Prototipe Perangkat Lunak Studi dan Implementasi Sistem Antrian

Sukenda, Roosaleh Laksono T.
Jurusan Teknik Informatika
Universitas Widyatama
Jl. Cikutra 204 A Bandung

Abstract

Queueing is one of the way to create discipline. In the queueinng, there are customers who demands service and owner of the service who provide those service. There is conflict of interest between those two because the customers do not want to queue too long and the service owner who wants to serve as many customers as possible. Hence, the service owner must be able to provide conditions such that customer queueing is not too long. To create good queue the observation of waiting time for customer, service time, and queueing time need to be closely examined. With those examination, the condition that satisfactory to cusomer and service owner with minimal extra cost. Based on those requirements, software called pro-que is engineered to create optimum condition for queueing. This system is built based on the object oriented method of software engineering known as OMT (Object Modelling Technique) by Rumbaugh.

***Keyword** : queue, customer satisfaction, queue time, service time, queue system time, OMT.*

1. Pendahuluan

Antrian merupakan cara yang dibangun orang dalam berbagai kegiatan atau aktivitas agar tercipta suasana yang tertib. Hampir semua kegiatan memerlukan antrian misalnya pada pembelian tiket kereta api, pembelian tiket pesawat terbang, pembayaran atas pembelian barang-barang yang dibeli di supermarket, dan sebagainya.

Dalam sistem antrian, ada pihak-pihak yang berperan dalam menjalankan sistem antrian tersebut. Pelanggan merupakan entitas yang bertindak sebagai pengantri dalam sistem antrian. Server (pelayan) merupakan entitas yang bertindak sebagai pelayan dalam sistem antrian. Keduanya sangat penting dalam sistem antrian, dikarenakan tidak bisa terbentuk jika dalam suatu antrian tidak ada pelanggan dan server. Pelanggan menghendaki waktu tunggu dalam antrian seminimal mungkin sedangkan pemilik pelayanan menghendaki pelanggan yang masuk antrian sebanyak mungkin sehingga timbul kontradiksi antara pelanggan dan pemilik pelayanan. Kontradiksi ini timbul karena pelanggan merasa enggan berlama-lama di waktu tunggu dalam antrian, disisi lain pemilik pelayanan tidak menginginkan kehilangan pelanggan dan tidak menginginkan biaya untuk membuka server baru yang terlalu besar. Dalam permasalahan ini, akan melahirkan persoalan optimasi antara pelanggan dan pemilik pelayanan. Persoalan optimasi tersebut, sudut pandang pelanggan yaitu terlayaninya dengan cukup puas sedangkan sudut pandang pemilik pelayanan tidak mengeluarkan biaya besar untuk membuka server baru.

Untuk mengatasi masalah pencatatan data yang berdasarkan data pengataman secara nyata dan diikuti dengan proses pengolahan data tersebut agar bisa menghasilkan informasi yang cepat dan tepat dibangunlah sebuah perangkat lunak. Pembangunan dan perancangan perangkat lunak untuk mengimplementasikan

sebuah sistem antrian yang berupa prototipe perangkat lunak implementasi antrian. Dalam membangun dan merancang prototipe perangkat lunak implementasi antrian akan mengasumsikan bahwa sebuah kasus sistem antrian yang memiliki satu server dan satu antrian dalam sistem antrian dengan disiplin antrian *FCFS* (*First Come First Serve*).

2. Pembahasan

Pembahasan ini mencoba untuk menguraikan pokok pembahasan dalam membangun *prototipe perangkat lunak studi dan implementasi sistem antrian* dengan menggunakan metode OMT (*Object Modelling Technique*). Pembahasan tersebut dibagi dalam tahap-tahap dalam membangun sistem prototipe perangkat lunak sebagai berikut :

1. Tahap Analisis Sistem
2. Tahap Perancangan Sistem
3. Tahap Implementasi Sistem

2.1. Tahap Analisis Sistem

Tahap analisis sistem dengan metodologi OMT, umumnya menggunakan 3 buah model yaitu model objek, model dinamis, dan model fungsional. Model objek adalah model yang menggambarkan objek dalam sistem dan keterhubungannya, model dinamik adalah model yang menggambarkan interaksi antar objek dalam sistem, dan model fungsional adalah model yang menggambarkan tranformasi data pada sistem.

Pada kesempatan ini dijelaskan ketiga model tersebut berdasarkan hasil pengamatan. Disamping itu, penjelasan tentang deskripsi pemakai dan deskripsi sistem. Sehingga pokok bahasan tahap analisis sebagai berikut :

1. Deskripsi pemakai, menggambarkan calon pemakai dari sistem yang dikembangkan.
2. Deskripsi sistem, menggambarkan secara umum apa yang akan dilakukan oleh sistem dan apa yang tidak dilakukan oleh sistem. Deskripsi ini dilengkapi dengan kamus data untuk menjelaskan kata-kata kunci yang ada pada sistem.
3. Model objek, diagram objek bentuk statis, yang menjelaskan struktur setiap objek serta keterhubungan antar objek.
4. Model dinamis, berisi aktivitas yang terjadi pada sistem apabila terjadi suatu *event*. Bagian ini terdiri dari dua bagian yaitu *event flow* dan diagram status.
5. Model fungsional, menjelaskan diagram data dalam sistem. Bagian ini terdiri dari diagram konteks dan diagram aliran data umum.

2.1.1 Deskripsi Pemakai

Penerapan sistem *pro-que* (pengertian sistem *pro-que* bisa dilihat pada deskripsi sistem) dinamakan *pro-que*. Oleh karena itu, diperlukan suatu tahapan penyesuaian untuk data-data yang masuk ke dalam *pro-que*, ataupun data-data yang keluar dari *pro-que*. Untuk memasukan data-data ke dalam *pro-que* diperlukan seorang

operator. Sedangkan untuk menganalisa hasil pengolahan dari *pro-que* diperlukan seorang manajemen. Dengan demikian, *pro-que* memerlukan entitas luar yaitu operator dan manajemen yang berguna untuk menjalankan *pro-que*. Operator yaitu orang yang diberikan wewenang untuk mengoperasikan *pro-que* sedangkan manajemen yaitu orang yang diberikan wewenang untuk menganalisis sistem antrian yang terjadi. Diagram fungsinya dapat dilihat pada gambar 2.1.

Sehingga pernagkat lunak *pro-que* akan dikembangkan berdasarkan dua kategori pemakai yaitu operator dan manajemen. Tugas seorang operator yang menggunakan *pro-que* yaitu :

1. Memasukan data-data yang diperlukan oleh *pro-que* untuk proses pengolahan berdasarkan konsumen yang masuk antrian, layanan sistem terhadap konsumen, waktu tunggu konsumen, dan berbagai masalah yang berkaitan dengan antrian.
2. Memastikan bahwa data masukan atau keluaran *pro-que* benar.

Adapun tugas seorang manajemen yang menggunakan *pro-que* yaitu :

1. Menganalisis terhadap hasil pengolahan *pro-que* berdasarkan data-data yang dimasukan oleh operator *pro-que*.
2. Menentukan kebijakan terhadap sistem layanan yang didasarkan pada hasil analisis dari pengolahan *pro-que*.

diharapkan manajemen yang menggunakan *pro-que* ini yaitu seorang analis serta pengambil kebijakan yang memiliki pengetahuan tentang antrian yang pada kenyataannya berjalan secara dinamik.

2.1.2 Deskripsi Sistem

Sistem *pro-que* adalah suatu perangkat lunak untuk melakukan pemrosesan terhadap waktu antrian konsumen dan waktu layanan terhadap konsumen supaya lebih efektif. Fungsi dasar implementasi antrian *pro-que* adalah menentukan antrian yang sesuai dengan kerja sistem fisik pada suatu daerah kondisi operasi sistem yang berjalan, dengan kandidat-kandidat antrian yang ada. Antrian sistem dilakukan untuk mengimplementasikan atau bisa juga mengoperasikan *pro-que* dalam layanan terhadap konsumen dengan skala waktu nyata, bisa dilihat pada gambar 2.1.

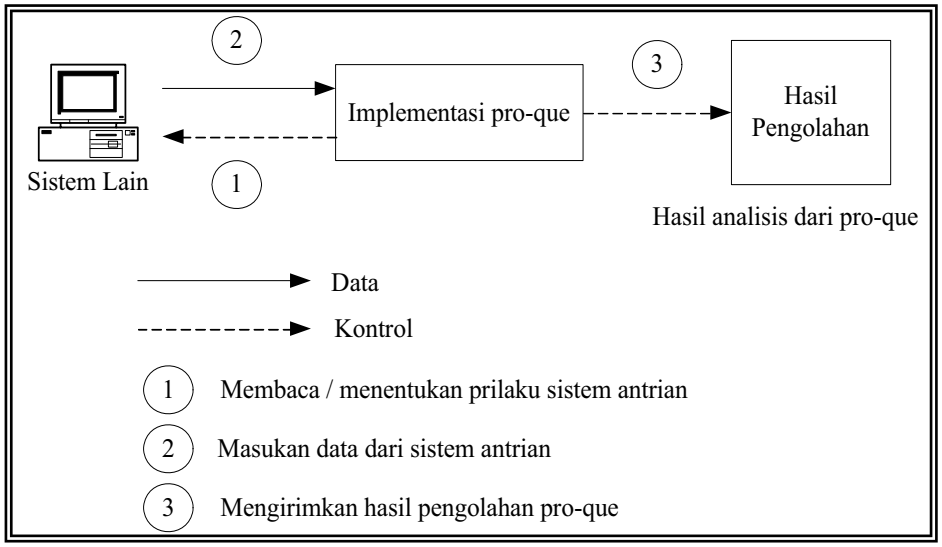
Adapun rumusan masalah yaitu :

1. Seberapa jauh waktu tunggu dalam antrian dan waktu tunggu layanan bisa dikatakan efektif.
2. Berapa kuota antrian agar sistem antrian masih terpenuhi kondisi efektif.
3. Kapan suatu sistem antrian membuka server baru sehingga menimbulkan permasalahan yang akan menggunakan model antrian dengan server lebih dari satu.

