BAB II

LANDASAN TEORI

2.1 Pengertian Sistem

Terdapat dua kelompok pendekatan didalam mendefinisikan sistem, yaitu yang menekankan pada prosedurnya dan yang menekankan pada komponen atau elemennya.

Pendekatan sistem yang lebih menekankan pada prosedur mendefinisikan sistem sebagai berikut ini:

“ Suatu sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu ”.

Menurut Jerry Fitzgerald, Ardra F. Fitzgerald dan Warren D. Stallings, Jr., mendefinisikan prosedur sebagai berikut :

” Suatu prosedur adalah urut-urutan yang tepat dari tahapan-tahapan instruksi yang menerangkan Apa yang harus dikerjakan, Siapa yang mengerjakannya , kapan dikerjakan dan Bagaimana mengerjakannya”.

Pendekatan sistem yang lebih menekankan pada elemen atau komponennya mendefiniskan sistem sebagai berikut ini :

“Sistem adalah kumpulan dari elemen - elemen yang berinteraksi untuk mencapai suatu tujuan tertentu”.

Pendekatan sistem yang merupakan kumpulan elemen-elemen atau komponen-komponen atau subsistem-subsistem merupakan definisi yang lebih luas.

2.1.1 Elemen Sistem

Syarat-syarat yang harus dimiliki oleh suatu sistem, yaitu:

1. Sistem harus dibentuk untuk menyelesaikan suatu tujuan.

2. Elemen sistem harus mempunyai rencana yang ditetapkan.

3. Adanya hubungan diantara elemen sistem.

4. Unsur dasar dari proses (arus informasi, energi, dan material) lebih penting daripada elemen sistem.

Dengan definisi di atas kita bisa menggambarkan system dengan menentuKan bagian – bagian dan cirri-ciri system dari tujuan yang harus di capai .

Sistem memiliki ciri-ciri yang secara umun terdiri dari:

1. Tujuan sistem merupakan target atau sasaran akhir yang di capai oleh suatu sistem .

2. Batas system yang garis abtraksi yang memisahkan antara sub system dengan Lingkungannya.

3. Sub sistem merupakan komponen atau bagian dari suatu system.

4. Hubungan sistem adalah hubungan yang terjadi antara sub sub–sub system lainnya setingkat.

5. *Input-Proses-Output* merupakan tiga komponen dari fungsi atau sub sistem.

2.1.2 Karakteristik Sistem

Terdapat beberapa karakteristik tertentu dari sistem, yaitu mempunyai komponen (*components*), batas sistem (*boundary*), lingkungan luar sistem **(***environments*), penghubung (*interface*), masukan(*input*), keluaran (*output*), pengolah (*proses*), dan sasaran suatu tujuan (*goal*). Adapun penjelasan dari karakteristik suatu sistem yaitu :

a. Komponen sistem **(***Components***)**

   Bagian sistem yang saling berinteraksi dan membentuk satu kesatuan komponen atau elemen sistem dapat berupa subsistem atau beberapa bagian sistem.

b. Batas sistem **(***Boundary***)**

   Daerah yang membatasi antara suatu sistem dengan lingkungannya atau dengan sistem lainnya. Batas sistem inilah yang membuat sistem dipandang sebagai satu kesatuan.

c. Lingkungan luar sistem **(***Environments***)**

   Segala sesuatu yang berada di luar sistem yang mempengaruhi sistem. Lingkungan luar sistem dapat bersifat menguntungkan sistem atau merugikan sistem.

d. Penghubung Sistem **(***Interface***)**

   Merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Penghubung inilah yang menyebabkan beberapa subsistem berintegrasi dan membentuk satu kesatuan.

e. Masukan Sistem **(***Input***)**

   Sesuatu yang dimasukkan ke dalam sistem yang berasal dari lingkungan.

f. Keluaran Sistem **(***Output***)**

Suatu hasil dari proses pengolahan sistem yang dikeluarkan ke lingkungan.

g. Pengolah Sistem **(***Proses***)**

Bagian dari sistem yang mengubah masukan (*input*) menjadi keluaran (*output*).

h Sasaran Sistem **(***Objectives***)** atau Tujuan **(***Goal***)**

Sasaran sistem adalah sesuatu yang menyebabkan mengapa sistem itu dibuat atau ada. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya.

2.1.3 Klasifikasikan Sistem

Sistem dapat dikelompokkan atau di klasifikasikan menjadi beberapa sudut pandang diantaranya adalah sebagai berikut :

1. Sistem di klasifikasikan sebagai sistem abstrak *(abstract sistem)* dan sistem fisik *(physical sistem)*, sistem abstrak terbentuk dari gagasan-gagasan atau konsep-konsep saling berkaitan. Sistem fisik didefinisikan sebagai suatu sistem yang unsur-unsur pendukungnya secara fisik dapat dilihat atau dirasakan dan memiliki keterkaitan antara satu unsur dengan unsur lainnya.
2. Sistem diklasifikasikan sebagai sistem alamiah dan sistem buatan manusia *(human made sistem)*. Yaitu sistem alamiah merupakan sistem yang terjadi melalui proses alam, tidak dibuat manusia, sedangkan sistem buatan manusia merupakan sistem yang dirancang oleh manusia, sistem manusia yang melibatkan interaksi manusia dengan mesin (*Komputer*) disebut *human machine sistem*.

