tepat untuk menggali *use*

case. *Use case* dibuat berdasarkan keperluan aktor. *Use case* harus merupakan ”apa” yang dikerjakan perangkat lunak aplikasi dan bukan ”bagaimana” perangkat lunak aplikasi mengerjakannya [Dharwiyanti, 2003]. Setiap *use case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan aktor. Nama *use case* boleh terdiri dari beberapa kata, dan tidak boleh ada dua *use case* yang memiliki nama yang sama.

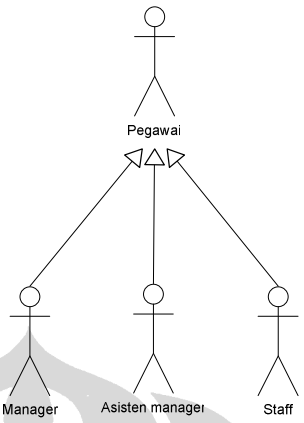
Dalam diagram *use case* dikenal pula istilah *stereotype* dan generalisasi. *Stereotype* adalah sebuah model khusus yang terbatas untuk kondisi tertentu. Untuk menunjukkan *stereotype* digunakan simbol ”<< ..... >>”. Simbol <<*extend*>> digunakan untuk menunjukkan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* yang lain, jika kondisi atau syarat tertentu yang dipenuhi. Sedangkan <<*include*>> digunakan untuk menggambarkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya. Biasanya <<*include*>> digunakan untuk menghindari pengkopian suatu *use case* karena sering dipakai. Contoh penggunaan <<*include*>> dan <<*extend*>> dapat dilihat pada gambar Gambar II-4:



Gambar II-4. Contoh Penggunaan *Stereotype Extend* dan *Include*

Generaliasai digunakan untuk spesialisasi aktor yang bisa berpartisipasi di semua *use case* yang diasosiasikan dengan aktor yang lebih *general*. Untuk lebih

jelasnya dapat dilihat pada gambar Gambar II-5.



Gambar II-5. Contoh Penggunaan Generalisasi

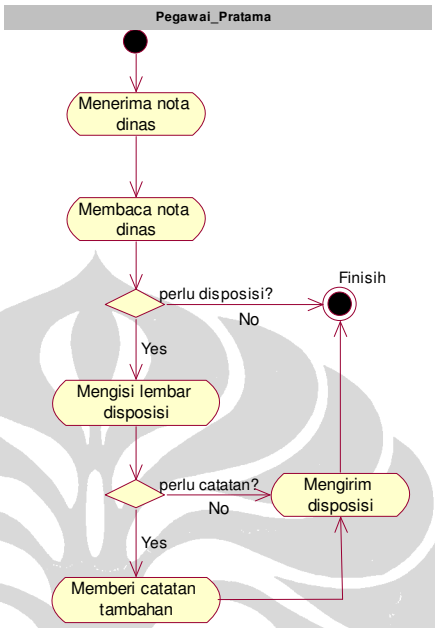
Seringkali perubahan pada proses bisnis adalah jalan terbaik untuk menyelesaikan masalah. Oleh karena itu, *use case* perlu dibedakan menjadi sistem *use case* dan *business use case*. Sistem *use case* adalah sebuah interaksi dengan perangkat lunak, sedangkan *business use case* lebih menekankan kepada bagaimana sebuah bisnis merespon ke pelanggan atau kejadian (*event*).

II.3.2 Diagram *Activity*

Diagram ini memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem. Diagram ini penting dalam pemodelan fungsi-fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar obyek. Calon pengguna dan analis sistem dapat melihat diagram *activity* untuk memahami aliran aktivitas yang terjadi di dalam sistem.

Diagram *activity* mempunyai peran seperti halnya diagram alir (*flowchart*), akan tetapi ada perbedaan dengan diagram alir; yaitu dapat mendukung perilaku paralel, sedangkan diagram alir tidak. Diagram *Activity* dapat pula digunakan

untuk menunjukkan ”siapa” mengerjakan ”apa” dengan teknik partisi. Contoh sederhana dari diagram *activity* dapat dilihat pada gambar Gambar II-6.