Bisa dipenuhi dengan cara sebagai berikut :

1. Untuk memenuhi waktu tunggu dalam antrian dan waktu tunggu layanan yang efektif jika jumlah waktu tunggu dalam antrian dan waktu tunggu layanan oleh server tidak melewati waktu sistem antrian dari mulai buka sampai dengan tutup.

2. Kuota antrian agar sistem antrian memenuhi kondisi efektif jika jumlah konsumen yang berada dalam antrian tidak menghabiskan waktu yang telah ditentukan oleh sistem antrian.
3. Sistem antrian akan membuka server baru jika kondisi berikut terpenuhi :
   - Jumlah waktu tunggu konsumen dalam antrian sudah melewati batas waktu yang ditentukan oleh sistem antrian : waktu buka sampai dengan waktu tutup.
   - Jumlah waktu tunggu layanan sudah melewati batas waktu yang ditentukan oleh sistem antrian : waktu buka sampai dengan waktu tutup.
   - Jumlah konsumen yang menghabiskan waktu sistem antrian yaitu waktu buka sampai dengan waktu tutup

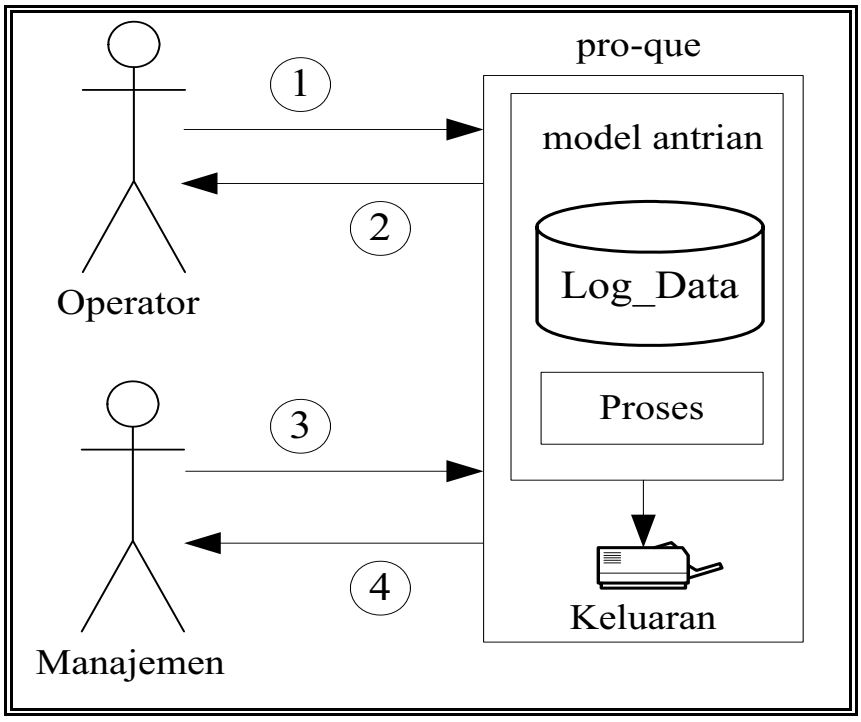


Gambar 2.1 : Diagram Fungsi Dasar Implementasi *pro-que*

Interaksi *pro-que* dengan sistem lain dilakukan dengan membaca data masukan yang didasarkan pada keadaan layanan, antrian, waktu tunggu layanan, waktu tunggu dalam antrian, dan waktu kedatangan dari sistem antrian. Dan membaca masukan yang dilakukan dengan cara memasukan data layanan sistem, konsumen yang masuk antrian, waktu kedatangan konsumen dalam antrian, dan waktu layanan oleh operator (*emulator*). Sedangkan operator sebagai orang yang memasukan data-data antrian ke *pro-que*, lihat gambar 2.2.

Pro-que dengan operasinya dalam waktu nyata dengan sistem lain, menerima masukan data secara sekuensial yang dimasukan ke *pro-que* untuk diolah. Oleh karena itu, diperlukan suatu *emulator* (operator) yang bertugas untuk mengatur jalannya *pro-que* terhadap masukan data dari sistem lain. Hasil dari *pro-que* yang merupakan analisa akan dikirimkan ke *emulator* lain (manajemen) untuk dianalisis. Hasil dari analisis *emulator* akan berpengaruh terhadap sistem lain, sehingga akan terjadi perubahan yang diperlukan untuk mengatasi antrian pada sistem lain supaya lebih efektif, lihat gambar 2.2.

Pro-que akan menempatkan data masukan ke *log_data* yang nantinya akan diproses atau diolah oleh *pro-que* berdasarkan model antrian. Model antrian yang akan diterapkan pada *pro-que* ditetapkan berdasarkan seorang manajemen dari sistem lain. Dalam hal ini perancangan *pro-que* sudah menyiapkan model antrian dengan server lebih dari satu yang dikembangkan dari model antrian yang menggunakan satu server, lihat gambar 2.2.



Gambar 2.2 : Interaksi *pro-que* dengan sistem lain

Keterangan :

⟶ Data

1. bulatan nomor 1 : setting model antrian, masukan data-data, dan setting hasil
2. bulatan nomor 2 : hasil pengolahan
3. bulatan nomor 3 : setting model antrian
4. bulatan nomor 4 : hasil pengolahan

Proses pendeteksian terhadap suatu antrian dilakukan secara sekuensial sesuai dengan kedatangan konsumen dan layanan sistem antrian. Penanganan masalah kedatangan konsumen dan layanan sistem antrian dilakukan oleh *operator*, seseorang yang akan memasukan waktu kedatangan konsumen dan waktu layanan sistem. Pada *pro-que* akan ditangani oleh *scheduler*, program yang menangani masukan data ke *log_data* berdasarkan model antrian yang ditangani oleh *konfigurator* dan selanjutnya dikirimkan ke *korektor* dan *eksekusor*. Selanjutnya *korektor* akan mengirimkan kontrol perintah baca data kontrol dari *log_data* berdasarkan model antrian dan mengirimkan data ke

eksekusor. Sedangkan *eksekusor* memproses data dari *korektor*, dalam hal ini dibutuhkan *grafiktor* untuk menghasilkan grafik dari pengolahan data yang dilakukan oleh *eksekusor*. Dan *eksekusor* membutuhkan *analisator* untuk menghasilkan data analisa yang dibutuhkan oleh pihak manajemen sistem lain.

Karakteristik sistem *pro-que* untuk saat ini yaitu :

- perangkat lunak *pro-que* masih bersifat manual, karena data masih dimasukan oleh seorang operator bukan dilakukan oleh mesin dalam pengontrolan antrian (bukan automatis).
- sedangkan antrian yang digunakan yaitu masih satu antrian, dan servernya masih satu.
- hasil pengolahan data yang diinformasikan ke manajemen masih bersifat primitif, karena masih berdasarkan asumsi-asumsi yang masih sederhana, atau tidak berdasarkan ilmu yang sekarang berkembang, seperti : *fuzy logic*, *neural network* dan sebagainya.

Adapun hal-hal yang menjadi kunci dari perangkat lunak *pro-que* dan sekaligus untuk menerangkan kamus data dan sebagai kandidat *class* dalam analisis perangkat lunak yaitu :

- ***operator*** : entitas yang mewakili hubungan *pro-que* dengan sumber data dari sistem antrian. Dan yang memasukan data-data ke *pro-que*.
- ***scheduler*** : suatu koding program yang memasukan kedatangan konsumen dalam antrian, waktu kedatangan konsumen, waktu layanan sistem, dan waktu keberangkatan.
- ***log_data*** : suatu tempat untuk menyimpan data masukan dari *scheduler* dan *korektor*.
- ***korektor*** : suatu koding program yang membaca data-data dari *log_data* yang telah dimasukan oleh *scheduler* dan menyimpan data hasil pengolahan *korektor* ke *log_data* untuk kebutuhan pengolahan yang dilakukan oleh *eksekusor*.
- ***eksekusor*** : suatu koding program yang mengolah data masukan dari *korektor*. Dan hasil pengolahan dikirimkan ke *grafikan* dan *analisator*.
- ***konfigurator*** : suatu koding program yang menyatakan model antrian yang digunakan pada *pro-que*. Konfigurator ini ditentukan oleh operator atau manajemen sistem lain.
- ***grafikan*** : suatu koding program yang menampilkan grafik dari pengolahan data yang dilakukan oleh *eksekusor*.
- ***analisator*** : suatu koding program yang menentukan hasil dari pengolahan data yang dilakukan oleh *eksekusor* untuk dianalisa kelayakan sistem antrian.

2.1.3 Model Objek

Berdasarkan deskripsi sistem, didapatkan beberapa kelompok objek yaitu :

- Objek pelaksana masukan data, yaitu : *scheduler*, *log_data*, *korektor* dan *konfigurator*.
- Objek pelaksana pengolahan data, yaitu : *eksekusor* dan *korektor*.
- Objek pelaksana tampilan hasil pengolahan, yaitu : *analisator* dan *grafikan*.