3. Sistem diklasifikasikan sebagai sistem tertentu *(deterministic sistem)* dan sistem tak tentu *(probabilistic sistem)*, sistem tertentu beroprasi dengan tingkah laku yang sudah diprediksi, Interaksi diantara bagian-bagian nya dapat dideteksi dengan pasti, sehingga keluaran dari sistem dapat diramalkan. Sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.
4. Dan yang terakhir adalah klasifikasi sistem sebagai sistem tertutup (*closed sistem)* dan sistem terbuka *(open sistem)*, dimana sistem tertutup tidak terpengaruh oleh lingkungan lainnya, sistem ini bekerja secara otomatis tanpa ada campur tangan dari pihak luar. Sebaliknya sistem terbuka merupakan sistem yang berhubungan dan terpengaruh dengan lingkungan luar.

2.2 Pengertian Informasi

Informasi adalah hasil sebuah pengolahan data dalam suatu bentuk yang lebih berguna dan lebih berarti bagi penerimanya yang menggambarkan suatu kejadian-kejadian yang nyata yang digunakan untuk pengambilan keputusan. Informasi merupakan data yang telah diklasifikasikan atau diolah atau diinterpretasi untuk digunakan dalam proses pengambilan keputusan.

2.3 Pengertian Sistem Informasi

Informasi dapat diperoleh dari sistem informasi (*Information system*) atau disebut juga dengan *processing system* atau *information processing system* atau *information generating system*.

Sistem informasi sendiri memiliki sejumlah komponen tertentu. Seperti yang di kemukakan oleh Robert dan Donald Symanzky, bahwa sistem informasi terdiri dari beberapa komponen yang berbeda yaitu, manusia, data, *hardware*, dan *software*. Sebagai suatu sistem, setiap komponen tersebut berinteraksi satu dengan lainnya dan membentuk satu kesatuan untuk mencapai sasarannya.

2.4 Metode Pendekatan Sistem

Metode Pendekatan Sistem yang diterapkan dalam tahap analisis system E - Procurement menggunakan pendekatan system dengan orientasi objek, dimana orientasi objek secara deskripsi dan definitive dejelaskan pada subbab dibawah ini.

2.4.1 Metode Pendekatan Sistem Berorientasi Objek

Metode beorientasi objek mulai berkembang ketika Grady Booch pada tahun 80 an mempublikasikan suatu paper bagaimana melakukan perancangan untuk bahasa ADA namun memberi judul paper tersebut dengan : *Object-Oriented Desig*n. Selanjutnya ide tersebut terus ia kembangkan sampai tahun 90 an.

Pada tahun 1991 Peter Coad dan Yourdon memperkenalkan metode berorientasi objek yang lebih sederhana dibandingkan Booch. Metode ini menjadi cepat populer

karena mendukung layanan-layanan yang terdapat pada C++. Pada waktu itu C++ merupakan bahasa pemrograman berorientasi objek yang paling populer .

Pada tahun 1991 juga Rumbaugh memperkenalkan Object Modelling Technique (OMT). Pendekatan yang digunakan tidak jauh berbeda dengan pendekatan yang digunakan Coad Yourdon namun dengan notasi yang berbeda. OMT tidak hanya sepenuhnya berbasis pada *data driven* tapi juga memisahkan proses dari data dengan penggunan *data flow diagram* yang terpisah dengan diagram kelas. OMT juga menggunakan notasi *state transition diagram* untuk memodelkan aspek dinamis sistem.

Pada tahun 1994 Ivar Jacobson memperkenalkan konsep *use case* dan *object oriented software engineering.*

Pada tahun 1994 itu juga yaitu bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, mempelopori usaha untuk penyatuan notasi pendekatan berorientasi objek. Pada tahun 1995 dihasilkan *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh **Object Management Group** (OMG – http://www.omg.org).

Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

2.4.2 Keunggulan Pendekatan Sistem Berorientasi Objek

2.4.2.1 Menaikkan tingkat keterpakaian kembali (*reusability*)

Perangkat lunak bersifat dinamis. Hal ini disebabkan kebutuhan pengguna berubah dengan cepat. Perkembangan teknologi informasi dan kebutuhan akan pengolahan informasi itu memaksa setiap organisasi memperbarui sistemnya. Dengan demikian perangkat lunak harus dibangun dengan *reusability* tinggi. Metode yang mendukung *reusability* tersebut adalah metode beroientasi objek.