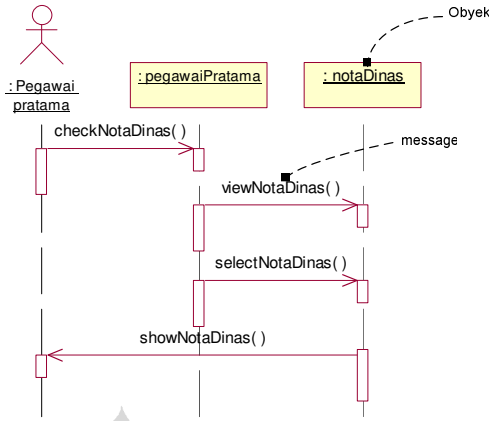


Gambar II-6. Contoh Diagram *Activity*

II.3.3 Diagram *Sequence*

Diagram *Sequence* merupakan diagram yang menyajikan interaksi antar obyek yang digambarkan dalam grafik dua dimensi. Sumbu horisontal merepresentasikan obyek / individu yang berperan dalam proses interaksi. Sumbu vertikal merepresentasikan waktu, dimana dimensi waktu bergerak dari atas ke bawah. Garis waktu setiap obyek digambarkan sebagai garis terputus-putus di bawah masing-masing obyek. Proses interaksi / komunikasi antar obyek dilakukan melalui pengiriman pesan (*message*) yang digambarkan sebagai sebuah garis panah lurus dari satu obyek ke obyek lainnya.

Contoh penggunaan diagram *sequence* dapat dilihat pada gambar Gambar II-7:

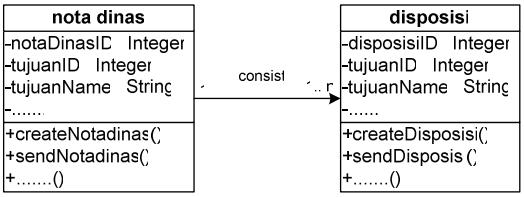


Gambar II-7. Contoh Diagram *Sequence*

II.3.4 Diagram *Class*

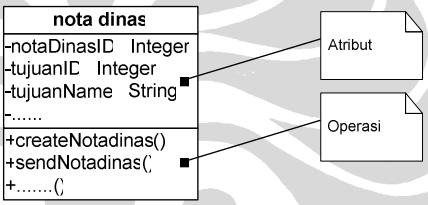
Diagram *Class* adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem / perangkat lunak yang sedang kita kembangkan. Diagram ini memberi gambaran atau diagram statis tentang sistem / perangkat lunak serta relasi-relasi yang ada di dalamnya.

Diagram *class* adalah kakas perancangan utama bagi tim pengembang. Diagram *class* akan membantu para pengembang untuk melihat dan merencanakan struktur dari sistem / perangkat lunak sebelum kode pemrograman dituliskan oleh para pemrogram kelak (atau dihasilkan secara otomatis oleh *Rational Rose*). Hal ini akan membantu para pengembang untuk memastikan bahwa sistem / perangkat lunak dianalisis dan dirancang dengan baik. Contoh penggunaan diagram *class* dapat dilihat pada gambar Gambar II-8.



Gambar II-8. Contoh Diagram *Class*

Kelas, seperti juga obyek adalah sesuatu yang membungkus (*encapsulate*) informasi (atribut) dan perilaku (operasi) dalam dirinya. Pendekatan berorientasi obyek menggabungkan potongan-potongan informasi dengan perilaku-perlaku yang akan mengaksesnya dalam apa yang dinamakan kelas.



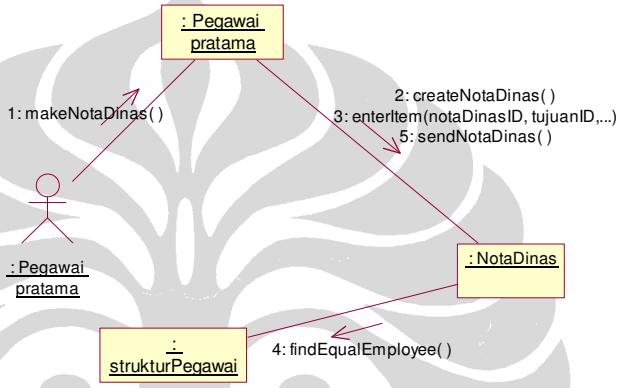
Gambar II-9. Atribut Diagram *Class*

Gambar Gambar II-9 memperlihatkan notasi atau atribut kelas dalam UML. Bagian paling atas memuat nama kelas dan (jika ada) *stereotype*-nya. Bagian tengah menunjukkan atribut-atribut yang dimiliki sebuah kelas. Sedangkan bagian paling bawah menunjukkan operasi-operasi yang dimiliki kelas yang bersangkutan. Dalam notasi kelas, biasanya juga akan terlihat tingkat berbagi pakai dan penampakan (*visibility*) atribut-atribut serta operasi-operasi.