Pada model dinamis sesi tentang diagram status, kelompok objek tersebut akan diwakili oleh kelompok status berikut : objek pelaksana masukan data diwakili oleh status konfigurasi, objek pelaksana pengolahan data diwakili oleh status eksekusi, dan objek pelaksana tampilan hasil pengolahan diwakili oleh status tampil. Penunjukan wakil tersebut untuk memudahkan asumsi pengertian status dari *class-class* yang terdapat pada model objek.

Pelaksanaan masukan data dilakukan oleh objek pelaksana masukan data yaitu memasukan nilai-nilai yang dibutuhkan oleh *pro-que*. Dalam hal ini, *scheduler*, memasukan data waktu kedatangan konsumen, waktu layanan sistem, waktu keberangkatan, dan konsumen yang masuk ke sistem antrian. Data masukan selanjutnya dikirim ke *log_data*. Sebelum *scheduler* melaksanakan tugasnya, ada objek *konfigurator* yang mengkonfigurasi model antrian yang diinginkan oleh *scheduler*. Sehingga *scheduler* sudah siap menerima masukan data dari operator berdasarkan konfigurasi model antrian yang dimasukanya. *Korektor* merupakan objek yang melakukan proses terhadap input yang dilakukan oleh *scheduler*. Sehingga hasil yang diharapkan oleh *korektor* yaitu suatu data yang siap untuk dieksekusi oleh *eksekusor*.

Pelaksanaan pengolahan data dilakukan oleh objek pengolahan data. *Eksekusor* mengawali pembacaan *log_data* lewat *korektor*. Dengan penerimaan atau pembacaan terhadap data yang dikirimkan oleh *korektor* maka *eksekusor* melakukan proses yang diharapkan oleh sistem *pro-que*. *Eksekusor* mengirimkan hasil pemrosesan yang telah dilakukannya ke *analisator* dan *grafikan*.

Hasil pengolahan akan ditampilkan untuk kebutuhan analisa bagi manajemen dilakukan oleh objek pelaksana analisa hasil. Pengolahan hasil analisa *pro-que* dilakukan oleh *analisator* berupa nilai-nilai hasil pengolahan dari *eksekusor*. Sedangkan hasil pengolahan *eksekusor* yang berupa grafik dikirimkan ke *grafikan* untuk ditampilkan secara grafik. Adapun *class* objek tahapan analisis bisa dilihat pada tabel 2.1.

Tabel 2.1 : *Class* Objek pada Tahapan Analisis

No	***Class***	**Atribut – Operasi**

1	*Scheduler* : *class* objek yang mempersiapkan tempat penyimpanan, mengisikan dan pengeksekusian data input.	create() //scheduler membuat tempat simpanan di log_data
		open() //schedelur membuka log_data
		append() //scheduler memasukan data ke log_data

2	*Log_data* : class objek yang menyimpan data masukan untuk proses pengolahan.	konsumen_ke : integer //konsumen, time_in : date //waktu masuk antrian, time_service : date //waktu layanan,

No	*Class*	Atribut – Operasi

		time_out : date //waktu keberangkatan
		kon_ke : integer //konsumen,
		time_q : date //waktu tunggu dalam antrian,
		time_ser : date //waktu tunggu layanan
		time_sian : date //waktu tunggu dalam sistem
		read() //baca data dari scheduler atau korektor,
		write() //simpan data dari scheduler atau korektor

3	*Eksekusor* : class objek yang mengolah data masukan dari korektor	N : integer //jumlah konsumen
		tq : date //jumlah waktu tunggu dalam antrian
		ts : date //jumlah waktu tunggu layanan
		tsian: date //jumlah waktu tunggu dalam sistem
		open() //membuka log_data dari korektor
		run() //menjalankan eksekusi data masukan
		sum() //menjumlah data masukan

4	*Korektor* : class objek yang mengatur pembacaan data untuk pengolahan *eksekusor*	create() //korektor membuat tempat simpan di log_data
		open() //membuka tempat simpanan data
		append() //korektor mengisi data ke *log_data*.
		selisih() //mengurangkan data dari *log_data scheduler*

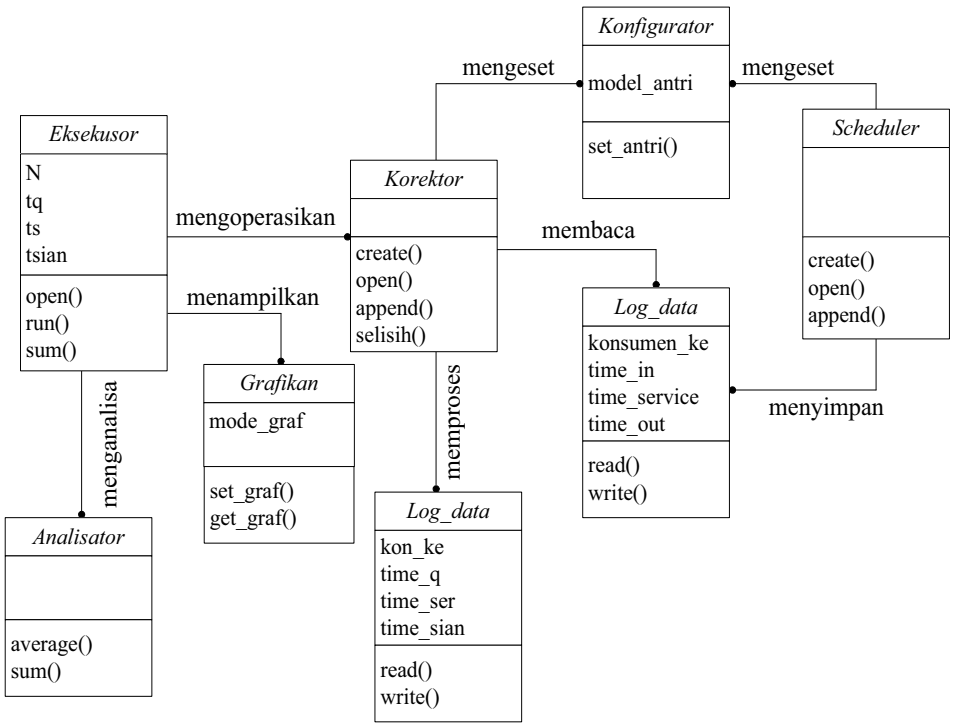
5	*Konfigurator* : class objek yang mengatur model antrian	model_antri : integer //mengatur model antrian set_antri() //mengeset model antrian

No	*Class*	Atribut – Operasi

6	*Analisator* : class objek yang mengatur analisa hasil pengolahan *eksekusor*.	average() //merata-ratakan waktu tunggu antrian, waktu tunggu layanan dan waktu tunggu sistem
		sum() // menjumlahkan waktu tunggu antrian, waktu tunggu layanan dan waktu tunggu sistem

7	*Grafikan* : class objek yang mengatur penampilan grafik hasil pengolahan *eksekusor*	mode_graf : integer //menampilkan mode grafik
		set_graf() //mengset bentuk grafik
		get_graf() //mengambil model grafik

Diagram objek pada tahapan analisis bisa dilihat pada gambar 2.3 berikut.



Gambar 2.3 : Diagram Objek Tahapan Analisis

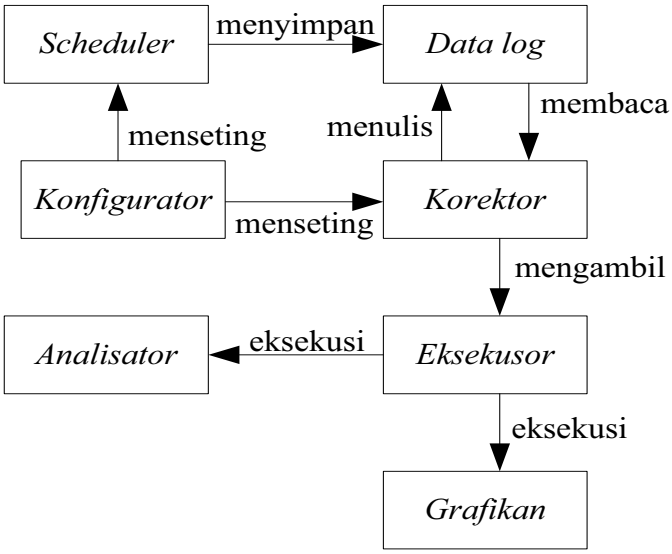
2.1.4 Model Dinamis

Model dinamis menjelaskan konsep tentang aliran kontrol, interaksi dan urutan operasi dalam suatu lingkungan objek yang aktif secara konkuren. Konsep utama dari model dinamis adalah *event* dan *status* (*state*). *Event* merepresentasikan stimulus eksternal, sedangkan *status* merepresentasikan nilai objek. Diagram status dipakai untuk menerangkan model dinamis, apabila suatu *event* datang maka status

berikutnya ditentukan oleh status sekarang dan jenis *event*. Perubahan ini dinamakan transisi. Model dinamis biasanya digambarkan dengan 3 buah diagram yaitu *event trace*, *event flow* dan diagram status. Adapun pembahasan yang akan dilakukan yaitu *event flow* dan diagram status sedangkan *event trace* tidak dibahas dikarenakan sudah terwakili pada diagram status.