2.4.2.2 Menghilangkan kompleksitas transisi antar tahap pada pengembangan perangkat lunak

Pada pendekatan konvensional (tertruktur), notasi yang digunakan pada tahap analisis, perancangan dan tahap lainnya berbeda-beda. Hal ini menyebabkan transisi antar tahap pengembangan menjadi kompleks. Pada pendekatan berorientasi objek notasi yang digunakan pada tahap analisis, peanccangan dan implementasi relatif sama.

2.4.3 Konsep dalam Metode Beorientasi Objek

2.4.3.1 Objek

Objek adalah suatu kombinasi antara data dan aspek lojik yang merepresentasikan suatu entitas dunia nyata. Contoh : mobil toyota kijang adalah suatu objek, maka yang menjadi data adalah warna, pembuat, harga, konsumsi bahan bakar, mekanisme pengereman, dll. Sedangkan yang menjadi aspek lojiknya adalah menampilkan jarak tempuh, berhenti, bejalan, belok, dll.

2.4.3.2 Kelas

Pada sistem berorientasi objek, kelas didefinisikan sebagai himpunan objek yang memiliki struktur umum dan perilaku umum yang sama [BAH99]. Sedangkan objek dapat dikatakan sebagai instansiasi suatu kelas. Sebagai contoh mobil adalah suatu kelas, maka mobil Toyota kijang, mobil Proton, dll adalah objek.

2.4.3.3 Atribut

Atribut merepresentasikan karakteristik atau keadaan objek. Pada contoh kasus di atas, sebuah mobil dapat memiliki atribut warna, harga, dan pembuat. Pada tataran implementasi, warna dapat direpresentasikan sebagai suatu *string* (domain nilainya misalnya : merah, biru, kuning, dll). Harga dapat berupa bilangan floating point atau bilangan integer. Sedangkan pembuat dapat bertipe struktur yang terdiri dari nama, identitas korporat, dll.

2.4.3.4 Metode

Metode adalah suatu fungsi atau prosedur yang didefinisikan untuk dapat mengakses keadaan internal suatu objek dari suatu kelas. Tiap fungsi atau prosedur mendefinisikan dan mendeskripsikan perilaku khusus suatu objek. Sebagai contoh: kelas Pegawai memiliki metode Hitung Gaji. Metode sebenarnya merupakan antarmuka yang disediakan untuk dapat memanfaatkan perilaku objek tersebut. Sebagai contoh : jika diinginkan dilakukannya perhitungan gaji, maka *message* ‘Hitung Gaji” harus dikirimkan ke objek Pegawai.

2.4.3.5 Messag*e*

Message pada dasarnya adalah pemanggilan fungsi. Namun *message* berbeda dari pemanggilan subrutin. Dengan *message* yang sama dua objek berbeda dapat melakukan operasi yang berbeda pula. Konsep ini dikenal sebagai *Polymorphism.*

2.4.3.6 Enkapsulasi

Enkapsulasi memadukan karakteristik unit di dalam suatu objek (data dan metode). Konsep ini bertujuan untuk menyembunyikan informasi dan karakteristik objek. Objek dapat dimanfaatkan hanya dengan cara memanggil metode yang dimiliki objek tersebut.

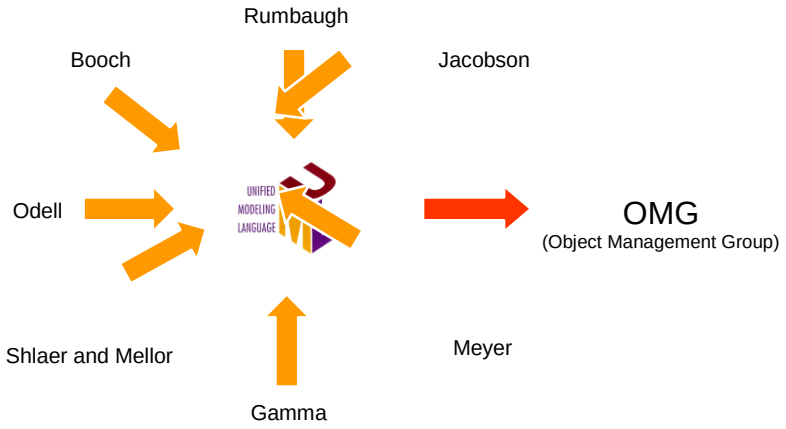
2.4.4 Alat Bantu Analisis (UML – Unified Modelling Languange)

2.4.4.1 Pengertian UML

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

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering).



Gambar 2.1: Hubungan Permodelan Berorientasi Objek

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh **Object Management Group** (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru

adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999 [7] [8] [9]. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek

2.4.4.2 Konsepsi Dasar UML

*D*ari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar UML bisa kita rangkumkan dalam gambar dibawah.