II.3.5 Diagram *Collaboration*

Sama halnya dengan diagram *sequence*, diagram *collaboration* merupakan diagram yang menggambarkan hubungan interaksi antar obyek di dalam sistem. Perbedaanya adalah pada diagram *collaboration* lebih fokus pada hubungan atau

relasi antar obyek tersebut dan tidak terikat dengan waktu [Boogs, 2002]. Hubungan antar obyek digambarkan dengan garis, yang didalamnya berisi pesan (*message*) yang dikirimkan dari satu obyek ke obyek lainnya. Pesan yang dikirimkan tersebut diberi urutan nomor, yang mengindikasikan urutan waktu dari pesan tersebut. Contoh penggunaan diagram *collaboration* dapat dilihat pada gambar Gambar II-10.



Gambar II-10. Contoh Diagram *Collaboration*

II.4 ADMINISTRASI PERKANTORAN

Sistem administrasi perkantoran PT (Persero) Angkasa Pura I diatur dan ditetapkan dengan Keputusan Direksi Nomor SKEP.108/UM.00/2003. Sistem administrasi perkantoran merupakan penunjang manajemen dengan lingkup kegiatan mengatur dan mengolah lalu lintas informasi tertulis yang dikenal juga sebagai kegiatan surat menyurat dan kearsipan, oleh karena itu kegiatan administrasi perkantoran sebagai suatu sistem menjadi sangat strategis [KepDir 108, 2003].

Sistem administrasi perkantoran PT (Persero) Angkasa Pura I bersifat seragam, terpadu, tersentralisasi, dan memilki mekanisme khusus dalam proses pengelolaan

dan pengolahan naskah.

II.4.1 Maksud dan Tujuan Administrasi Perkantoran

Sistem administrasi perkantoran disusun dengan maksud untuk digunakan sebagai pedoman umum dalam pengelolaan persuratan dan kearsipan di lingkungan organisasi yang bertujuan untuk [SAP, 2003]:

- Menciptakan keseragaman dan keterpaduan penyelenggaraan pola kegiatan persuratan dan kearsipan dalam berbagai media;
- Mewujudkan tertib administrasi umum yang tepat guna dan berhasil guna;
- Menunjang kelancaran komunikasi kedinasan baik internal maupun eksternal dalam rangka pencapaian visi dan misi perusahaan;
- Menjamin otentikasi, reliabilitas, dan keselamatan bahan-bahan bukti berupa dokumen-dokumen perusahaan.

II.4.2 Azas Sistem Administrasi Perkantoran

Azas sistem administrasi perkantoran PT (Persero) Angkasa Pura I dibagi menjadi 4 (empat), yaitu [SAP, 2003]:

1) Azas Keamanan

   Pada dasarnya semua isi produk administrasi bersifat tertutup, sehingga kerahasiaan isinya tetap harus dijaga. Oleh karena itu para pejabat dan petugas persuratan dan kearsipan khususnya dan pejabat serta pegawai pada umumnya tidak boleh memberikan informasi kepada yang tidak berkepentingan, baik secara lisan maupun tertulis.

2) Azas Pembakuan

Semua produk administrasi harus diproses dan disusun menurut tata cara serta bentuk yang telah ditetapkan. Di dalam petunjuk teknis tata cara dan bentuk-bentuk yang siatnya menyeluruh hendaknya dibakukan dengan tetap memperhitungkan kegiatan yang bersifat khusus. Dengan demikian diperoleh 2 (dua) manfaat sekaligus yaitu tepat guna dan berhasil guna.

3) Azas Keterpaduan

Persuratan dan kearsipan merupakan bagian integral dari manajemen atau administrasi perusahaan. Dengan demikian seluruh kegiatan persuratan dan kearsipan hendaknya dilaksanakan secara terpadu sejak penciptaan hingga penyusutan untuk mendukung kelancaran aktivitas perusahaan.

4) Azas Kecepatan

Guna mendukung kelancaran penyelenggaraan tugas dan fungsi satuan kerja atau satuan organisasi, semua kegiatan persuratan dan kearsipan harus dapat diselesaikan tepat waktu.

II.4.3 Tulisan Dinas

II.4.3.1 Pengertian Tulisan Dinas

Tulisan dinas adalah semua bentuk naskah yang diterima dan dikeluarkan oleh pejabat berwenang di lingkungan PT (Persero) Angkasa Pura I dalam rangka pelaksanaan tugas kedinasan pada bidangnya masing-masing.