2.1.4.1 *Event Flow* pada Sistem *Pro-que*

Sistem *pro-que* yang berjalan pada sistem lain, sehingga sistem *pro-que* merupakan suatu sistem yang mengolah antrian yang terjadi pada sistem lain. Dengan demikian, diagram aliran *event* yang ada pada sistem pengolahan *pro-que* bisa dilihat pada gambar 2.4 berikut.



Gambar 2.4 : *Event Flow* pada Sistem *Pro-que*

Scheduler memasukan data ke *log_data* untuk disimpan. *Log_data* akan memberikan sinyal ke *korektor* untuk baca dan tulis data. *Eksekusor* mengeksekusi data lewat *korektor*, *korektor* mengirimkan data dan *eksekusor* mengambil data dari *korektor*. *Konfigurator* mengirimkan setingan model antrian ke *scheduler*, dan *korektor*. Sehingga *scheduler* dan *korektor* akan menyimpan dan mengambil data berdasarkan setingan model antrian dari *konfigurator*. *Eksekusor* mengirimkan sinyal ke *analisator* untuk menampilkan hasil pengolahan yang berupa data-data analisa. Dan *eksekusor* mengirimkan hasil pengolahan ke *grafikan* untuk menampilkan hasil pengolahan dengan cara grafik.

2.1.4.2 Diagram Status

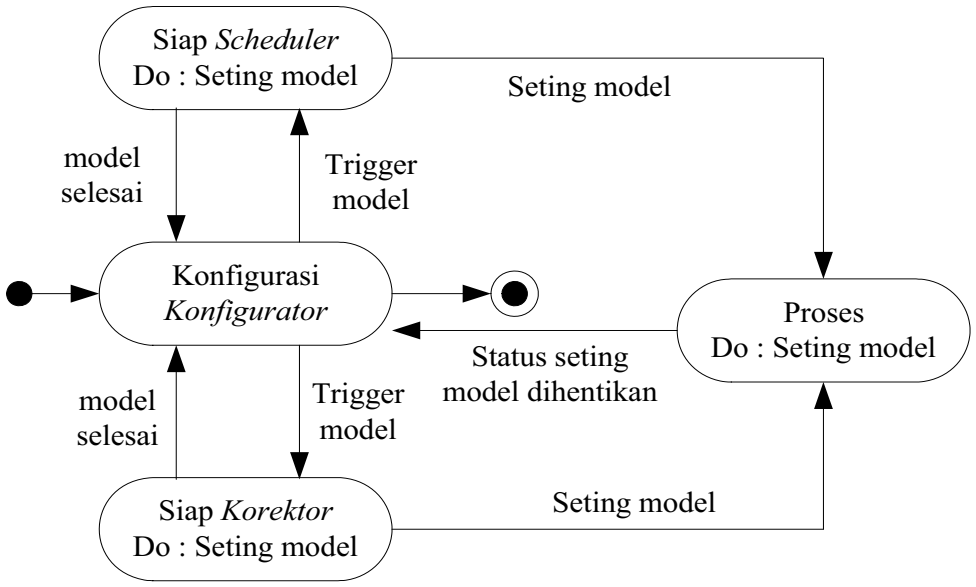
Diagram status sistem *pro-que*, terdiri dari lima buah status, yaitu status pengambilan model antrian, status simpan data, status pengambilan data, status eksekusi, dan status menampilkan hasil. Dari kelima status tersebut dikelompokan menjadi 3 status berdasarkan pada kelompok objek yang sudah diterangkan pada sesi tentang model objek. Pertama yaitu status konfigurasi menggambarkan

keadaan awal sistem yang membuat, membuka, menyimpan atau mengisi data input, dan menseting model antrian. Kedua yaitu status eksekusi merupakan keadaan sistem saat terjadi proses eksekusi pengolahan yang diawali dengan membaca atau mengambil data input. Ketiga yaitu status tampil merupakan keadaan sistem menampilkan hasil pengolahan. Keterkaitan kelompok model objek dengan status dapat dilihat pada tabel 2.2 berikut.

Tabel 2.2 : Keterkaitan Model Objek dengan Status

No	Nama Model Objek	Nama Status	Diagram Status yang Terlibat
1	Pelaksana masukan data	Konfigurasi	▪ Pengambilan model antrian ▪ Simpan data ▪ Pengambilan data
2	Pelaksana pengolahan data	Eksekusi	▪ Eksekusi
3	Pelaksana tampilan hasil pengolahan	Tampil	▪ Menampilkan hasil

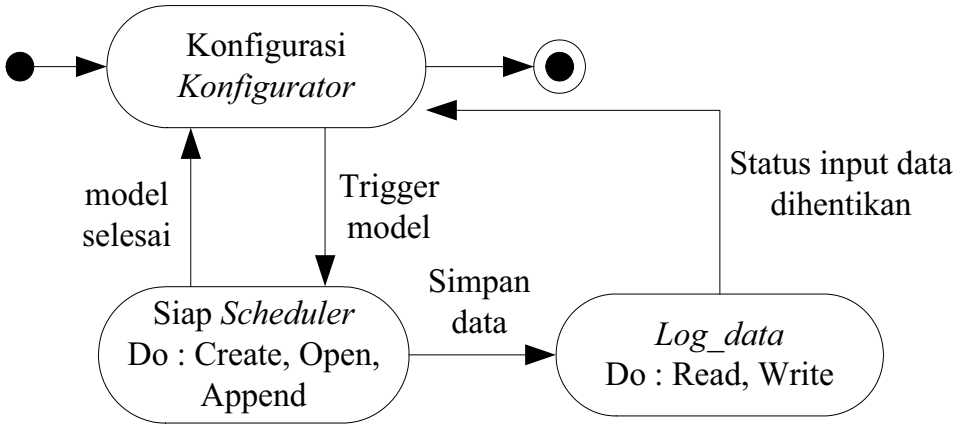
Dalam eksekusi terhadap suatu data tidak hanya data saja yang diperlukan tetapi model antrian juga diperlukan. Dengan demikian, *konfigurator* diperlukan untuk menseting model antrian. Model antrian dikirimkan ke *scheduler* untuk menentukan jenis data *input* berdasarkan model antrian yang diterima dari *konfigurator*. Begitupun *korektor* menerima setingan model antrian dari *konfigurator* untuk memproses data atau mengambil data berdasarkan model antrian. Selanjutnya status menseting model antrian dihentikan. Lihat gambar 2.5 berikut.



Gambar 2.5 : Diagram Status Pengambilan Model Antrian

Pada saat mulai untuk menyimpan data masukan, status *konfigurator* dalam keadaan proses yang artinya mengeset model antrian ke *scheduler*. Status *scheduler* dalam keadaan siap yang artinya *scheduler* menerima setingan model

antrian. Selanjutnya *scheduler* akan meng-*create*, meng-*open*, dan meng-*append* sebuah tempat penyimpanan data *input*. Selanjutnya *scheduler* menerima input data yang dimasukan oleh *emulator* atau operator. Operator memasukan data input berdasarkan suatu *event* yang terjadi pada sistem antrian. *Event* tersebut yaitu konsumen yang datang ke antrian, waktu kedatangan konsumen ke antrian, waktu konsumen mendapatkan layanan, dan waktu keberangkatan konsumen. Data *input* yang diterima oleh *scheduler* dari operator disimpan ke tempat penyimpanan yang sebelumnya sudah dipersiapkan yaitu *log_data*. Untuk *log_data* mempunyai aksi yaitu *read* dan *write*. Aksi *read* yaitu membaca setiap input yang diterima dari *scheduler* sedangkan aksi *write* yaitu menuliskan atau menyimpan data input yang telah dibaca oleh *log_data*. Selanjutnya status input data dihentikan jika operator tidak memasukan data input. Lihat gambar 2.6 berikut.

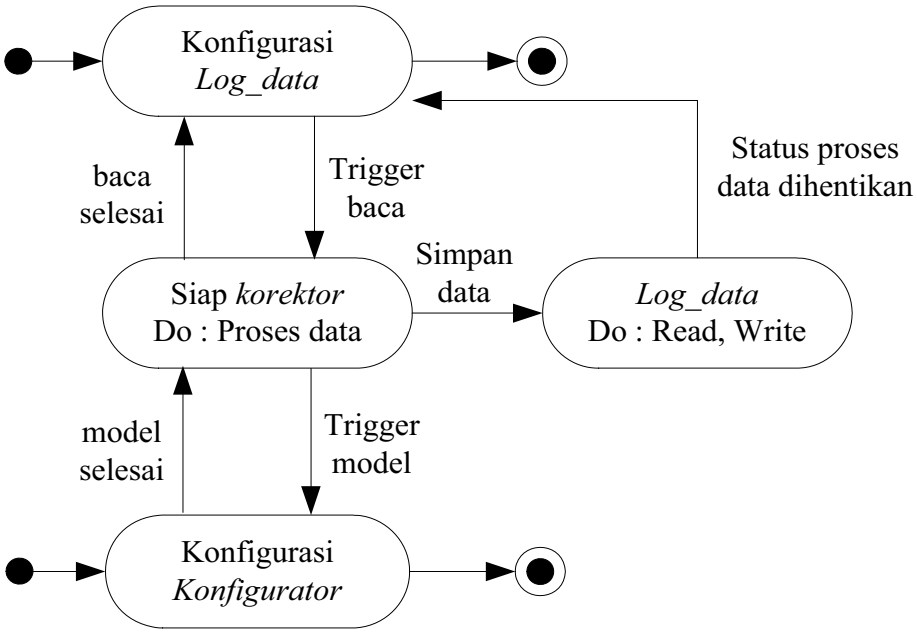


Gambar 2.6 : Diagram Status Simpan Data

Proses pengambilan data yang dilakukan oleh *korektor* akan digunakan untuk proses pengolahan *pro-que*. *Korektor* menerima setingan model antrian dari *konfigurator* yang dalam status konfigurasi, disamping itu *korektor* membaca data yang tersimpan di *log_data* yang dalam status konfigurasi. Berdasarkan setingan model antrian yang diterima dari *konfigurator*, *korektor* melakukan operasi pengambilan data dari *log_data*. Operasi yang dilakukan *korektor* yaitu : meng-*create*, meng-*open*, dan meng-*append* sebuah tempat penyimpanan yang dijadikan sebagai hasil proses pengambilan data oleh *korektor*. Disamping itu, operasi *korektor* yaitu selisih yang berarti mengurangkan data-data input dari *scheduler*, kecuali data konsumen yang masuk sistem antrian. Hasil pengolahan dari *korektor* akan disimpan ke *log_data* dengan kriteria sebagai berikut.