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

Gambar 2.2 Rangkuman Konsepsi UML

Abstraksi konsep dasar UML yang terdiri dari *structural classification*, *dynamic behavior*, dan *model management*, bisa kita pahami dengan mudah apabila kita melihat gambar diatas dari *Diagrams*. *Main concepts* bisa kita pandang sebagai term yang akan muncul pada saat kita membuat diagram. Dan view adalah kategori dari diagaram tersebut.

Seperti juga tercantum pada gambar diatas UML mendefinisikan diagram-diagram sebagai berikut:

- *use case diagram*
- *class diagram*
- *statechart diagram*
- *activity diagram*
- *sequence diagram*
- *collaboration diagram*
- *component diagram*
- *deployment diagram*

2.4.4.3 Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor

dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya.

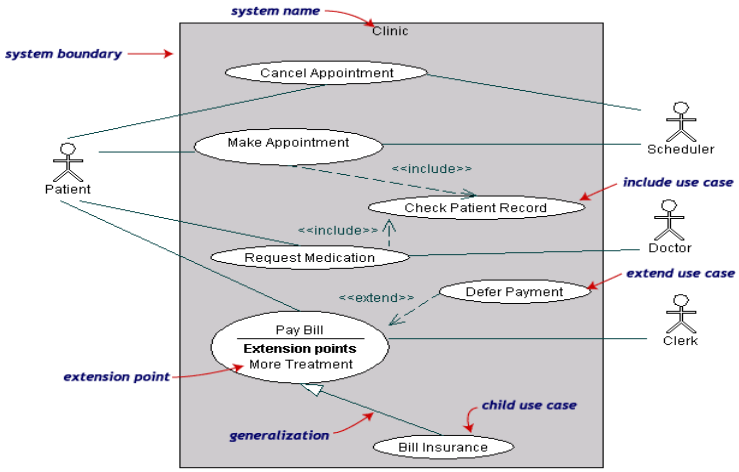
Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

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

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



Gambar 2.3 Contoh Use Case Diagram

2.4.4.4 Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

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

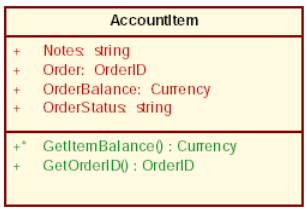
Class memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

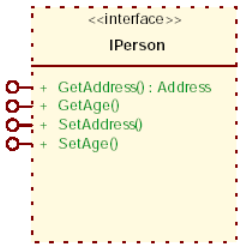
Atribut dan metoda dapat memiliki salah satu sifat berikut :

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan

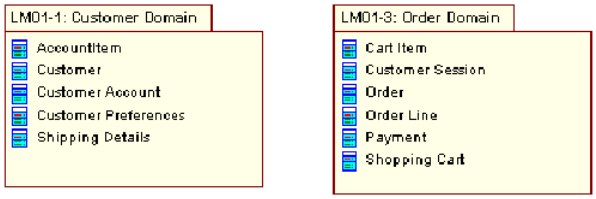
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
- *Public*, dapat dipanggil oleh siapa saja



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



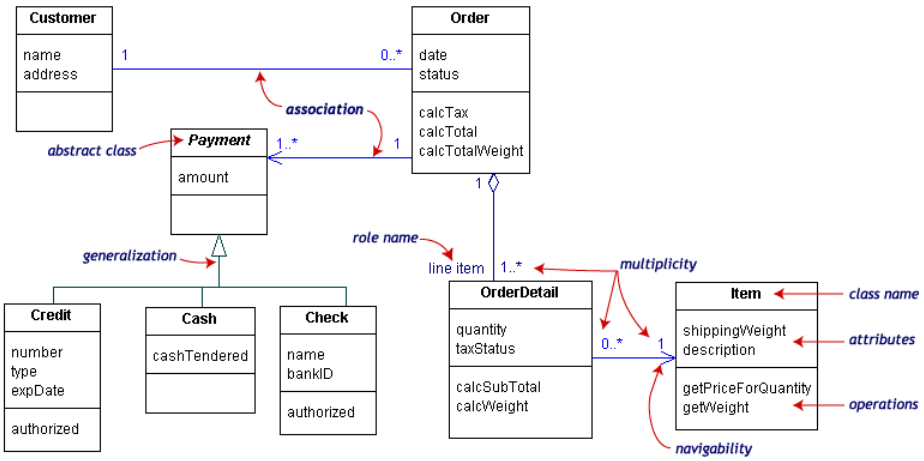
Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.



Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.