II.4.3.2 Jenis Tulisan Dinas

Tulisan dinas di dalam PT (Persero) Angkasa Pura I Bandara Juanda dibagi

menjadi 2 (dua) jenis, yaitu [SAP, 2003]:

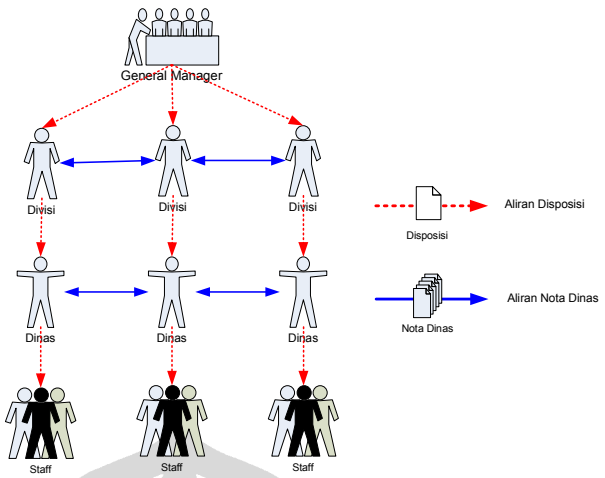
(1) Nota Dinas

Yang dimaksud dengan nota dinas adalah surat kedinasan (formal) yang dibuat dan dikirm antar pejabat dalam satu struktur organisasi. Nota dinas dibuat hanya untuk keperluan internal organisasi. Dalam lingkup Bandara Juanda, maka penggunaan nota dinas hanya dibuat dan dikirim oleh pegawai yang memangku jabatan struktural di Bandara Juanda, yaitu manager yang mengepalai divisi dan asisten manager yang memimpin dinas. Nota dinas hanya dapat dikirim antar pejabat yang memiliki tingkatan / jabatan yang sama secara horisontal (setingkat) di dalam struktur organisasi. Susunan nota dinas dibagi menjadi (tiga) bagian, yaitu: kepala surat, isi surat, dan penutup.

(2) Disposisi

Disposisi adalah arahan, petunjuk, saran, dan tanggapan yang diambil pimpinan atas nota dinas. Disposisi diberikan oleh pejabat ke pejabat / staff lain yang memiliki tingkatan / jabatan lebih rendah dalam struktur organisasi (secara vertikal). Disposisi dituliskan dalam sebuah lembar disposisi, yaitu suatu lembaran yang dipergunakan untuk memberikan arahan, petunjuk, saran, dan tanggapan yang diambil pimpinan atas suatu nota dinas. Disposisi hanya dapat diberikan dari pimpinan ke bawahan yang terdapat dalam satu divisi yang dipimpinnya. Khusus untuk pimpinan utama (general manager) dapat memberikan disposisi ke seluruh divisi yang berada di bawah tanggung jawabnya.

Gambar aliran nota dinas dan disposisi dapat dilihat sebagai berikut:



Gambar II-11 Aliran Nota Dinas dan Lembar Disposisi

II.4.4 Pemusnahan Tulisan Dinas

Proses pemusnahan tulisan dinas diatur sebagai berikut [SAP, 2003]:

- Menseleksi arsip dari duplikasi yang berlebihan;
- Pemusnahan arsip dilakukan secara total, sehingga tidak dapat lagi dikenali isi maupun bentuknya;
- Pemusnahan arsip yang sudah tidak bernilai guna hanya dilakukan oleh administrasi pengguna setelah mendapat ijin dengan unit-unit pemilik atau pencipta arsip;
- Arsip yang dapat dimusnahkan adalah arsip yang sudah tidak bernilai guna atau arsip yang sudah disimpan dalam jangka waktu minimal 3 (tiga) tahun.

II.5 KAJIAN SISTEM APLIKASI PERSURATAN

Sistem aplikasi persuratan sudah umum digunakan untuk mendukung kinerja sebuah perusahaan. Umumnya selain untuk tujuan korespondensi, aplikasi persuratan juga digunakan sebagai sarana untuk berkomunikasi dan berkoordinasi antar unit bisnis yang ada di dalam organisasi. Proses ini menjadi penting apabila organisasi tersebut memiliki struktur organisasi yang gemuk, dimana banyak unit bisnis yang terdapat di dalamnya.

Aplikasi persuratan yang lazim digunakan adalah menggunakan teknologi web *mail*. Dengan teknologi ini unit bisnis atau staff dapat saling berkomunikasi dengan berkirim surat. Namun terdapat kelemahan dalam teknologi web *mail* ini. Teknologi ini hanya mampu mengirim surat dengan cepat dan tepat sasaran, namun tidak tersedia sarana untuk melacak *progress* sebuah surat yang dikirimkan. Proses pelacakan ini penting mengingat seseorang yang mengirim sebuah surat dan ingin mengetahui jawaban dari surat tersebut tidak dapat memantau status surat yang dikirim. Dengan memiliki kemampuan untuk memantau progress sebuah surat, seseorang dapat mengingatkan pihak / staff lain bahwa ada surat yang harus dijawab. Selain itu sarana ini juga dapat berfungsi untuk meningkatkan pengawasan dalam bekerja.