- Nama tempat penyimpanan hasil pengolahan *korektor* diberi nama yang berbeda dengan hasil data input *scheduler*.
- Data-data hasil pengolahan *korektor* disimpan pada suatu lokasi yang beratribut berikut : kon_ke, time_q, time_ser, dan time_sian.
- Data-data yang diperoleh seperti time_q, time_ser, dan time_sian diperoleh dari operasi selisih, sedangkan kon_ke diperoleh dari konsumen_ke dari data-data input *scheduler*.

Operasi *log_data korektor* yaitu *read* dan *write* yang mempunyai arti yang sama dengan operasi *log_data* dari *scheduler*. Dan data-data yang diperoleh oleh *korektor* akan digunakan untuk pengolahan oleh *eksekusor*. Lihat gambar 2.7 berikut.



Gambar 2.7 : Diagram Status Pengambilan Data

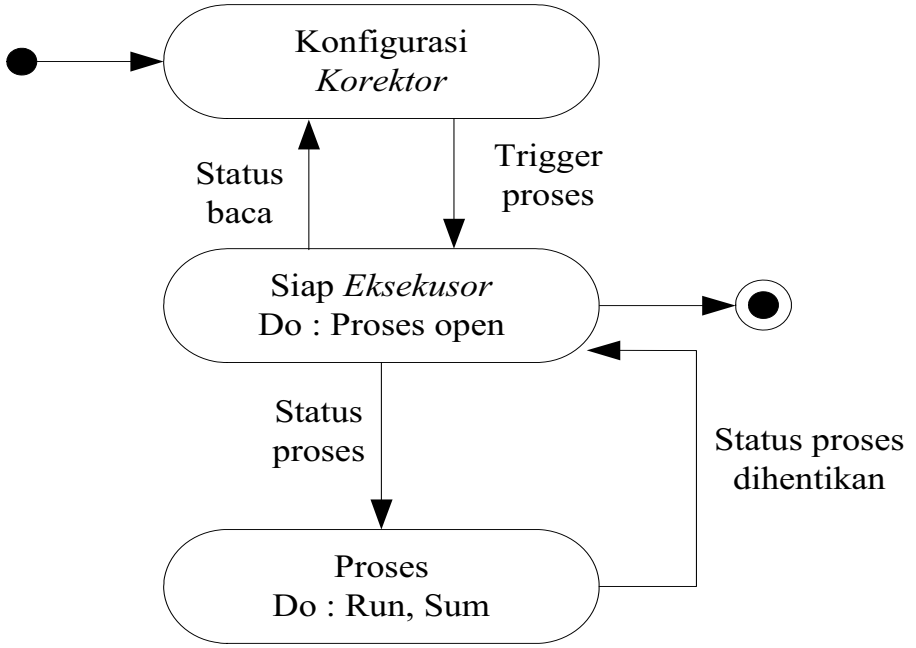
Proses pengolahan data-data input dari *korektor* akan dilakukan oleh *eksekusor*. Pengolahan ini bertujuan untuk melihat suatu fungsi kriteria yang ditetapkan oleh *pro-que*. Status *korektor* dalam keadaan konfigurasi yang kemudian menerima status baca dari *eksekusor*, selanjutnya *korektor* membangkitkan proses yang akan diterima oleh *eksekusor*. Selanjutnya status *eksekusor* dalam keadaan siap dan diikuti oleh operasi open yaitu membuka data-data yang dikirimkan oleh *korektor* yang kemudian dilanjutkan ke status proses. Dalam status proses *eksekusor* mempunyai operasi run berarti mengeksekusi data-data *input* dari *korektor* dan operasi sum yang berarti jumlahkan masukan data dari *korektor* yang berdasarkan fungsi kriteria yang telah ditetapkan. Selanjutnya status proses dihentikan dari *eksekusor*. *Eksekusor* akan mengirimkan hasil pengolahan ke objek berikutnya yaitu : *analisator*. Atribut operasi yang dimiliki oleh *eksekusor* yaitu :

- N yang mempunyai arti jumlah konsumen dalam suatu sistem antrian.
- tq yang berarti jumlah waktu tunggu dalam antrian.
- ts yang berarti jumlah waktu tunggu dalam layanan.
- tsian yang berarti jumlah waktu tunggu sistem antrian.

Hasil pengolahan *eksekusor* berdasarkan fungsi kriteria yang nantinya dikirimkan ke *analisator* yaitu :

- Jumlah konsumen dalam antrian.
- Jumlah waktu tunggu dalam antrian.
- Jumlah waktu tunggu dalam layanan.
- Jumlah waktu tunggu sistem antrian.

Bisa dilihat pada gambar 2.8 berikut.



Gambar 2.8 : Diagram Status Eksekusi

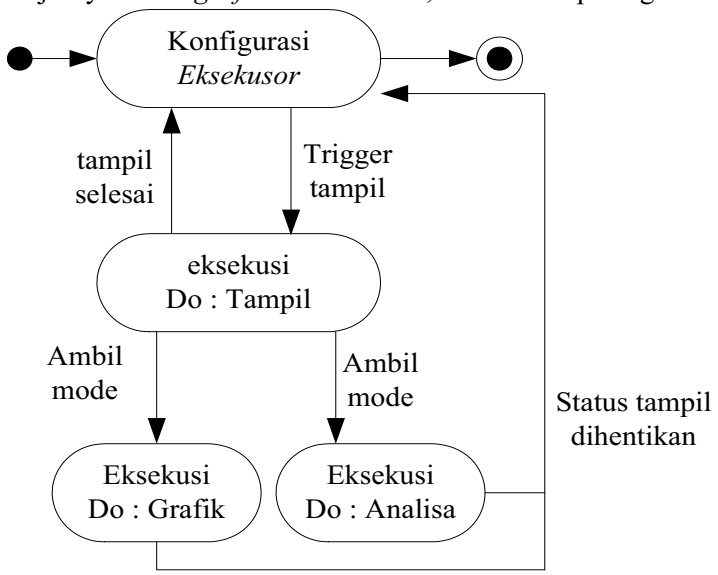
Eksekusor melakukan pengolahan data yang diperlukan dan bertugas untuk menangani hasil pengolahan. *Eksekusor* mengirimkan hasil pengolahan data yang berupa analisa ke *analisator*. *Analisator* mempunyai operasi sebagai berikut :

- Sum yang berarti menerima sejumlah informasi yang diolah oleh *eksekusor* dengan operati atribut yang sama yaitu : operasi sum.
- Average yang berarti merata-ratakan sejumlah informasi yang diterima oleh *analisator* dari *eksekusor*.

Selanjutnya status *analisator* dihentikan. Disamping itu, *eksekusor* mengirimkan hasil pengolahan data yang berupa grafik ke *grafikan*. *Grafikan* mempunyai atribut yaitu mode_graf yang berarti menyediakan mode grafik dari hasil pengolahan *eksekusor* dan mempunyai operasi sebagai berikut :

- Set_graf yang berarti menseting mode grafik.
- Get_graf yang berarti mengambil setingan grafik untuk ditampilkan sebagai informasi hasil pengolahan *eksekusor* dalam bentuk grafik.

Selanjutnya status *grafikan* dihentikan, bisa dilihat pada gambar 2.9 berikut.