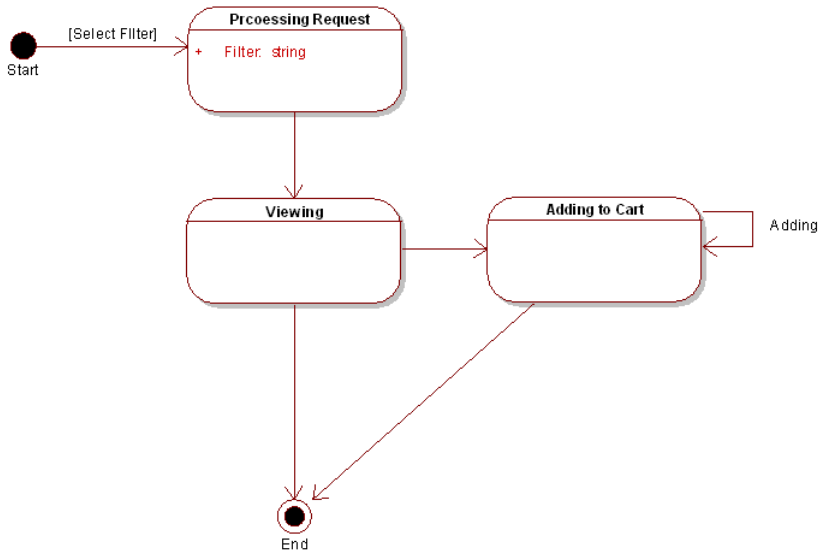
Gambar 2. 4 Contoh Class Diagram

2.4.4.5 Statechart Diagram

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

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.



Gambar 2.5 : Contoh Statechart Diagram

2.4.4.6 Activity Diagram

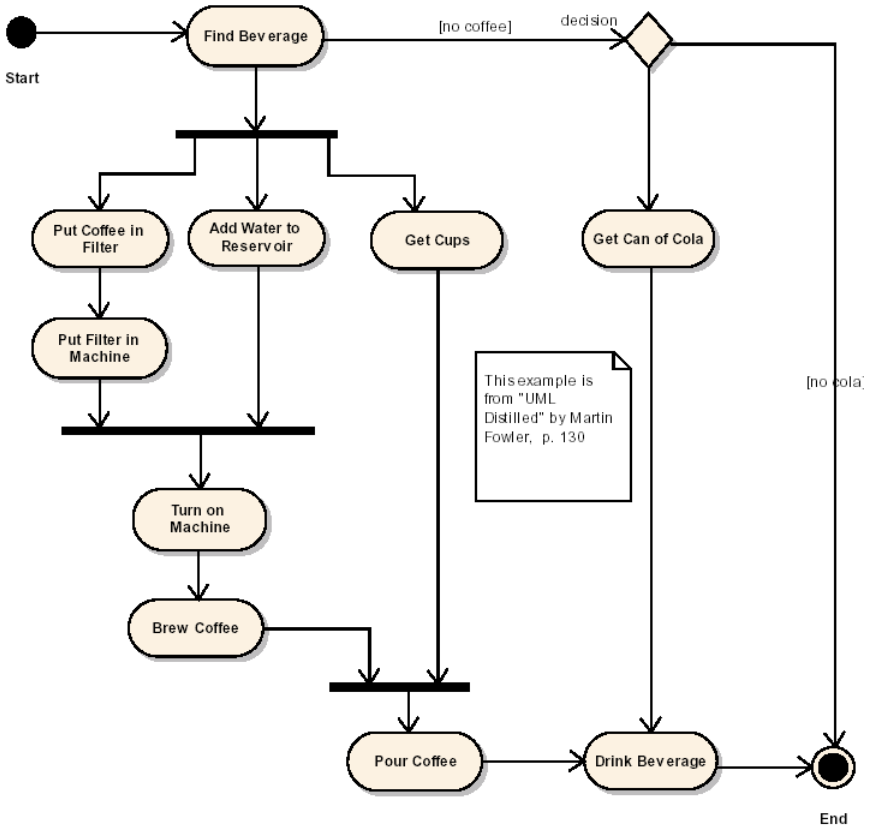
Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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

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

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar 2.6 Contoh Activity Diagram

2.4.4.7 Sequence Diagram

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

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut,

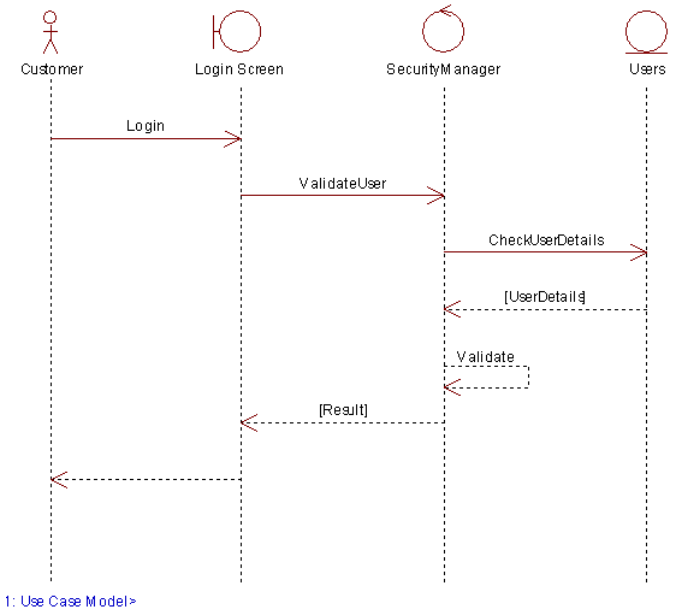
proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal.

Message digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*.

Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.

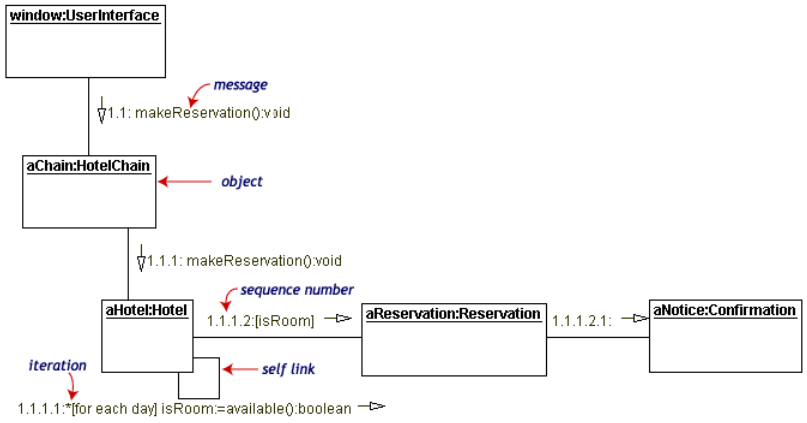


Gambar 2.7 : Contoh Sequence Diagram

2.4.4.8 Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*.

Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.



Gambar 2.8 Contoh Collaboration Diagram

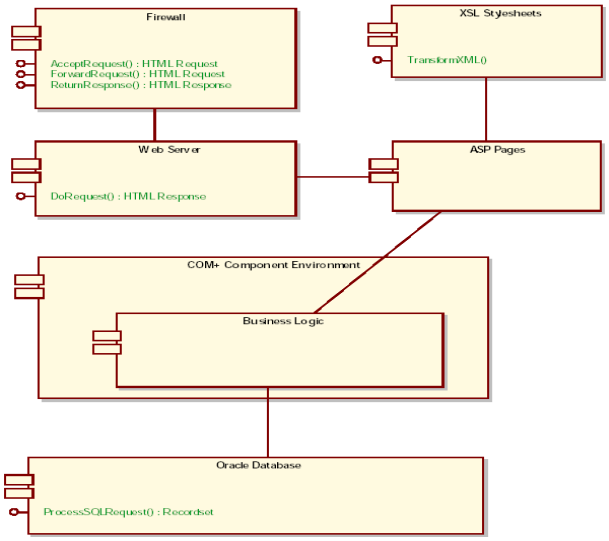
2.4.4.9 Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya.

Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari

beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil.

Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.



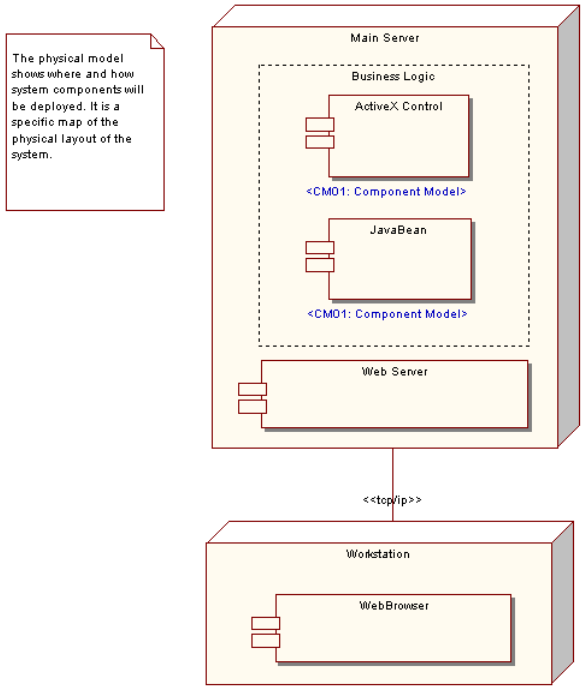
Gambar 2.9 : Contoh Component Diagram

2.4.4.10 Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak

(pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal

Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.



Gambar 2.10 : Contoh Deployment Diagram

2.5 Sekilas Mengenal Rational Rose

Rational Rose adalah tools pemodelan visual untuk pengembangan system berbasis objek yang handal untuk digunakan sebagai bantuan bagi para pengembang

dalam melakukan analisis dan perancangan system. *Rational rose* mendukung permodelan bisnis yang membantu para pengembang memahami system secara komprehensif. Ia juga membantu analisis system dengan cara pengembang membuat diagram use case untuk melihat fungsionalitas system secara keseluruhan sesuai dengan harapan dan keinginan pengguna. Kemudian, ia juga menuntut pengembang untuk mengambangkan Interaction Diagram untuk melihat bagaimana objek-objek saling bekerjasama dalam menyediakan fungsionalitas yang diperlukan.