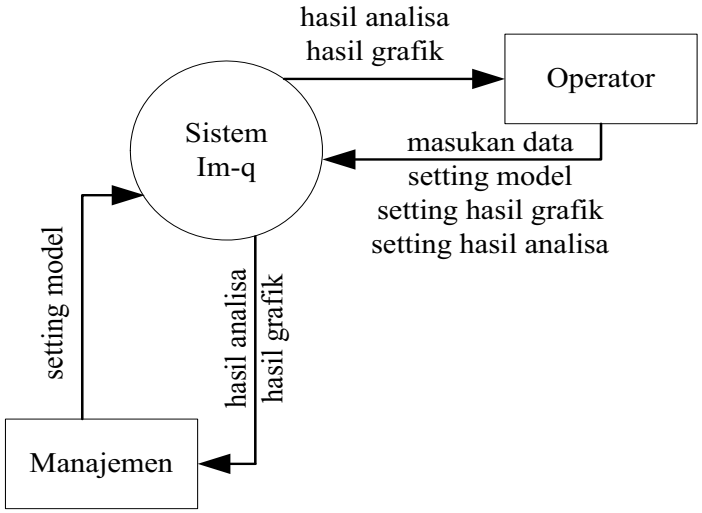


Gambar 2.9 : Diagram Status Menampilkan Hasil

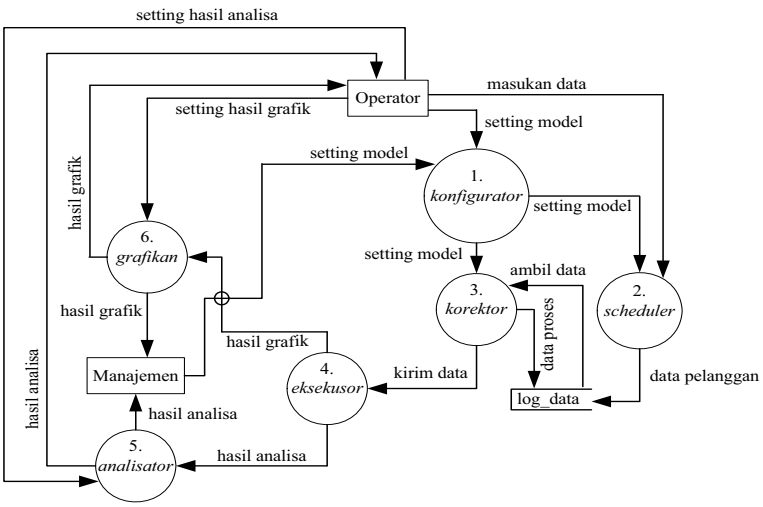
2.1.5 Model Fungsional

2.1.5.1 Diagram Konteks Sistem *Pro-que*

Sistem *pro-que* berhubungan dengan entitas luar yaitu : pemakai (operator, manajemen). Operator akan mengirimkan data untuk konfigurasi (*log_data*, *setting* model, dan *setting* hasil). Respon sistem *pro-que* akan mengirimkan hasil pengolahan, berupa data hasil eksekusi. Data hasil bisa berbentuk analisa dan grafik yang akan digunakan oleh manajemen. Operator menerima hasil pengolahan berdasarkan data-data masukan yang berupa waktu kedatangan, waktu pelayanan, waktu keberangkatan dan jumlah konsumen yang antri, lihat gambar 2.10.



Gambar 2.10 : Diagram Konteks Sistem *Pro-que*



Gambar 2.11 : DAD Level 1 Sistem *Pro-que*

4.5.1 Diagram Aliran Data Level-1 Sistem *Pro-que*

Ada enam buah proses utama yang menggambarkan tahapan dalam pelaksanaan pengolahan data pada *pro-que*, yaitu *konfigurator*, *scheduler*, *korektor*, *eksekusor*, *analisator*, dan *grafikan*. Seperti terlihat pada gambar 2.11.

Disamping itu, ada 1 *datastore* yang terdapat dalam sistem *pro-que* yaitu *log_data*. Dalam Log_data ada 2 tabel yang dibangkitkan secara dinamik dan digunakan untuk menyimpan data masukan, adapun tabel pertama yaitu : tabel yang digunakan oleh *scheduler* untuk menyimpan data input yang dilakukan oleh operator dan dikatakan sebagai data konsumen yang terdiri dari :

1. konsumen yaitu suatu entitas luar yang masuk dalam suatu antrian sistem.
2. waktu kedatangan yaitu waktu yang dibutuhkan oleh konsumen pada saat masuk antrian sistem.
3. waktu layanan yaitu waktu yang dibutuhkan oleh server pada saat melayani konsumen.
4. waktu keberangkatan yaitu waktu konsumen keluar atau berangkat dari sistem antrian.

Kemudian tabel kedua yaitu tabel yang digunakan oleh *korektor* untuk menyimpan hasil pengolahan yang dilakukan oleh *korektor*. Dan hasil pengolahan data *korektor* digunakan untuk proses pengolahan data yang dilakukan oleh *eksekusor*. Data hasil yang diperoleh oleh *korektor* yaitu :

1. konsumen yang masuk dalam antrian.
2. waktu tunggu dalam antrian yaitu waktu tunggu yang digunakan konsumen dalam antrian.
3. waktu tunggu dalam layanan yaitu waktu tunggu yang digunakan konsumen dalam layanan.
4. waktu tunggu sistem antrian yaitu waktu tunggu yang digunakan konsumen dalam sistem antrian.

2.2. Tahap Perancangan Sistem

Perancangan perangkat lunak *pro-que* menggunakan hasil analisis sistem *pro-que* yang telah diterangkan sebelumnya sebagai bahan acuan. Sedangkan perancangan perangkat lunak akan menjadi sebuah panduan dalam implementasi perangkat lunak. Beberapa asumsi atau batasan dari perancangan adalah :

1. Sistem operasi yang akan dipakai yaitu *Microsoft Windows 98* ke atas.
2. Perangkat lunak *pro-que* memanfaatkan beberapa tabel sebagai tempat penyimpanan data masukan serta hasil pengolahan data *pro-que*.
3. Memanfaatkan fasilitas antar muka untuk operator dalam mengoperasikan *pro-que* dan untuk manajemen dalam menganalisis hasil pengolahan *pro-que* yang dijadikan acuan untuk kebijakan sistem antrian.
4. Model antrian yang dibangun untuk kebutuhan implementasi antrian dengan kasus satu server dan satu antian.
5. Perangkat lunak belum dilengkapi dengan fasilitas cetak.
6. Masukan data dari sistem lain masih dilakukan secara manual, sehingga dibutuhkan operator.

Pada tahap ini dibahas mengenai hal-hal berikut :

1. Perancangan objek, penyempurnaan terhadap model objek hasil tahapan analisis.
2. Perancangan modul, berisi pembagian sub sistem berdasarkan kesamaan fungsi.

2.2.1 Perancangan Objek

Pada tahap perancangan, model objek hasil tahap analisis disempurnakan dengan menambahkan aspek *inheritance* dan penambahan objek lain untuk membentuk aplikasi secara keseluruhan. Adapun penyempurnaan terhadap model objek hasil analisis yaitu :

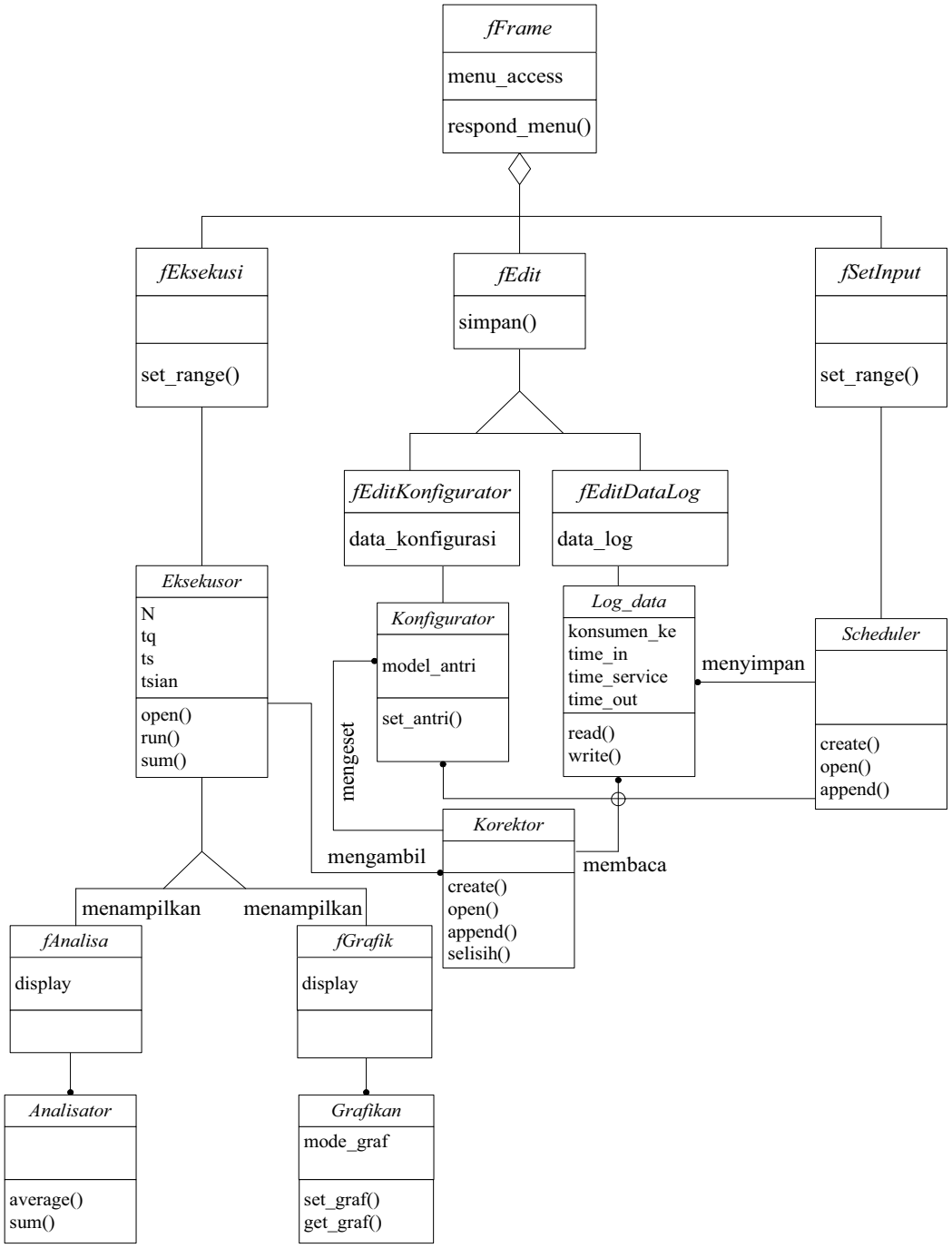
1. Penambahan objek untuk penyedia fasilitas antarmuka dengan pemakai, berupa layar antarmuka yaitu *window*. Beberapa kandidat *window* penyedia fasilitas antarmuka ini yaitu :
   a. *Window* untuk menseting model antrian.
   b. *Window* untuk memasukan data (*input* data).
   c. *Window* untuk pengambilan data input yang digunakan untuk pengolahan.
   d. *Window* untuk pengeksekusian data.
   e. *Window* untuk menampilkan informasi bentuk analisa dan bentuk grafik.
2. Model objek juga akan memunculkan objek aplikasi. Objek aplikasi merepresentasikan organisasi aplikasi secara keseluruhan.
3. Pengaturan penamaan. Objek *visual*, yaitu *window* mempunyai awalan **f** (contoh fModel), objek non-visual dari domain masalah tidak berawalan.