Dalam *Rational rose*, pemodelan adalah cara melihat system dari berbagai sudut pandang. Ia mencakup semua diagram yang dikenal dalam UML, actor-aktor yang terlibat dalam system, use-case, objek-objek, kelas-kelas, komponen-komponen, serta simpul-simpul penyebaran. Model juga mendeskripsikan rincian yang diperlukan system dan bagaimana ia akan bekerja, sehingga para pengembang dapat menggunakan model itu sebagai blue print untuk system yang akan dikembangkan.

Dalam Rational Rose mendukung dalam permodelan diantaranya seperti di bawah ini :

- **[Membuat Use Case Diagram]**
- **[Membuat Class Diagram]**
- **[Membuat Sequence Diagram]**
- **[Membuat Collaboration Diagram]**
- **[Membuat State Diagram]**

2.6 Deskripsi Kerja Bagian Pekerjaan

Ruang lingkup Bagian Perencanaan pada PT. PLN (Persero) J&P Unit Produksi Citarum ruang lingkupnya secara umum yaitu dideskripsikan dalam penjelasan dibawah ini :

1) Membuat rencana - rencana tekhnik untuk pengembangan peralatan pembangkit dan penyaluran.
2) Membuat rencana - rencana kerjauntuk pengembangan pembangkit dan penyaluran.
3) Menyusun rencana anggaran rehabilitas dan investasi untuk pengembangan peralatan pembangkit dan penyaluran.
4) Membuat evaluasi hasil pelaksanaan rencana pengembangan peralatan, pembangkit dab penyaluran.
5) Membuat laporan berkala dalam bidangnya

2.7 E-Procurement (Sistem Lelang On – Line) PT PLN Jasa dan Produksi U.P. Citarum

Good Corporate Governance (GCG) atau dikenal dengan tata kelola Perusahaan yang baik dalam aspek bisnis dan pengelolaan perusahaan pada semua jajaran perusahaan, PLN menyusun tatakelola Teknologi Informasi dalam lingkup bisnis dan pelaksanaan pengelolaan perusahaan. Dukungan Teknologi Informasi dapat meningkatkan kapabilitas perusahaan dalam memberikan kontribusi bagi penciptaan nilai tambah, serta mencapai efektifitas dan efisiensi. Aspek kunci dari

prinsip GCG meliputi adil, responsibilitas, transparansi, independensi, akuntabilitas, keselarasan dan kewajaran serta tanggung jawab untuk mencapai tujuan perusahaan.

Dengan Panduan Kebijakan Tata Kelola Teknologi Informasi BUMN (IT Governanve), seluruh BUMN diminta untuk melaksanakan GCG pada setiap aspek bisnis dan juga pengelolaan perusahaan pada semua jajarannya.

Hal ini dapat mencerminkan dengan sangat baik suatu proses pengambilan keputusan juga leadership dalam penyelenggaran tata kelola Teknologi Informasi.

E-Procurement PLN (eProc) sebagai salah satu aplikasi yang merupakan implementasi dari IT Governance yang mendukung GCG. Terwujudnya aplikasi tersebut merupakan hasil kebijakan Manajemen PT. PLN (Persero) tahun 2000 terkait dengan Informasi Stok Material PLN, Penyusunan HPS, dan Monitoring Pergerakan Material. Sedangkan hasil Amanat RUPS tahun 2003 menetapkan agar PLN mengoptimalkan eProc yang sudah dikembangkan untuk tercapainya harga pembelian yang optimal dan tercapainya inventoru PLN yang efisien. Proses pengadaan secara manual dapat mengakibatkan sulitnya informasi mengenai harga satuan khusus di internal PLN, perlakuan yang tidak sama kepada Calon Penyedia Barang/Jasa (CPBJ), dan lemahnya pertanggung jawaban terhadap proses pegadaan sehingga mengakibatkan resiko di kemudian hari.

Terkait tidak adanya informasi stok barang di gudang, mengakibatkan sulitnya mencapai sasaran stok optimal. Aplikasi eProc mampu membawa manfaat bagi Perusahaan yakni adanya standardisasi proses pengadaan, terwujudnya transparansi dan efisiensi pengadaan yang lebih baik, tersedianya informasi harga satuan khusus di

internal PLN, serta mendukung pertanggung-jawaban proses pengadaan. Beberapa kendala dalam implementasi eProc dapat teratasi dengan adanya komitmen pada seluruh jajaran manajemen dan pelaksana pengadaan untuk menggunakan eProc sebagai sarana proses pengadaan barang/jasa di PLN, dan melakukan sosialisasi secara bertahap serta melakukan penyederhanaan proses pengadaan, memanfaatkan teknologi dan pengembangan aplikasi yang bersifat fleksibel.