Ada *Class* tambahan yang terdapat tahap pada perancangan *pro-que* bisa dilihat pada tabel 2.3. Dengan demikian, *class* yang terdapat pada tahap perancangan yaitu ada tujuh *class* utama proses yang dihasilkan juga pada tahap analisis perangkat lunak *pro-que* dan tujuh *class* tambahan untuk kebutuhan perancangan antarmuka. Tujuh *class* utama proses yang terdapat di tahap perancangan perangkat lunak *pro-que* terdapat juga di tahap analisis perangkat lunak *pro-que* yang bisa dilihat pada tabel 2.1. Sedangkan diagram objek tahap perancangan perangkat lunak *pro-que*

bisa dilihat pada gambar 2.12 sebagai penyempurnaan dari diagram objek tahap analisis perangkat lunak *pro-que*.

Tabel 2.3 : Deskripsi *class* tambahan tahap perancangan

No	Class	Atribut - Operasi
1	fFrame : *class* objek yang menampilkan *window* utama dan menu utama	menu_access : menu, respond_menu() //respon terhadap menu
2	*fEdit :* class *objek yang digunakan untuk pengeditan*	simpan() //simpan data dan konfigurasi model antrian
3	*fSetInput :* class *objek yang mempersiapkan input data.*	set_range() //mempersiapkan input data ke *scheduler*
4	*fEditKonfigurator : class objek yang digunakan untuk mempersiapkan dan pengeditan terhadap model antrian*	data_konfigurasi : struct_data konfigurasi
5	*fEditDataLog :* class *objek yang mempersiapkan log data.*	data_log : menseting tempat penyimpanan log_data
6	*fEksekusi :* class *objek yang menampilkan window eksekusi.*	set_range() //menseting area eksekusi
7	*fAnalisa :* class *objek yang menampilkan* window *analisa.*	display : menu untuk menampilkan hasil pengolahan berupa analisa
8	*fGrafik :* class *objek yang menampilkan* window *grafik.*	display : menu untuk menampilkan hasil pengolahan berupa grafik



Gambar 2.12 : Diagram Objek Tahap Perancangan

2.2.2 Perancangan Modul

Berdasarkan suatu model objek, model dinamis, dan model fungsional hasil tahap analisis serta penyempurnaan objek maka hasil perancangan ada 2 modul perangkat lunak *pro-que*. Setiap modul akan melibatkan satu atau lebih proses operasi dari

class pelaksana. Tabel 2.4 berikut yang menjelaskan keterlibatan antara kedua modul dengan *class-class*-nya.

Tabel 2.4 : Class yang terlibat dalam Modul Rancangan

No	Nama Modul	Deskripsi	Nama Class
1	Konfigurasi	Modul yang dirancang untuk keperluan berikut : ▪ menseting model antrian ▪ membuat tabel simpanan ▪ membaca dan menulis tabel simpanan ▪ memasukan data konsumen ▪ memvalidasi dan mengolah data konsumen ▪ mengirimkan data pengolahan ke modul eksekusi	▪ fFrame ▪ fEdit ▪ fSetInput ▪ fEditKonfigurator ▪ fEditDataLog ▪ konfigurator ▪ scheduler ▪ log_data ▪ korektor
2	eksekusi	Modul yang dirancang untuk keperluan berikut : ▪ menerima data pengolahan modul konfigurasi ▪ mengeksekusi data ▪ menampilkan hasil pengolahan data berupa analisa ▪ menampilkan hasil pengolahan data berupa grafik	▪ fFrame ▪ fEksekusi ▪ fAnalisa ▪ fGrafik ▪ korektor ▪ eksekusor ▪ analisa ▪ grafik

2.2.2.1 Modul Konfigurasi

Modul konfigurasi adalah bagian perangkat lunak untuk menyiapkan pengolahan pada sistem *pro-que*. Modul ini berisi objek penyedia sarana interaksi dengan pemakai dan objek pelaksanaan : input data, simpan data, baca atau ambil data, dan penyedia setingan model antrian. Dalam modul konfigurasi ada beberapa sub-modul dan proses yang dilakukan oleh sub-modul tersebut. Proses pada sub-modul tersebut bisa berupa prosedur, fungsi, atau koding program yang melakukan suatu aksi terhadap sub-modul. Berikut tabel 2.5 yang menjelaskan sub-modul, proses yang terjadi dan *class* yang terlibat.

Tabel 2.5 : Proses sub-modul dari Modul Konfigurasi

ID	Sub-Modul	*Class* yang terlibat	Fungsi dari Class yang terlibat
MK1	Akses	▪ fFrame	*respond_menu()* //penyedia layanan akses menu utama.
MK2	seting_model	▪ fEdit ▪ fEditKonfigurator ▪ *konfigurator*	*simpan()* //penyedia layanan editing untuk fEditKonfigurator dan fEditDataLog. *set_antri()* //mengeset model antrian dan mengirimkan model antrian ke scheduler dan korektor.

ID	Sub-Modul	*Class* yang terlibat	Fungsi dari Class yang terlibat
MK3	input_data	▪ fSetInput ▪ *scheduler* ▪ *korektor*	*set_range()* //mempersiapkan input data ke *scheduler*. *create()* //scheduler membuat tempat simpanan di log_data. *open()* //schedelur membuka log_data. *append()* //scheduler memasukan data ke log_data. *create()* // korektor membuat tempat simpan di log_data. *open()* //korektor membuka tempat simpan data *append()* //korektor mengisi data ke *log_data*. *selisih()* //mengurangkan data dari *log_data scheduler*
MK4	simpan_data	▪ fEdit ▪ fEditDataLog ▪ *DataLog*	*read()* //baca data dari scheduler atau korektor. *write()* //simpan data dari scheduler atau korektor.

2.2.2.2 Modul Eksekusi

Modul eksekusi adalah bagian perangkat lunak yang melaksanakan proses eksekusi terhadap data korektor. Modul eksekusi berisi *class* pelaksana proses pengolahan dan *class* penyedia layanan interaksi dengan pemakai. Dalam modul eksekusi ada beberapa sub-modul dan proses yang dilakukan oleh sub-modul tersebut. Proses pada sub-modul tersebut bisa berupa prosedur, fungsi, atau koding program yang melakukan suatu aksi terhadap sub-modul. Berikut tabel 2.6 yang menjelaskan sub-modul, proses yang terjadi dan *class* yang terlibat.

Tabel 2.6 : Proses sub-modul dari Modul Eksekusi

ID	Sub-Modul	*Class* yang terlibat	Fungsi dari Class yang terlibat
ME1	akses	▪ fFrame	*respond_menu()* //penyedia layanan akses menu utama.
ME2	eksekusi	▪ fEksekusi ▪ *korektor* ▪ *eksekusor*	*set_range()* //menseting area eksekusi. *open()* //membuka log_data dari korektor. *run()* //menjalankan eksekusi data masukan. *sum()* //menjumlah data masukan.

ID	Sub-Modul	*Class* yang terlibat	Fungsi dari Class yang terlibat
ME3	tampil	▪ fAnalisa ▪ fGrafik ▪ *analisator* ▪ *grafikan*	*average()* //merata-ratakan waktu tunggu antrian, waktu tunggu layanan dan waktu tunggu sistem. *sum()* // menjumlahkan waktu tunggu antrian, waktu tunggu layanan dan waktu tunggu sistem. *set_graf()* //mengset bentuk grafik *get_graf()* //mengambil model grafik

2.3. Tahap Implementasi Sistem

Perangkat lunak *pro-que* diharapkan menjadi bagian sistem lain yang lebih besar yang diimplementasikan pada sistem operasi *Microsoft Windows 98* ke atas dengan bahasa pemrograman yang berorientasi objek. Sedangkan tool bahasa pemrograman yang berorientasi objek bisa pakai *Microsoft Visual Basic 6.0*. Adapun kebutuhan untuk pengopersian database bisa gunakan *Microsoft Access 2000*. Untuk menampilkan grafik bisa gunakan *Microsoft Chart 2000*. Disamping itu, untuk menghubungkan antara file database yang dibuat dan sistem operasi menggunakan *engine database* yaitu ODBC (*Open Database Connections*).

Perangkat keras untuk prototipe perangkat lunak *pro-que* yang bisa menjalankan tool *Microsoft Visual Basic 6.0*, *Microsoft Access 2000*, *Microsoft Chart 2000*, dan sistem operasi *Windows 98* ke atas yaitu :

1. Prosesor Pentium III (AMD-Duron).
2. Memori 128 MB.
3. Card montior 16 MB (minimum 1 MB).
4. Layar Monitor SVGA (minimum 800 x 600 pixels).
   Kebutuhan perangkat diatas tidak bersifat dimutlak, yang terpenting bisa menjalankan kebutuhan untuk pembangunan perangkat lunak *pro-que*.