Ruang lingkup eProc PLN dibagi menjadi 3 (tiga) kebutuhan utama, antara lain : Cataloging Information System, Supply Chain Management (SCM) System, Portal e-Proc PLN. Pada kebutuhan Cataloging Information merupakan pemenuhan kebutuhan atas terbentuknya database katalog material (MDU, sparepart, SCADA, Pembangkit, Bahan Bakar, dll); sharing informasi dari persediaan, bursa, harga satuan, HPS, daftar pemasok; menyusun daftar rencana pengadaan material. Pada kebutuhan SCM System merupakan perwujudan dari pengadaan material melalui bursa antar Unit PLN, pengadaan barang/jasa melalui e-bidding dan e-auction. Sedangkan sarana portal eProc merupakan usaha untuk memberikan hosting portal kepada pihak lain yang inign menggunakan jasa layanan pengadaan barang/jasa, memberikan layanan promosi/iklan melalui portal eProc, dan menjadi pusat penyedia informasi.

Selama tahun 2005-2008, eProc mencatat saving sebesar 4,56% terhadap realisasi Harga Perkiraan Sendiri (HPS), yakni Rp.249,40 Milyar dan pengehematan sebesar Rp.1,6 Trilyun dari Realisasi Rencana Anggaran Biaya (RAB) terhadap Total RAB. Sedangkan total pengadaan yang telah direalisasikan melalui e-Proc selama 4

tahun tersebut adalah sebanyak 3352 pengadaan dari total rencana sebanyak 5071 pengadaan atau 66,1%. Jumlah realisasi pengadaan yang dilakukan melalui e-Proc terhadap rencana pengadaan cenderung meningkat dari tahun 2005 hingga tahun 2008 dengan rata-rata pertumbuhan realisasi pengadaan sebesar 63.91% setiap tahunnya. Sedangkan pada tahun 2007 sampai dengan tahun 2008 terjadi penpenurunan pertumbuhan sebesar 5,89%. Sedangkan pada tahun 2008, e-Proc berhasil mencatat saving sebesar Rp.90,80 Milyar atau sebesar 4.91% berdasarkan Perolehan HPS terhadap Realisasi HPS dan sebesar Rp.457,9 Milyar atau sebesar 8,06% terhadap Realisasi RAB.

Penekanan terhadap HPS tersebut dapat diraih dengan pelaksanaan e-Auction pada pengadaan melalui pelelangan umum, seleksi umum, dan lainnya. e-Auction adalah teknik penyampaian penawaran harga melalui eProc PLN dimana harga yang sudah disampaikan tersebut dikompetisikan di antara CPBJ selama selang waktu tawar menawar yang ditentukan. Aplikasi eProc PLN merupakan representasi dari Kepres 080 tahun 2003 Tentang Pedoman Pengadaan Barang/Jasa Pemerintah, sehingga implementasi eProc nanti dapat dijadikan acuan (benchmark) bagi Instansi Pemerintah atau BUMN lainnya.

Dalam perkembangannya, eProc menjadi aplikasi yang mampu mendukung pelaksanaan perwujudan kinerja yang lebih baik kalangan internal PLN. Penghematan Realisasi HPS terhadap Total RAB dan Total HPS merupakan salah satu tolak ukur keberhasilan aplikasi tersebut. Dari sisi publikasi pengadaan di media cetak, eProc mampu memberikan kontribusi yang baik dengan memberikan

penghematan sebesar Rp.6,6 Milyar per tahun dengan rata-rata pengadaan per tahun adalah 660 buah dan rata-rata biaya publikasi per pengadaan adalah Rp.10 Juta.

Dengan adanya pertumbuhan user pada tahun 2008, yakni lebih dari 12000 user eksternal (calon penyedia barang/jasa), dan 4000 diantaranya yang masih dilakukan verifikasi, terdapat peluang bisnis yang mampu menjadikan sistem dan aplikasi tersebut sebagai salah satu profit center dari PT. PLN (Persero). Di antaranya adalah :

· Iklan tetap di portal E-Procurement (corporate advertising or marketing) dengan prediksi pertumbuhan user adalah 20% per tahun hingga tahun 2011;
· SMS Mobile untuk penyedia informasi terkait pengumuman pengadaan, proses pengadaan, hingga penunjukan pemenang;
· Bekerja sama dengan situs pencarian informasi (dengan jumlah hit 20.000 per hari dengan prediksi pertumbuhan hit adalah 10% per tahun hingga 2011), serta mewujudkan konsep marketplace.

Jika aplikasi eProc sudah bisa menjadi salah satu profit center dari PT.PLN (Persero), maka tidak menutup (Persero), maka tidak menutup kemungkinan bisa dikembangkan untuk menjadi Unit Bisnis atau Anak Perusahaan dari PT.PLN (Persero) yang mampu mengelola seluru proses bisnis e-commerce internal dan beberapa bagian bisnis eksternal.