2.3.1 Batasan Implementasi

Beberapa batasan dalam implementasi perangkat lunak *pro-que* pada kesempatan ini yaitu :

1. Menggunakan file database yang bernama *log_data.mdb* untuk membuat tabel dinamik sebagai tempat penyimpanan data masukan dan sebagai tempat data yang diambil untuk dieksekusi.
2. Menggunakan tabel permanen yang bernama *model1* untuk membangun tabel dinamik dan berguna untuk menyimpan data masukan yang dimasukan oleh operator, dengan struktur sebagai berikut :
   Model1

Field Name	Data Type	Field Size	Dec
konsumen_ke	Number	Integer	0
time_in	Date/Time	Long Time	0

time_service	Date/Time	Long Time	0
time_out	Date/Time	Long Time	0

3. Menggunakan tabel permanen yang bernama *amdat1* untuk membangun tabel dinamik dan berguna untuk menyimpan data yang diambil dari simpanan data untuk keperluan eksekusi, dengan struktur sebagai berikut :
   Amdat1

Field Name	Data Type	Field Size	Dec
kon_ke	Number	Integer	0
time_q	Date/Time	Long Time	0
time_ser	Date/Time	Long Time	0
time_sian	Date/Time	Long Time	0

4. Implementasi algoritma yang digunakan :
   - Input data dari sistem antrian, algoritma ini untuk menangani input data.
   - Ambil data untuk keperluan eksekusi, algoritma ini untuk mengambil data input dari scheduler yang akan diambil oleh korektor untuk kebutuhan pengolahan data eksekusor.
   - Eksekusi data, alagoritma ini untuk mengolah data input dari korektor yang dilakukan oleh eksekusor.

   Sedangkan algoritma untuk konfigurator diimplementasikan berdasarkan kasus untuk server dan antrian yang ada, sehingga akan terjadi pemilihan model antrian.

2.3.2 Implementasi Modul Konfigurasi

Modul konfigurasi dioperasikan melalui menu utama. Pengaturan masukan data dari sistem antrian dan pengambilan data untuk eksekusi yang akan dilakukan dengan menggunakan *window* konfigurasi. *Window* konfigurasi dalam implementasinya dibagi atas, pertama *window* konfigurasi yang digunakan untuk masukan data dari sistem antrian dan masukan data dilakukan secara manual oleh seorang operator. Dan kedua *window* konfigurasi untuk pengambilan data masukan yang akan digunakan oleh *korektor*. Oleh karena itu, *window* konfigurasi merupakan suatu area dengan *control* standar seperti *textbox*, *label*, *DBGrid*, dan lain-lain. Sedangkan dari sisi pengembang *window* konfigurasi merupakan objek yang mempunyai atribut untuk mengambil data masukan dari operator dan inisialisasi operasi dilakukan. Pengaksesan data konfigurasi dapat juga melalui pemanggilan *file* log_data.

Semua komponen program berada dalam satu atau beberapa file yang memuat baik kode maupun komponen lain (user *class*, *window*, *window* konfigurasi, menu) dari program. Komponen dari modul konfigurasi yang berada dalam file yang berupa form yaitu :

- FrmMenUtama.frm
- FrmDialogInModAntri.frm
- FrmDialogAmModAntri.frm
- FrmDatInMod1.frm
- FrmDatAmMod1.frm

Tabel 2.7 : Implementasi komponen pembentuk Modul Konfigurasi

No	Nama – Jenis	Deskripsi	File yang terlibat
1	*Data_log* – *class*	*Class data_log* yaitu *class* yang digunakan untuk menyimpan data masukan scheduler dan data pengambilan korektor	FrmDatInMod1.frm, FrmDatAmMod1.frm
2	Scheduler – *class*	*Class scheduler* yaitu *class* yang digunakan untuk menyiapkan tempat penyimpanan data input	FrmDatInMod1.frm.
3	Konfigurator – *class*	*Class* konfigurator yaitu *class* yang digunakan untuk mensetting model antrian	FrmDialogInModAntri.frm, FrmDialogAmModAntri.frm.
4	Korektor – *class*	*Class* korektor yaitu *class* yang digunakan untuk proses pengambilan data dan juga proses pengurangan terhadap data input *scheduler*	FrmDatAmMod1.frm
5	fFrame – user object	*Window* utama yaitu objek yang menampilkan *window* utama dan menu utama	FrmMenUtama.frm
6	fEdit – user object	*Window* Edit yaitu suatu objek yang digunakan untuk pengeditan	FrmDialogInModAntri.frm, FrmDialogAmModAntri.frm, FrmDatInMod1.frm, FrmDatAmMod1.frm
7	fEditKonfigurator – user object	*Window* EditKonfigurator yaitu objek yang digunakan untuk mempersiapkan dan pengeditan terhadap	FrmDialogInModAntri.frm, FrmDialogAmModAntri.frm.

No	Nama – Jenis	Deskripsi	File yang terlibat
		model antrian	
8	fEditDataLog – user object	*Window* EditDataLog yaitu objek yang mempersiapkan log data.	FrmDatInMod1.frm, FrmDatAmMod1.frm
9	fSetInput – user object	*Window* SetInput yaitu objek yang mempersiapkan input data.	FrmDatInMod1.frm.

2.3.3 Implementasi Modul Eksekusi

Implementasi modul eksekusi dapat dioperasikan melalui menu utama. Implementasi *event click* START yang menampilkan *window* sebagai fasilitas untuk eksekusi. Tombol *command button RUN* memulai operasi eksekusi data berdasarkan pengambilan data oleh korektor.

Hasil eksekusi akan dikirimkan melalui *window* analisis. Dari *window* analisis maka seorang manajemen bisa mengetahui keadaan antrian pada waktu antrian berakhir melalui *pro-que*. Dan diharapkan hasil implementasi *pro-que* bisa dijadikan sebagai petunjuk untuk menciptakan suasana antrian yang lebih baik. Adapun komponen eksekutor yang dibuat melalui pemrograman yang berupa file form yaitu :

- FrmMenUtama.frm
- FrmEksekusi.frm
- FrmAnalisa.frm
- FrmGrafik.frm
- FrmDatAmMod1.frm.

Tabel 2.8 : Implementasi komponen pembentuk Modul Eksekusi

No	Nama - Jenis	Deskripsi	File yang terlibat
1	Eksekusor – *class*	*Class* Eksekutor yaitu *class* yang berguna untuk mengeksekusi data masukan dari *korektor* dan mengirimkan hasil ke *analisator*	FrmEksekusi.frm
2	Analisator – *class*	*Class* Analisator yaitu *class* yang berguna untuk menampilkan hasil pengolahan data berupa analisa	FrmAnalisa.frm
3	Grafikan – *class*	*Class* Grafikan yaitu *class* yang berguna untuk menampilkan hasil pengolahan data berupa grafik	FrmGrafik.frm
4	Korektor – *class*	*Class* korektor yaitu class yang digunakan untuk proses pengambilan data dan juga proses pengurangan terhadap data input	FrmDatAmMod1.frm

		scheduler	
5	fFrame – user object	*Window* utama yaitu objek yang menampilkan *window* utama dan menu utama	FrmMenUtama.frm
6	fEksekusi – user object	*Window* Eksekusi yaitu objek yang menampilkan window eksekusi	FrmEksekusi.frm
7	fAnalisa – user object	*Window* Analisa yaitu objek yang menampilkan *window* analisa	FrmAnalisa.frm
8	fGrafik – user object	*Window* Grafik yaitu objek yang menampilkan *window* grafik	FrmGrafik.frm

3. Penutup

Berdasarkan implementasi perangkat lunak *pro-que*, maka pihak manajemen sistem antrian bisa menentukan keputusan bahwa sistem antrian yang dapat memenuhi kepuasan konsumen didasarkan atas informasi hasil analisa dari perangkat lunak *pro-que*. Pada tahap implementasi perangkat lunak *pro-que* masih menggunakan asumsi dengan kasus server dan antrian.

Daftar Pustaka

[Dup01] *Dupac, Jan (2001), "Hierarchy of Used Markovian Models", Available: http://cmp.felk.cvut.cz/~xdupac/.*

[Hal91] *Hall, Randolph W. (1991), "Queueing Methods for Service and Manufacturing", Prentice-Hall International Edition.*

[Lie96] *Liem, Inggriani (1996), "Pemodelan dan Simulasi", Diktat Kuliah, ITB.*

[McG94] *McGregor J.D. & Korson T.D. (1994), "Integrated Object-Oriented Testing and Development Processes", Communication of the ACM, vol. 37 – September, hal. 59 – 77.*

[Pre01] *Pressman, Roger S. (2001), "Software Engineering a Practitioner's Approach, Fifth Edition", McGraw-Hill, New York.*

[Rum91] *Rumbaugh, James; Blaha, Michael; Premerlan, William; Eddy, Frederick; Lorensen, William; (1991), "Object-Oriented Modeling and Design", Prentice-Hall, Englewood Cliffs, New Jersey.*

[Sek00] *Sekarwangi, Guntari (2000), "Pendeteksian Kondisi Operasi Sistem Dinamik Menggunakan Metode Multiple Hypothesis Testing", Tesis.*

[Sia87] *Siagian P. (1987), "Penelitian Operasional Teori dan Praktek", Universitas Indonesia.*